Quantitative and Qualitative Extensions of Event Structures

Joost-Pieter Katoen

CTIT Ph. D-thesis series No. 96-09

P.O. Box 217 - 7500 AE Enschede - The Netherlands
telephone +31-53-4893779 / fax +31-53-4893247

Centre for
Telematics and
Information
Technology

Promotiecommissie:

prof. dr. W.E. van der Linden (voorzitter)
prof. dr. ir. C.A. Vissers (promotor)
prof. dr. H. Brinksma (promotor)
dr. ir. A. Rensink (referent, Universität Hildesheim)
prof. dr. U. Herzog (Universität Erlangen-Nürnberg)
prof. dr. M. Rem (Technische Universiteit Eindhoven)
prof. dr. F.W. Vaandrager (Katholieke Universiteit Nijmegen)
prof. dr. ir. Th. Krol
dr. L. Ferreira Pires, M.Sc.

Druk: Ponsen & Looijen, Wageningen

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Katoen, Joost-Pieter

Quantitative and qualitative extensions of event
structures / Joost-Pieter Katoen. - Enschede : Centre for
Telematics and Information Tenchnology. – Ill. – (CTIT
Ph. D-thesis series, ISSN 1381-3617 ; 96-09)
Proefschrift Universiteit Twente, Enschede. - Met index,
lit. opg.
ISBN 90-365-0799-5
Trefw.: procesalgebra / systeemontwerp.

QUANTITATIVE AND QUALITATIVE EXTENSIONS OF EVENT STRUCTURES

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof.dr. Th.J.A. Popma,
volgens besluit van het College voor Promoties
in het openbaar te verdedigen op
donderdag 18 april 1996 te 15.00 uur.

DOOR

Joost-Pieter Katoen

geboren op 6 oktober 1964

te Krimpen aan den IJssel

Dit proefschrift is goedgekeurd door de promotores:

prof. dr. ir. C.A. Vissers
prof. dr. H. Brinksma

Acknowledgements

In 1987 I had the privilege to work for about 7 months at Philips Research Laboratories. Under the supervision of Pierre Jansen and Lex Augusteijn a simulator for a parallel computer was built. From then on I was captured by the phenomenon 'concurrency'.

To enlarge my practical background with some theoretical insights I spent two years as a 'twaio' at Eindhoven University of Technology. Martin Rem, together with Rob Hoogerwoord, learned me to appreciate a formal attitude to the design of programs, including concurrent ones. The spontaneously introduced HG 7.37 sessions with my roommates Berry Schoenmakers, Pieter Struik and Wim Kloosterhuis made it all work: besides the consumption of 'Bossche bollen' my interest for theory and its application(s) increased.

Back at Philips Research, I worked on the engineering and performance analysis of communication protocols, an exciting application field in which concurrency (again) plays a prominent rôle. Together with Marnix Vlot I worked on the definition of a (standard) communication system for various types of equipment in domestic environments, while having a great time in sharing a room with another 'sport-en-in-het-bijzonder-Tour-gek', Frans Sijstermans.

In the spring of 1992 I started to work under the supervision of Chris Vissers at the University of Twente. Due to the freedom he created to perform research in a pleasant and stimulating environment I was able to work on several fascinating subjects during the last four years. His comments on earlier versions of this dissertation, his view on conceptual issues, and our discussions about the relationship between formal methods and (basic) architectural concepts have been a continuous inspiration and have taught me very much. I also like to thank my other promotor, Ed Brinksma, for sharing his creativity, enthusiasm, and knowledge about concurrency and formal methods. With a small amount of information he was able to give me the right hints at the right moment to overcome technical problems. His active contributions to several parts of this dissertation are greatfully acknowledged.

This dissertation would not have its current state without the cooperation with Rom Langerak and Diego Latella. The many sessions we had together were the basis for most of the material conducted in this dissertation. Their enthusiasm, detailed comments on draft chapters, and constructive attitude were invaluable; this includes the many nice Italian dinners and lunches (even those in Castelleto) and trips to various exotic places like Barga, San Miniato, Lucca and Buti. I hope we will continue our friendship and our professional cooperation. In this respect I also like to thank Mieke Massink, Tommaso Bolognesi, Stefania Ciompi, and Maurizio Caneve for their hospitality and friendship.

Then I would like to thank my referent, Arend Rensink. His comments and recommendations have strongly improved this dissertation. Our 3-day session in Hildesheim was, despite catching a cold, very fruitful and constructive. Various colleagues have read parts of my dissertation; Pedro d'Argenio, Lex Heerink, Jan Tretmans and Luís Ferreira Pires are kindly acknowledged for their effort and comments. Boudewijn Haverkort and Victor Nicola are thanked for discussions and suggestions concerning Chapter 8. Furthermore, I like to thank some of the many people with which I had the pleasure to cooperate during the last years: Jakob Brunekreef, Wouter van den Broek, Robert Huis in 't Veld, Ron Koymans, Harro Kremer, Sjouke Mauw and (last but not least) Albert Nymeyer.

To conclude I would like to thank my family, other friends, and relatives. In particular I like to thank my parents Joost and Korrie Katoen for supporting me during my study and work, and for their love and understanding. Unfortunately, they are not privileged to be a witness of this milestone. Finally my grateful thanks go to my wife Erna for her never-ceasing support and understanding, and for reminding me together with our son Joost, that life encompasses much more than writing a dissertation.

Summary

An important application of formal methods is the specification, design, and analysis of functional aspects of (distributed) systems. Recently the study of quantitative aspects of such systems based on formal methods has come into focus. Several extensions of formal methods where the occurrence of actions can be assigned a (fixed) probability and/or the time of occurrence of actions can be constrained are known from the literature.

An important reason for enhancing formal methods with quantitative notions is to facilitate the analysis of performance characteristics of system designs. In this way the efficiency of design alternatives can be assessed such that in early design stages designs can be rejected because of unsatisfactory performance characteristics, thus avoiding costly redesign at later stages. A formal specification incorporating quantitative aspects can also be very useful for establishing a well-understood and effective way of developing performance models, such as Markov chains and queuing networks, from system specifications.

Quantitative extensions of formal methods that are based on *interleaving* of causally independent actions have been amply investigated. Interleaving models abstract from the fact that a system is actually composed of a set of (partly) independent subsystems. The global system state is considered without due regard of its distributed nature. The system's behaviour is modelled in terms of sequences of actions that are totally ordered by precedence and in which actions of one independent subsystem are merged with actions of others.

This dissertation deals with quantitative and qualitative extensions of *event structures*, a prominent branch of partial-order, or *noninterleaving*, models for concurrency. Example extensions are the incorporation of issues like time, both real-time and stochastic of nature, urgency (timeouts), and probability. Nowadays the treatment of these concepts in noninterleaving models has only been scarcely addressed.

Noninterleaving models do not abstract from the fact that a system consists of a set of (partly) independent subsystems. The notion of global state does not play a central rôle in these models. The systems' behaviour is modelled in terms of sequences of actions that are not required to be totally ordered, but that are partially ordered. The causal dependencies between actions are reflected in this partial order.

Interleaving and noninterleaving models are *complementary* in the system's design process. In this dissertation we basically deal with noninterleaving models, but also provide the ingredients to obtain corresponding interleaving models. This facilitates the use of both types of models in a coherent way and enables a comparison with existing approaches.

Starting points for this dissertation are

- *extended bundle event structures*, an adaptation of the traditional event structures of Winskel to fit the specific requirements of multi-party synchronization and disruption, and

- *process algebras*, abstract description formalisms for distributed systems that consist of powerful composition operators.

Extended bundle event structures consist of *labelled events* modelling occurrences of actions (indicated by the labels), a *bundle* relation indicating the causal dependencies between events, and an (asymmetric) *conflict* relation modelling exclusions between events. Event structures, in particular extended bundle event structures, are treated in Chapter 2.

The bundle relation relates a set of events, the bundle set, to an event. The interpretation is that one event in the bundle set must have happened in order to enable (or cause) the event to which it is related. All events in a bundle set are required to be mutually in conflict such that only one event in a bundle set can happen. By dropping this constraint more events in a bundle set can happen and the expressivity is increased, i.e., so-called *disjunctive causality* is supported. In Chapter 3 it is investigated how *labelled partial orders* (lposets), which are used in this dissertation as underlying semantical models for event structures, can be generated when this constraint is dropped. This chapter also investigates useful transformations for the resulting model that preserve equivalence in terms of lposets and considers the incorporation of a symmetric irreflexive *interleaving relation* between events.

Event structures describe system behaviours by causal orderings (bundles) among events and their branching structure (conflicts). To facilitate the specification of timing-based systems, such as communication protocols, the concept of *time* is considered. Chapters 4, 6, and 7 treat the incorporation of time in extended bundle event structures. Real-time event structures associate a set of time instants to bundles, indicating relative time constraints between causally dependent events, and to events, modelling absolute time constraints (Chapter 4 and 7). Urgent event structures allow for the specification of minimal time constraints only, but incorporate *urgent* events, events that are forced to happen once they are enabled (Chapter 6). Urgent events are typically used to model timeouts. The generalization of deterministic time towards time of a more dynamic stochastic nature is treated in Chapter 8. Stochastic event structures attach *distribution functions* to bundles and events, rather than sets of time instants. Finally, in Chapter 9 we consider the incorporation of *probabilities* in extended bundle event structures. Probabilities are attached to events and quantify the likelihood of appearance of events once they are enabled.

Event structures are well-suited to provide a noninterleaving semantics for process algebras in a *compositional* way. That is, the interpretation of any composite behaviour expression in the process algebra is defined as a function of the interpretation of its constituents. In this dissertation we investigate whether the quantitative extensions of event structures can be used to define a noninterleaving semantics to quantitative extensions of process algebras. To that purpose we take the process algebra PA as a basis, which is in fact the international standardized process algebra LOTOS with a somewhat more concise syntax. The principles do, however, also apply to related process algebras like Milner's CCS and Hoare's CSP. For each quantitative variant of PA the noninterleaving semantics of the plain process algebra PA is tried to retain as much as possible, aiming at maximal *backwards compatibility*.

The quantitative extensions of process algebras that we consider are real-time variants that incorporate *timeout*, *watchdog* and *urgency* operators, stochastic variants in which the occurrence times of actions is constrained by *exponential*, or the more general and practical, *phase-type* distributions, and a probabilistic variant that contains an (internal) *probabilistic choice* operator. For each variant a denotational semantics in terms of the corresponding quantitative extension of extended bundle event structures is provided. This is performed in a *modular way* such that combinations (like time and probability) can be made in a rather straightforward way.

In addition, for most aforementioned process algebras an *event-based operational semantics* is presented. This operational semantics keeps track of the occurrence of actions, rather than the actions themselves (as usual in structured operational semantics), and provides a basis for comparison with existing quantitative extensions of interleaving models. The operational rules obtained for the real-time case are a novel (and minimal) extension to the untimed case; for the urgent case the rules strongly resemble a proposal of Bolognesi, Lucidi and Trigila; for the stochastic exponential case the rules resemble that of several existing stochastic process algebras, and for the probabilistic case we obtain rules that are related (but simpler) to work of Hansson and Jonsson. The relationship between these operational semantics and the denotational semantics is thoroughly investigated.

The incorporation of recursion in all extensions of process algebras in this dissertation is treated in Chapter 10. Using standard domain theory the denotational semantics of the quantitative extensions of PA is extended in order to cover recursively defined processes. The same is done for the event-based operational semantics. It is shown that the consistency results for the finite case carry over to the recursive case.

Chapter 11 contains a retrospective view on the work presented in this dissertation, summarizes the main technical results and provides some overall conclusions.

Contents

1 Introduction

"Abandonment of causality as a matter of principle
should be permitted only in the most extreme emergency"
ALBERT EINSTEIN, 1924[1]

This chapter highlights the main topics of this dissertation and sketches its context. The chapter briefly introduces the aspects of using formal models for concurrency in the design of distributed systems, and motivates the need for integrated formal and quantitative methods to effectively support this design process. The importance of the notion of causality for distributed systems' design is described. A synopsis is given of the contents of this dissertation.

1.1 Introduction

Concurrency is a phenomenon that plays a prominent rôle in systems of different nature. In fact, only a minority of the systems in real-life is purely sequential. The functionality in communication systems such as the mobile telecommunication system GSM (Global System for Mobile telecommunication) is distributed over several geographically separated subsystems, each having its own functionality, and a VLSI chip comprises several components connected via a network of on-chip wires.

Sequential computer systems have been extensively studied and a number of well-established mathematical models that describe the behaviour of such systems have been developed. These models usually describe a relation between input and output values and characterize behaviours as computations that evolve from an initial to a final state. For systems whose functionality is distributed over subsystems interactions do not conform to this simple scheme—usually inputs to the system depend on previous system outputs—and typically such systems are required not to terminate. Systems whose behaviour is characterized by their interaction with the environment are often referred to as *reactive systems*.

During the last decade several models for concurrent systems have been (and still are being) investigated. We confine ourselves to *formal* models for concurrency. Formal models for concurrency have a mathematically sound basis which is used to specify and reason about concurrent systems. Their main aim should be to effectively support the system *design* process, where the design process consists of a sequence of specification and transformation phases. For an overview of formal models for concurrency we refer to Winskel & Nielsen [156].

[1] A. Pais - 'Subtle is the Lord....' – The science and the life of Albert Einstein. Oxford University Press, 1983.

The correct design of concurrent systems is known to be a complex task and is usually carried out in a step-wise fashion, starting from a set of user requirements evolving towards a concrete instance of the system via a sequence of design steps. Formal models can support this design process in several ways. For instance, they allow to provide unambiguous specifications of designs (within the constraints of the model at hand) and due to their mathematical basis they enable to verify properties like absence of deadlocks and livelocks. In addition, based on a formal notion of whether a design conforms to its specification the formal specification can be used as a blueprint to generate correct tests for assessing this conformance relation. Finally, we mention that formal models provide a basis for design transformations that given a specification S generate a specification S' by incorporating some design decisions, while guaranteeing the correctness of this process (in terms of some formally defined relation). Altogether this renders important benefits for reaching correctness during the design process.

1.2 Interleaving versus noninterleaving models

A main distinction between formal models for concurrency is that of *interleaving* versus *noninterleaving* models. In interleaving models one abstracts from the fact that a system is actually composed of a set of (partly) independent subsystems. They consider the global system state without regarding its distributed nature. The system's behaviour is modelled in terms of sequences of actions that are totally ordered by precedence. Actions of one independent subsystem are merged, or interleaved, with actions of others. Interleaving models allow for the transformation of the parallel composition of finite subsystems into an equivalent specification in which parallel composition (denoted $|||$) is replaced by alternative composition (denoted $+$) and sequencing (denoted ;), e.g.,

$$a \,|||\, b = a \,;\, b + b \,;\, a$$

This transformation—in its complete form known as the *expansion theorem*—eases the verification process. A main shortcoming of interleaving models is that they do abstract from the distribution and independence of subsystems and their actions. Well-known examples of interleaving models for concurrency are (labelled) transition systems of Keller [84], synchronization trees of Milner [103] and traces of Hoare [74].

Noninterleaving models do not abstract from the fact that a system consists of a set of (partly) independent subsystems. The notion of global state does not play a central rôle in these models. The systems' behaviour is modelled in terms of sequences of actions that are not required to be totally ordered, but that are only partially ordered. This partial order reflects the causal dependencies between actions. Noninterleaving models are therefore also referred to as *partial-order* or *causality-based* models.[2] Prominent examples of noninterleaving models for concurrency are Petri nets, Reisig [125], event structures, Nielsen *et al.* [114], Mazurkiewicz traces [101], asynchronous transition systems, Shields [137] and pomsets (partially ordered multisets), Pratt [121].

[2] Terminology in the literature is not always clear; e.g., there are models for concurrency that are neither interleaving nor causality based, such as ST-bisimulation of Van Glabbeek and Vaandrager [54].

There is sometimes a strong debate between advocates of interleaving and noninterleaving models about the question 'which model is better?'. It is doubtful whether this is the right question to be answered. There are a lot of cases in which interleaving models impose the right amount of abstraction, and the same applies to noninterleaving ones. When we want to reason about, for instance, the observational behaviour of a system it is usually not so relevant to take into account the fact that a system is composed of subsystems, but it suffices to consider a system as a *black box* while ignoring this composition aspect. This applies, for instance, to the field of conformance testing where usually (and often deliberately) no knowledge is available about the internal structure of a system. Also in the realization phase of the design trajectory when (part of) a specification has to be realized on a single processor, interleaving models suffice. Finally, for verification purposes it has been proven by numerous case studies that interleaving models are appropriate to prove important and interesting properties of distributed systems.

Interleaving models are not that appropriate for design stages in which the distribution aspects of the system play a prominent rôle. The global state assumption of interleaving models hampers to faithfully model that a system consists of several co-operating subsystems at different locations, each having its own local state. In these design stages the system is considered as a *white box* where the internal system structure prevails. In particular, if the specification serves as a prescription for the system's implementation rather than as a description of the observational behaviour of a system, interleaving models become unattractive or even misleading since the independence of actions is not reflected properly, see Vissers [147]. Also for an important design technique, known as *action refinement*, where an abstract action is implemented by a number of more concrete actions, it appears that noninterleaving models are more appropriate.

We, therefore, believe that both models are legitimate and *complementary* in the design process. Going from one design stage to another may therefore imply a transition from one model for concurrency to another, and this might involve a change from an interleaving to a causality-based model or vice versa. Of course, such transitions should be carried out in a *consistent* way: there must be a strong (and formal) correspondence between the two models.

This dissertation deals with event structures, a prominent branch of noninterleaving models. Although we mainly deal with noninterleaving models, we will provide the ingredients to obtain consistent interleaving models such that both models can be used in a coherent way.

1.3 Integration of formal and quantitative methods

Originally, formal models concentrated on the specification, design and analysis of functional aspects of distributed systems. This is not at all surprising as traditionally the design process is carried out focusing entirely on the functional aspects without due regard of performance issues. During the design trajectory quantitative modelling is often disregarded, and only in the implementation (or realization) phase—or even worse, after finishing this phase—performance aspects come into focus. As pointed out in Harvey [66] it is not unusual that a system is completely designed and tested for its conformance with respect to the functional specification before any attempt is made to assess its performance characteristics. In case the finally obtained design has unsatisfactory efficiency characteristics such an *a posteriori* performance assessment may lead to a complete re-design or to the operation of the system with degraded efficiency. From several perspectives this is not desirable.

Performance should therefore be considered as one of a number of design constraints and one should aim at a close integration of performance modelling in the design process. In this way, even in the early phases of the design trajectory the efficiency of design alternatives can be assessed such that designs can be rejected because of unsatisfactory performance characteristics, thus avoiding costly re-design at later phases. Obviously, such design decisions are only of value if the performance information is adequate and reliable. Since at each phase of the design a system specification is available it seems beneficial to consider this specification not only as a basis for the functional design, but also as the starting-point for carrying out a performance assessment.

In order not to burden the design engineer with details of performance modelling and analysis it would be optimal if system specifications can be enhanced with quantitative information in an easy and conservative way. This embodies that the specification language should have a high level of 'ease of expressiveness', that it allows for the addition of quantitative information only in parts of the specification where it is really necessary, and that functional specifications can be annotated with quantitative information in such a way that when deleting this information the original functional specification is obtained (while preserving its semantics).

Performance models are typically developed by experienced performance engineers. Usually performance models are developed while intuitively simplifying the system specification that is used for the qualitative analysis and functional design of the system. Even in cases when the system specification is used as a basis, the process of going from this specification to a performance model is based on human ingenuity and is carried out manually. As a result the link between the performance model and the system specification that is used for the design is weak—there is no guarantee for the correspondence between the two—and the adequacy and reliability of the obtained results from the performance model may be limited. The validity of the performance model could be increased significantly when performance models are derived from (formal) system specifications in an algorithmic way.

We believe that the integration of formal and quantitative methods is needed. Starting from a formal model facilitates tool support—which is indispensable to support the design process and performance engineering—and allows for a provably correct mapping of specifications onto performance models. The first step towards such an integration is the extension of formal models for concurrency with quantitative information such as time (both deterministic and stochastic) and probability, which is the main topic of this dissertation. Timing information can be used to constrain the time of occurrence of actions while probabilities can be used to quantify the likelihood of happening of actions.

Quantitative extensions of interleaving models have been investigated thoroughly in the last 5–10 years. Although there does not yet seem to be a consensus on how to incorporate issues like time and probability in labelled transition systems—the most prominent interleaving model—the different ways in which this can be done seem to be quite well-understood. Various recipes on how to incorporate time in transition systems, for instance, are described by Nicollin &

Sifakis [112] and Alur & Dill [5], while different approaches for the incorporation of probabilities are described by Van Glabbeek *et al.* [53].

The incorporation of quantitative information in noninterleaving models has received scant attention in the literature. Since these models seem to be attractive at the design stages in which the observational behaviour is no longer prevalent, but where the intensional system characteristics dominate, one might even argue that such models in particular should deal with issues like time and probability. In these design stages it is of utmost importance how actions are scheduled in time and with what probability certain alternative executions, which at a more high level of abstraction could be faithfully modelled by means of nondeterminism, can appear.

In addition, if one aims at the integration of formal and quantitative methods for the support of the system design process there are several reasons why it seems to be beneficial to start from a noninterleaving model. Noninterleaving models retain explicit information about the parallelism between system components. As performance models typically are based on abstractions of the control and/or data flow structure of the systems, the use of causality-based models is thought to be a direct way of narrowing the gap with functional models. Additional advantages of these models are that they are less affected by the problem of 'state explosion', since parallelism leads to a sum of the components states, rather than to their product (as in interleaving), and that they have the possibility of *local analysis*. This means that it is relatively easy to study only that part of a system in which one is interested, isolating it from the rest.

In this dissertation we investigate several quantitative extensions of event structures. Although it has been argued, for instance by Baeten [6], that the incorporation of features like time and probability is "more difficult to achieve in the full generality of partial order semantics" and "is so much more complex in partial order semantics, that the key issues and main difficulties do not stand out so easily" we believe that most of the quantitative extensions discussed in this dissertation prove the opposite. Also the consistent interleaving models for these extensions often turn out to be simpler than various extensions of interleaved models that have been proposed in the literature. One might pose that starting from a model that explicitly reflects the causal dependencies between actions provides another, and often clarifying, insight into the intertwining of notions like time, probability, causality and independence.

1.4 Process algebra

Although formal models for concurrency aim (amongst others) at facilitating unambiguous specifications of designs, they are not attractive as such for this purpose, but they are usually used as semantical models for more abstract description languages. A prominent branch of such description languages is formed by the family of *process algebras*, like ACP of Bergstra & Klop [13], CSP of Hoare [74] and Milner's CCS [104].

Process algebras are characterized by a high level of abstraction and the presence of a number of powerful composition operators that facilitate the development of well-structured specifications. It has been widely recognized that due to these characteristics process algebras

syntactic construct	*syntax*	*label set* $\mathsf{Act}(B)$
inaction	$\mathbf{0}$	$\varnothing$
successful termination	$\surd$	$\varnothing$
action-prefix	$(a\,;\,B)$, $a \in \mathsf{Act}$	$\{\,a\,\} \cup \mathsf{Act}(B)$
	$(\tau\,;\,B)$	$\mathsf{Act}(B)$
choice	$(B_1 + B_2)$	$\mathsf{Act}(B_1) \cup \mathsf{Act}(B_2)$
enabling	$(B_1 >> B_2)$	$\mathsf{Act}(B_1) \cup \mathsf{Act}(B_2)$
disrupt	$(B_1\,[>\,B_2)$	$\mathsf{Act}(B_1) \cup \mathsf{Act}(B_2)$
parallel composition	$(B_1\,\|\|_G\,B_2)$	$\mathsf{Act}(B_1) \cup \mathsf{Act}(B_2)$
hiding	$(B \setminus G)$	$\mathsf{Act}(B) \setminus G$
relabelling	$(B[H])$	$\{\,H(a) \mid a \in \mathsf{Act}(B)\,\}$
process instantiation	P	$\mathsf{Act}(B)$ for $P := B$

Table 1.1: The syntax of process algebra PA.

are appropriate for the effective support of the design process (see, for instance, Bolognesi *et al.* [21]) and the specification of real-life systems such as communication protocols, see e.g. Sharp [136]. Therefore, in this dissertation we will investigate for each quantitative extension of event structures whether such a model can be used to provide a denotational semantics to a quantitative extension of a process algebra, referred to as PA, in a compositional way.

According to the *compositionality* principle the interpretation of each composite behaviour expression in the process algebra is defined as a function of the interpretation of its constituents. Another important characteristic that is considered in this dissertation is called *backwards compatibility* [147]. This principle embodies that the semantic function for, let say a timed behaviour B, should not modify the semantics of the untimed behaviour B' obtained by omitting all timing information in B, but rather should preserve the semantics of B'. Stated otherwise, the semantics of e.g. a timed behaviour should be a *conservative extension* of the semantics of its corresponding untimed behaviour.

In this dissertation we consider the process algebra PA which is, in fact, the process algebra LOTOS (for an introduction to LOTOS see, for instance, Bolognesi & Brinksma [16] and Logrippo *et al.* [94]) with a somewhat more concise syntax. The syntax of PA is listed in Table 1.1. The table assumes a given set of observable actions Act and an additional *silent* or *internal action* τ; $\tau \notin \mathsf{Act}$. The special action δ, which is not user-definable, indicates the *successful termination* of a behaviour; $\delta \notin \mathsf{Act}$. $\mathsf{Act}(B)$ for behaviour B is the set of observable actions in B, i.e., $\mathsf{Act}(B) \subseteq \mathsf{Act}$. $G \subseteq \mathsf{Act}$ is a set of observable actions, and $H : \mathsf{Act} \cup \{\,\tau, \delta\,\} \longrightarrow \mathsf{Act} \cup \{\,\tau, \delta\,\}$ a relabelling function that satisfies $H(\tau) = \tau$, $H(\delta) = \delta$ and for $a \in \mathsf{Act} : H(a) \neq \tau$ and $H(a) \neq \delta$. PN is a set of process names with $P \in \mathsf{PN}$. For set of actions $G \subseteq \mathsf{Act}$ we often abbreviate $G \cup \{\,\tau\,\}$ by G^τ, and similarly for δ.

As syntactical sugar we let $||_\varnothing$ be denoted by $|||$, and $||_{\mathsf{Act}}$ by $||$. The precedences of the composition operators are, in decreasing binding order: ; , +, $||$, [>, >>, \ and []. Parentheses are omitted if this does not introduce ambiguities.

The simplest behaviour is the behaviour that can perform no actions at all, called *inaction* (or deadlock) and denoted by $\mathbf{0}$. $\surd$ represents the *successful termination* of a behaviour and can perform an action δ after which it behaves like $\mathbf{0}$.

For a an action and B a behaviour, $a\,;B$ denotes a behaviour which may engage in a after which it behaves like B. This operator is called *action-prefix.*

$B_1 + B_2$ denotes the *choice* between behaviours B_1 and B_2. It should be noted that this choice is resolved in interaction with the environment, that is, by a behaviour that is composed in parallel with $B_1 + B_2$.

$B_1 >> B_2$ denotes the *sequential composition* (or *enabling*) of behaviours B_1 and B_2. Initially this behaviour behaves like B_1 but at the successful termination of B_1 control is passed to the second behaviour B_2.

The intuitive interpretation of $B_1 [> B_2$ (pronounce *disrupt*) is that B_1 at any point of its execution may be disrupted by B_2, where the successful termination of B_1 leads to the successful termination of the entire behaviour $B_1 [> B_2$.

Parallel composition of behaviours is denoted by $B_1 \,||_G\, B_2$, where G is the set of actions which have to be performed by both behaviours in co-operation. B_1 and B_2 can perform actions that are not part of the (synchronization) set G independently of each other. Successful termination actions have to be commonly executed; this means that $B_1 \,||_G\, B_2$ terminates if and only if both components terminate.

Abstraction of a set of actions G in a behaviour B is supported by the *hiding* operator, denoted $B \backslash G$. Behaviour $B \backslash G$ behaves analogous to B except that actions in the set G are turned into silent actions (denoted by τ) such that those actions are no longer visible to the environment of the behaviour.

$B[H]$ (called *relabelling*) denotes a behaviour which is obtained by renaming the actions in B according to H. Notice that silent actions τ are not renamed.

P denotes a *process instantiation*; we assume a behaviour is always considered in the context of a set of process definitions of the form $P := B$ where B is a behaviour (possibly containing occurrences of P) .

1.5 Standard semantics and behavioural equivalences

The formal semantics of PA is given by a set of SOS (*Structured Operational Semantics*, Plotkin [120]) rules that define transitions of the form $\xrightarrow{a}$. $B \xrightarrow{a} B'$ denotes that behaviour B can perform action $a \in \mathsf{Act}^{\tau,\delta}$ evolving into B'. In the SOS-style the transition relation is defined by means of deduction rules. For every syntactical construct in PA rules will be presented that define the transitions that are possible for a behaviour of this form by referring to the possible transitions of the components of this behaviour. The general format for these rules is as follows:

This general rule should be read as follows: if condition is satisfied, the rule can be applied and it can be derived that the conclusion holds in case all preconditions premise$_1 \ldots$ premise$_n$

$$\frac{\text{premise}_1 \wedge \ldots \wedge \text{premise}_n}{\text{conclusion}} \quad (\text{condition})$$

are satisfied.

The transition relation $\xrightarrow{a}$ is defined as the smallest relation closed under all inference rules of Table 1.2.

Usually transition systems are too concrete in the sense that they distinguish behaviours which—from a particular perspective—are considered to represent the same thing. We recall five notions of equivalence from the literature that are used in this dissertation, viz. isomorphism, strong bisimulation of Milner [103] and Park [116], weak bisimulation of Milner [104], testing equivalence by De Nicola & Hennessy [111], and trace equivalence by Hoare [74]. For an overview and comparison of the different types of equivalence relations on labelled transition systems we refer to the studies of Van Glabbeek [49, 50]. The order of presentation of equivalence relations in this section is by decreased distinguishing power.

1.1. DEFINITION. (*Labelled transition system*)

A *labelled transition system* is a quadruple $\langle S, L, T, s_0 \rangle$ with

- S, a set of *states*
- L, a set of *labels*
- $T \subseteq S \times L \times S$, a *transition relation*, and
- $s_0 \in S$, the *initial state.*

□

$(s, a, s') \in T$ is usually denoted as $s \xrightarrow{a} s'$. The class of labelled transition systems is denoted by LTS and is ranged over by *TS*. In the remainder of this section we will identify a labelled transition system with its initial state. We recall the following (standard) notations. Let $a_i \in \mathsf{Act}^{\tau,\delta}$, $b_i \in \mathsf{Act}^{\delta}$, σ a finite sequence of actions $a_1 \ldots a_n$, and σ' a finite sequence of observable actions $b_1 \ldots b_n$.

$$\begin{array}{lcl}
s \xrightarrow{\sigma} s' & \triangleq & \exists\, s_1, \ldots, s_{n-1} : s \xrightarrow{a_1} s_1 \xrightarrow{a_2} \ldots \xrightarrow{a_{n-1}} s_{n-1} \xrightarrow{a_n} s' \\
s \overset{\varepsilon}{\Longrightarrow} s' & \triangleq & \exists\, n \geqslant 0 : s \xrightarrow{\tau^n} s' \\
s \overset{b}{\Longrightarrow} s' & \triangleq & \exists\, s_1, s_2 : s \overset{\varepsilon}{\Longrightarrow} s_1 \xrightarrow{b} s_2 \overset{\varepsilon}{\Longrightarrow} s' \\
s \overset{\sigma'}{\Longrightarrow} s' & \triangleq & \exists\, s_1, \ldots, s_{n-1} : s \overset{b_1}{\Longrightarrow} s_1 \overset{b_2}{\Longrightarrow} \ldots \overset{b_{n-1}}{\Longrightarrow} s_{n-1} \overset{b_n}{\Longrightarrow} s' \quad .
\end{array}$$

The $\Longrightarrow$ transition relation concentrates on *observable* actions. $s \overset{\varepsilon}{\Longrightarrow} s'$ denotes that s can evolve into s' in an unobservable way, either by executing a number of $\xrightarrow{\tau}$ steps or by performing no step at all (n=0). $s \overset{b}{\Longrightarrow} s'$ denotes that s may evolve into s' by performing observable action b, possibly preceded and/or followed by any finite number of $\xrightarrow{\tau}$ steps. $\xrightarrow{\sigma}$ and $\overset{\sigma'}{\Longrightarrow}$ are the generalizations for sequences of actions of $\xrightarrow{a}$ and $\overset{b}{\Longrightarrow}$, respectively.

$$\frac{}{\sqrt{} \xrightarrow{\delta} \mathbf{0}} \qquad \frac{}{a\,;\,B \xrightarrow{a} B}$$

$$\frac{B_1 \xrightarrow{a} B_1'}{B_1 + B_2 \xrightarrow{a} B_1'} \qquad \frac{B_2 \xrightarrow{a} B_2'}{B_1 + B_2 \xrightarrow{a} B_2'}$$

$$\frac{B_1 \xrightarrow{a} B_1'}{B_1 >> B_2 \xrightarrow{a} B_1' >> B_2} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{\delta} B_1'}{B_1 >> B_2 \xrightarrow{\tau} B_2}$$

$$\frac{B_1 \xrightarrow{a} B_1'}{B_1 \,[> B_2 \xrightarrow{a} B_1' \,[> B_2} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{\delta} B_1'}{B_1 \,[> B_2 \xrightarrow{\delta} B_1'}$$

$$\frac{B_2 \xrightarrow{a} B_2'}{B_1 \,[> B_2 \xrightarrow{a} B_2'}$$

$$\frac{B_1 \xrightarrow{a} B_1'}{B_1 \,||_G\, B_2 \xrightarrow{a} B_1' \,||_G\, B_2} \quad (a \notin G^\delta) \qquad \frac{B_2 \xrightarrow{a} B_2'}{B_1 \,||_G\, B_2 \xrightarrow{a} B_1 \,||_G\, B_2'} \quad (a \notin G^\delta)$$

$$\frac{B_1 \xrightarrow{a} B_1' \;\wedge\; B_2 \xrightarrow{a} B_2'}{B_1 \,||_G\, B_2 \xrightarrow{a} B_1' \,||_G\, B_2'} \quad (a \in G^\delta)$$

$$\frac{B \xrightarrow{a} B'}{B \setminus G \xrightarrow{a} B' \setminus G} \quad (a \notin G) \qquad \frac{B \xrightarrow{a} B'}{B \setminus G \xrightarrow{\tau} B' \setminus G} \quad (a \in G)$$

$$\frac{B \xrightarrow{a} B'}{B[H] \xrightarrow{H(a)} B'[H]} \qquad \frac{B \xrightarrow{a} B'}{P \xrightarrow{a} B'} \quad (P := B)$$

Table 1.2: Operational semantics of PA.

1.2. DEFINITION. For $TS \in \mathsf{LTS}$ let $der(TS) \triangleq \{ TS' \mid \exists\, \sigma \in (\mathsf{Act}^{\delta})^* : TS \stackrel{\sigma}{\Longrightarrow} TS' \}$. □

Two labelled transition systems are isomorphic if their reachable states can be mapped one-to-one to each other, preserving transitions and initial states.

1.3. DEFINITION. (*Isomorphism*)

For i=1, 2 let $TS_i = \langle S_i, L, T_i, s_{0_i} \rangle$. TS_1 and TS_2 are called *isomorphic*, denoted $TS_1 =_{\mathrm{iso}} TS_2$, iff there exists a bijection $\phi : der(TS_1) \longrightarrow der(TS_2)$ such that $\phi(s_{0_1}) = s_{0_2}$ and $s \xrightarrow{a} s'$ iff $\phi(s) \xrightarrow{a} \phi(s')$, for all $s, s' \in der(TS_1)$ and $a \in \mathsf{Act}^{\tau,\delta}$. □

Strong bisimulation equivalence requires the existence of a relation between the reachable states of two transition systems that can simulate each other: if one can perform action $a \in \mathsf{Act}^{\tau,\delta}$, the other must be able to do the same, and vice versa, and the resulting states must simulate each other again.

1.4. DEFINITION. (*Strong bisimulation equivalence*)

For i=1, 2 let $TS_i = \langle S_i, L, T_i, s_{0_i} \rangle$. TS_1 and TS_2 are called *strong bisimulation equivalent* , denoted $TS_1 \sim TS_2$, iff there exists a relation $\mathcal{R} \subseteq der(TS_1) \times der(TS_2)$ such that $(s_{0_1}, s_{0_2}) \in \mathcal{R}$ and if $(s_1, s_2) \in \mathcal{R}$ then for all $a \in \mathsf{Act}^{\tau,\delta}$

- $\forall\, s_1' \in S_1 : s_1 \xrightarrow{a}_1 s_1'$ implies $\exists\, s_2' \in S_2 : s_2 \xrightarrow{a}_2 s_2' \;\wedge\; (s_1', s_2') \in \mathcal{R}$;
- $\forall\, s_2' \in S_2 : s_2 \xrightarrow{a}_2 s_2'$ implies $\exists\, s_1' \in S_1 : s_1 \xrightarrow{a}_1 s_1' \;\wedge\; (s_1', s_2') \in \mathcal{R}$.

□

Weak bisimulation is defined similarly, but focuses on observable transitions.

1.5. DEFINITION. (*Weak bisimulation equivalence*)

For i=1, 2 let $TS_i = \langle S_i, L, T_i, s_{0_i} \rangle$. TS_1 and TS_2 are called *weak bisimulation equivalent* , denoted $TS_1 \approx TS_2$, iff there exists a relation $\mathcal{R} \subseteq der(TS_1) \times der(TS_2)$ such that $(s_{0_1}, s_{0_2}) \in \mathcal{R}$ and if $(s_1, s_2) \in \mathcal{R}$ then for all $\sigma \in (\mathsf{Act}^{\delta})^*$

- $\forall\, s_1' \in S_1 : s_1 \stackrel{\sigma}{\Longrightarrow}_1 s_1'$ implies $\exists\, s_2' \in S_2 : s_2 \stackrel{\sigma}{\Longrightarrow}_2 s_2' \;\wedge\; (s_1', s_2') \in \mathcal{R}$;
- $\forall\, s_2' \in S_2 : s_2 \stackrel{\sigma}{\Longrightarrow}_2 s_2'$ implies $\exists\, s_1' \in S_1 : s_1 \stackrel{\sigma}{\Longrightarrow}_1 s_1' \;\wedge\; (s_1', s_2') \in \mathcal{R}$.

□

The notion of testing equivalence is used to determine whether an implementation (concrete behaviour) is correct with respect to a specification (abstract behaviour). The following characterization of testing equivalence is taken from Tretmans [142].[3]

For $TS \in \mathsf{LTS}$ and trace σ the predicate TS **after** σ **deadlocks** is defined as:

$$TS \textbf{ after } \sigma \textbf{ deadlocks} \;\triangleq\; (\exists\, TS' : TS \stackrel{\sigma}{\Longrightarrow} TS' \;\wedge\; (\forall\, a \in \mathsf{Act}^{\delta} : TS' \stackrel{a}{\not\Longrightarrow})) \ .$$

[3] This characterization coincides with the definition of testing equivalence in De Nicola & Hennessy [111] for *strongly converging* labelled transition systems, that is, transition systems in which no infinite chains of internal actions appear.

That is to say, TS **after** σ **deadlocks** is true iff TS can evolve observedly via σ to TS' and TS' cannot perform any observable action.

As a second subsidiary notion let $Obs(TS_1, TS_2)$ denote the set of observable traces σ such that $TS_1 \,||\, TS_2$ deadlocks after performing σ.

$$Obs(TS_1, TS_2) \triangleq \{\, \sigma \in (\mathsf{Act}^\delta)^* \mid (TS_1 \,||\, TS_2) \textbf{ after } \sigma \textbf{ deadlocks} \,\}.$$

TS_1 and TS_2 are called testing equivalent iff there is no transition system (often called test) which can distinguish between TS_1 and TS_2.

1.6. Definition. (*Testing equivalence*)
TS_1 and TS_2 are called *testing equivalent*, denoted $TS_1 \approx_{te} TS_2$, iff

$$\forall\, TS \in \mathsf{LTS} : Obs(TS_1, TS) = Obs(TS_2, TS) \quad .$$

□

Let us define the set of sequences consisting of observable actions of TS. That is,

$$Traces(TS) \triangleq \{\, \sigma \in (\mathsf{Act}^\delta)^* \mid \exists\, s \in S : s_0 \overset{\sigma}{\Longrightarrow} s \,\} \quad .$$

Two transition systems are called trace equivalent if they have the same set of traces.

1.7. Definition. (*Trace equivalence*)
TS_1 and TS_2 are called *trace equivalent* iff $Traces(TS_1) = Traces(TS_2)$. □

The equivalence relations defined above for labelled transition systems will be used for behaviours in the same way.

1.6 The principles of event structures

Event structures constitute a major branch of noninterleaving models. The basic ingredients of event structures are labelled events, and the causality, conflict, and independence relation between events. Since the conception of event structures in Winskel's thesis [152] various types of event structures have been developed. Many of these models are introduced in Chapter 2. This section treats the elementary concepts of event structures.

The basic building blocks of behaviours are *actions*. An action models an activity, like consuming a sandwich, preparing a dinner, or pressing a button on a keyboard. Actions are atomic in the sense that they are indivisible. This implies that an action either takes place, or does not take place at all. It cannot take place partly, given the abstraction level at hand. At a lower abstraction level, however, an action may be refined into more detailed actions which at that level of abstraction are again considered to be atomic. For instance, preparing a dinner may be considered as a single action at some abstraction level, but at a more detailed level, it may consist of several (sub-)activities such as cleaning the ingredients, preparing the first course, preparing the second course, and so on. Actions are represented in event structures by *labels*. We assume the existence of some universe of actions, denoted $\mathcal{A}$, and indicate elements of this universe by $a, b, c, \ldots$.

The building blocks of event structures are *events*. An event models the occurrence of an action. For each occurrence of an action the time at which it occurs, the reasons for its occurrence, and the context in which it happens are different. An event is a specific occurrence of an action. For instance, preparing dinner at Christmas 1995 or on June 3th 1987 could be modelled as two distinct events of the action preparing a dinner (at any day). The relation between events and actions is provided by a *labelling function* that associates to an event the action whose occurrence is modelled by this event. Since different events may model distinct occurrences of the same action, and as there may exist actions to which no event corresponds, this labelling function is, in general, neither injective nor surjective.

Events are denoted in pictures as black dots; near the dot the action label is given. We usually denote an event labelled a by e_a. In case the event's label is irrelevant it is omitted and we simply write e, e', and so on. Event names are taken from some (arbitrary) domain such that events can be identified uniquely. We are actually not interested in explicitly defining event names and consider event structures up to event renaming.

Causality (or precedence) is a binary relation between events where the intuitive interpretation of e causes e', denoted by a directed arrow from e to e', is that if e and e' both occur then e' is caused by e. Stated otherwise, the occurrence of e is a condition for e' to be able to occur. It does not need to be a sufficient condition for e' to happen, because there may be other events on which e' causally depends, or there may be other events which may disable the occurrence of e' (see below). Causality is based on the intuition that there is a fixed cause-and-effect relation between occurrences of actions (i.e., events) in system runs. Causality is described at the level of events rather than at the level of actions since in general different action occurrences have different causes.

Conflict (or choice) is a symmetric binary relation between events, represented by a dotted line between e and e', with the intended meaning that e and e' will never both happen in a possible run of the system. Thus, if e (e') happens in a system run then e' (e) is permanently disabled.

Independence is a symmetric binary relation between events with the intended meaning that if e and e' are neither causally related nor in conflict, then they can happen independently of each other. That is, once enabled they can happen in any order or even simultaneously. The independence of two events is indicated by the absence of a causal relation and conflict relation between these events.

The representation of the basic ingredients of event structures is presented in Figure 1.1.

There are various types of event structures defined in the literature (see Chapter 2). The specific requirements of parallel composition with multi-way synchronization and the disrupt operator of our process algebra PA are appropriately addressed by Langerak's *extended bundle event structures* [89, 90]. In a nutshell, this type of event structures incorporates besides labelled events, an asymmetric conflict relation, denoted $\rightsquigarrow$, and a causality relation between

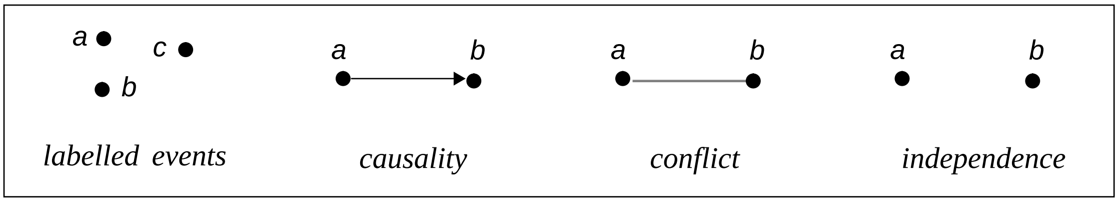


Figure 1.1: Basic ingredients of event structures.

a set X of events, that are pairwise in mutual conflict, and an event e. The intuitive meaning of $e \rightsquigarrow e'$ is that (i) e cannot occur once e' has occurred, and (ii) if e and e' both occur in a single system run then e causally precedes e'. The interpretation of $X \mapsto e$ is that if e happens in a system run, exactly one event in X has happened before (and caused e). This enables us to uniquely define a causal ordering between the events in a system run.

In this dissertation we take extended bundle event structures as a starting-point for our investigations on quantitative extensions of noninterleaving models.

Besides the use of event structures as a semantical model for process algebras we like to mention the increase of interest in causality-based models in other areas like, for instance, the automatic verification of temporal logic properties (known as model checking) [55], the design and verification of distributed algorithms [33, 134, 77], the modelling of advanced architectural concepts [46, 145], and the design and analysis of parallel computations [12].

1.7 Families of lposets

The interpretation of event structures is traditionally defined in terms of *families of configurations* as in Winskel & Nielsen [156]. A configuration is a representation of the system state by means of the set of events that have occurred up to a certain point. For extended bundle event structures it turns out that families of configurations are not sufficiently expressive. That is to say, there are extended bundle event structures that have identical families of configurations, but that are different from a causality point of view. We therefore take a more discriminating model, known as *labelled partially ordered sets*, or lposets, for short. Families of lposets do not only record the set of events that have happened so far, but also the causal ordering between the events. Rensink [126, 127] showed that lposets form a convenient underlying model for many formal models for concurrency. Let $\mathcal{A}$ denote a set of actions.

1.8. DEFINITION. (*Labelled partially ordered set*)

A labelled partially ordered set (*lposet*) is a triple $\langle E, \leqslant, l \rangle$ with

- E, a set of events
- $\leqslant \subseteq E \times E$, a partial order on E
- $l : E \longrightarrow \mathcal{A}$, the action labelling function.

$\square$

A relation is a partial order iff it is reflexive, antisymmetric and transitive.

For denoting lposets we use the following conventions. ε denotes $\langle\varnothing,\varnothing,\varnothing\rangle$, the empty lposet. Non-empty lposets are often graphically denoted: e.g., $\langle\{\, e_a, e_b \,\}, \leqslant, \{\, (e_a, a), (e_b, b) \,\}\rangle$ is denoted by $\boxed{\begin{matrix} e_a \\ e_b \end{matrix}}$ if e_a and e_b are unrelated under $\leqslant$, and by $\boxed{e_a \rightarrow e_b}$ if $e_a \leqslant e_b$. The arrow symbol $\rightarrow$ can be read as 'causes'.

An important relation on lposets is the *prefix* relation.

1.9. DEFINITION. (*Prefix of an lposet*)
$\langle E, \leqslant, l\rangle$ is a *prefix* of $\langle E', \leqslant', l'\rangle$ iff $E \subseteq E'$, $\leqslant = \leqslant' \cap (E' \times E)$ and $l = l' \restriction E$. $\square$

The second constraint says that no event in $E' \setminus E$ may precede under $\leqslant'$ an event in E. Evidently, the relation 'is a prefix of' is a partial order on lposets.

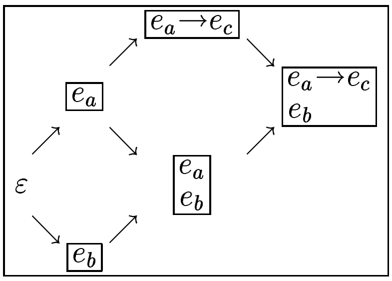
1.10. DEFINITION. (*Family of lposets*)
A family $\mathcal{P}$ of lposets is a non-empty set of finite lposets such that

$$\forall\, p \text{ an lposet}, q \in \mathcal{P} : p \text{ is a prefix of } q \;\Rightarrow\; p \in \mathcal{P} \quad .$$

$\square$

That is to say, a family of lposets is a non-empty set of (finite) lposets that is downwards closed with respect to the prefix ordering on lposets.

1.11. EXAMPLE. Consider the family of lposets graphically denoted as:



The arrows between the different lposets denote the prefix relation, omitting the transitive closure for convenience. $\square$

Lposets are a very discriminating model—lposets that only differ in their event names are considered to be different. Less discriminating semantical models such as pomset (partially ordered multiset), multiset, and interleaving models can be obtained from an lposet semantics by using the appropriate abstraction mechanism. Pomsets, for example, are equivalence classes of lposets under isomorphism.

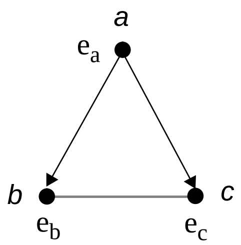
1.12. DEFINITION. $\langle E, \leqslant, l\rangle$ and $\langle E', \leqslant', l'\rangle$ are *isomorphic* iff there exists a bijection $\phi : E \longrightarrow E'$ such that $l(e) = l'(\phi(e))$ and $e \leqslant e'$ iff $\phi(e) \leqslant' \phi(e')$. $\square$

1.13. DEFINITION. A *pomset* is an isomorphism class of lposets. $\square$

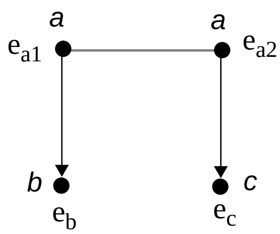
An important difference between pomsets and lposets is that pomsets are *linear-time* models, i.e., they abstract from the timing of choices, whereas lposets are *branching-time* models, i.e., they keep track of the moments of choice. (For an extensive discussion about the relevance of branching-time models we refer to Van Glabbeek [51].) In linear-time models we have

$$a\,;\,(b\,;\,\mathbf{0}+c\,;\,\mathbf{0}) = a\,;\,b\,;\,\mathbf{0}+a\,;\,c\,;\,\mathbf{0}$$

The left-hand side defines a choice between b and c after having performed an a, whereas the right-hand side the choice is made before an a is performed. The corresponding event structures of these expressions are as follows:



$a\,;\,(b\,;\,\mathbf{0}+c\,;\,\mathbf{0})$



$a\,;\,b\,;\,\mathbf{0}+a\,;\,c\,;\,\mathbf{0}$

The (maximal) lposets of the right-hand event structure are $\boxed{e_{a_1} \rightarrow e_b}$ and $\boxed{e_{a_2} \rightarrow e_c}$ whereas the (maximal) lposets of the left-hand event structure are $\boxed{e_a \rightarrow e_b}$ and $\boxed{e_a \rightarrow e_c}$. Since $\boxed{e_{a_1} \rightarrow e_b}$ and $\boxed{e_a \rightarrow e_b}$ are isomorphic, and $\boxed{e_{a_2} \rightarrow e_c}$ and $\boxed{e_a \rightarrow e_c}$ are isomorphic, we obtain the (maximal) pomsets $\boxed{a \rightarrow b}$ and $\boxed{a \rightarrow c}$ for both event structures. Lposets thus distinguish between these two event structures while pomsets do not.

1.8 Synopsis

This thesis is further organized as follows.

Chapter 2: Extended bundle event structures provides a brief survey of three traditional types of event structures: prime and stable event structures of Winskel, and the flow event structures of Boudol & Castellani. The adaptations made in Langerak's bundle and extended bundle event structures are described and justified. The latter model is extensively discussed and the major results that are of importance for this thesis are summarized. It will be shown how extended bundle event structures can be used to provide a compositional causality-based semantics to PA. In addition, a consistent event-based operational semantics of PA is presented.

Chapter 3: Disjunctive causality and interleaving presents two qualitative extensions of extended bundle event structures. In the first extension the stability constraint on bundles is dropped. The resulting model, called dual event structures, incorporates conjunctive causality—like all other event structures—and disjunctive causality—unlike most other event structures. The second extension comprehends the incorporation of an (irreflexive and symmetric) interleaving relation between events. We investigate for both models how lposets can be deduced and what transformation rules are supported. The expressiveness of the two models is compared with the event structures of Chapter 2.

Chapter 4: A simple timing module describes a simple timed variant of extended bundle event structures. We equip events and bundles with a time attribute. An event e with time t denotes that e is enabled from t time units on since the system has been started, usually assumed to be time 0. t associated with bundle $X \mapsto e$ denotes that the time between the occurrence of an event in X and the appearance of e should be at least t time units. The result is a causality-based model allowing the specification of minimal time constraints. The timing extension is a conservative extension of the untimed causality-based model, is suitable for discrete and continuous time, and does not include notions to explicitly force the passage of time. A temporal process algebra PA_T is defined that includes a delay function which constrains the occurrence time of actions. The suitability of timed event structures for providing a compositional causality-based semantics to this algebra is studied.

A preliminary version of part of this chapter has been published as [28].

Chapter 6: Timed operational semantics presents two timed event transition systems for the timed process algebra PA_T. Opposed to the standard case transitions are equipped with event and action (and time) labels. The timed event transition systems are defined by structured operational semantics. One transition model is based on timed-action transitions and the other is based on the separation between time- and (untimed) action-transitions. The compatibility of these timed transition models with the causality-based semantics of PA_T as provided in Chapter 4 is investigated. The timed event traces of the timed-action transition model and the causality-based semantical model are shown to coincide. For the model distinguishing between time- and action-transitions this holds when restricting to time-consistent traces.

Chapter 6: The urgency module introduces the concept of urgent events—events that are forced to occur once they are enabled—in timed event structures. Typically an urgent event 'guards' the occurrence time of an alternative event in the sense that this other event is prevented from happening after a particular time instant. Timeout mechanisms are well-known urgent phenomena. It is investigated how the theory of Chapter 4 carries over to this new model, referred to as urgent event structures. The timed process algebra PA_T is extended with an urgency operator that forces (local or synchronized) actions to happen in an urgent fashion. Urgent event structures are used as a vehicle to provide a denotational causality-based semantics for this formalism. In the spirit of Chapter 5 a consistent event-based operational semantics based on a separation of the passage of time and the occurrence of actions is presented.

An extended abstract of this chapter has been published as [83].

Chapter 7: The real-time module generalizes timed event structures by equipping events and bundles with sets of time instants and use urgent events for the sole purpose of modelling timeout mechanisms (thus restricting urgent event structures). An event e

with set T of time instants denotes that e can only occur at some $t \in T$ since the start of the system. T associated with bundle $X \mapsto e$ denotes that the time between the occurrence of an event in X and the appearance of e should equal t, for some $t \in T$. The result is a causality-based model allowing the specification of minimal, maximal and, for instance, periodic time constraints. This chapter generalizes the theory of Chapter 4 and uses urgent events in a controlled way. It investigates how the more expressive model, baptized real-time event structures, can be used as a vehicle to provide a semantics to a real-time process algebra including timeout and watchdog operators.

Chapter 8: The stochastic timing module treats stochastic variants of extended bundle event structures. As a result causality-based models are obtained that allow the specification of stochastic timing constraints. Events are supposed to happen after a delay that is determined by a stochastic variable with a certain distribution function. First, a simple model is discussed restricting the distribution functions to be exponential. Then the generalization of deterministic times towards more general types of distributions is investigated and a stochastic variant of event structures is proposed with (the more practical) phase-type distributions. This class of distributions includes exponential, Erlang, Coxian and mixtures of exponential distributions. It is shown how both stochastic models can be used to provide a compositional causality-based semantics to a stochastic extension of PA, and for the exponential case a corresponding event-based operational semantics is provided that is proven to coincide with various existing interleaving proposals.

This chapter has been published as [29].

Chapter 9: The probability module presents a probabilistic variant of extended bundle event structures, in which internal events (i.e., events labelled τ) can be assigned a (fixed) probability. In this way, a causality-based model is obtained that allows for the specification of (internal) probabilistic behaviour. For probabilistic event structures the notion of cluster, a set of mutually conflicting internal events such that the sum of the probabilities associated to these events is 1, is defined. A cluster corresponds to an independent stochastic experiment. A probabilistic process algebra PA_P is introduced and assigned a causality-based semantics. The integration of the probabilistic model with the simple timed model (of Chapter 4) is briefly discussed. By means of example it is shown how to obtain a performance model (i.e., a discrete-time semi-Markov chain) from a timed probabilistic event structure.

A preliminary version of part of this chapter has been published as [82].

Chapter 10: Recursion provides an event structure semantics for recursively defined processes. We consider the timed (and urgent) variant and the probabilistic variant, and show that the stochastic case can be taken into account by a straightforward generalization of the deterministic timed case. Recursion is dealt with using standard domain theory. A complete partial order is defined on each type of event structure and all operators on these structures (which correspond to operators in the related process algebra) are shown to be continuous with respect to this partial order. The semantics of $P := B$ is then defined as the limit of a series of better and better approximations.

Finally, for PA_T, PA_R, PA_U and PA_P we give an operational semantics for recursively defined processes and prove the consistency between this operational semantics and the denotational causality-based semantics.

Chapter 11: Conclusion contains a retrospective view on the work presented in this dissertation, summarizes the main technical results and provides some overall conclusions. In addition, some thoughts on future work are presented.

Appendix A: Stochastic processes provides an introduction to some basic notions of stochastic processes. Notions like distribution functions, memoryless distributions, discrete and continuous-time Markov chains are introduced and some basic results are summarized. The material of this appendix is used in Chapters 8 and 9.

Appendix B: Domain theory gives a brief introduction to standard domain theory and fixes some terminology. The material of this appendix is used in Chapter 10.

This dissertation presents 8 extensions of extended bundle event structures. These extensions and their dependencies are depicted in Figure 1.2. The numbers in brackets indicate the chapter numbers in which the corresponding model is treated. This figure thus provides also a reading guidance. For example, readers that are only interested in the stochastic extension should read Chapters 2, 4 and 8, whereas those that are interested only in the probabilistic aspects should consult Chapters 2 and 9. Chapter 10 considers recursion for all treated models.

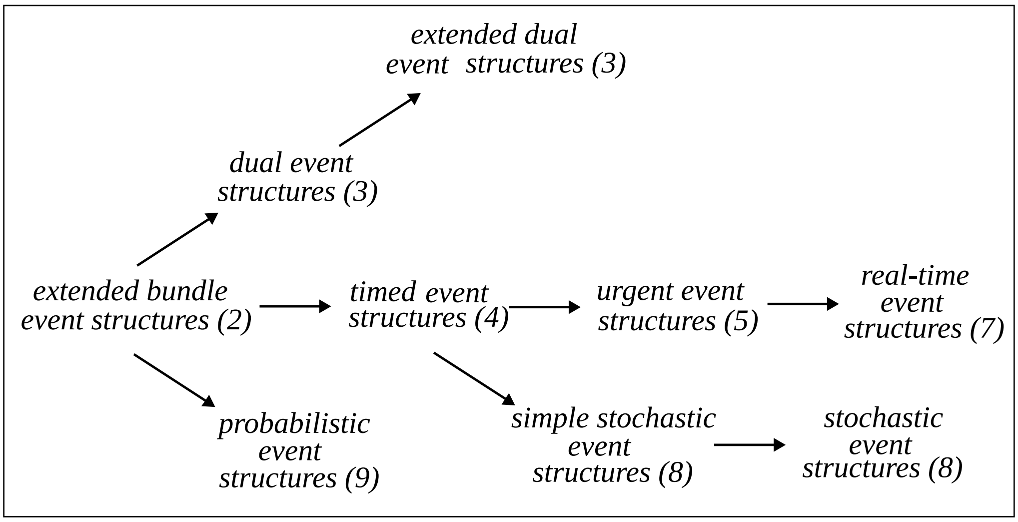


Figure 1.2: Overview of extensions of event structures.

2 Extended bundle event structures

This chapter provides a brief survey of three traditional types of event structures: prime and stable event structures of Winskel, and the flow event structures of Boudol & Castellani. The adaptations made in Langerak's bundle and extended bundle event structures are described and justified. The latter model is extensively discussed and the major results that are of importance for this thesis are summarized. It will be shown how extended bundle event structures can be used to provide a compositional causality-based semantics to PA. In addition, a consistent event-based operational semantics of PA is presented.

2.1 Introduction

For investigating qualitative and quantitative extensions of partial-order models we take Langerak's *extended bundle event structures* as a starting-point. This chapter is mainly devoted to this type of event structures. We start by briefly describing three traditional models of event structures: prime, stable and flow event structures. The descriptions of these models are not intended to give all details and internals of a certain model, but are meant to show the development and differences between the kinds of event structures.

The main difference of (extended) bundle event structures and prime and flow event structures is that the causality relation, denoted by $\mapsto$, is not a binary relation between events, but a relation between a set X of events and an event e. $X \mapsto e$ means that e is enabled if precisely one event in X has happened. As argued in [89] this relation is convenient for modelling multi-party synchronization, as present in process algebras like LOTOS and CSP, without the need for copying events.

Prime, flow, stable, and bundle event structures incorporate a symmetric conflict relation, denoted by $\#$. To model the disrupt operator $[>$ appropriately this relation is replaced by an asymmetric conflict relation (denoted by $\rightsquigarrow$) in extended bundle event structures. The intuitive meaning of $e \rightsquigarrow e'$ is that (i) e cannot occur once e' has occurred, and (ii) if e and e' both occur in a single system run then e causally precedes e'. Similar constructs have been considered in the study of architectural issues of distributed systems by Ferreira Pires *et al.* [46, 145], in the notions of event automata by Pinna & Poigné [118], and in the geometric automata of Gunawardena [60].

In the main part of this chapter we consider extended bundle event structures. We justify the use of families of lposets as an underlying semantical model and define how lposets can be generated. A pleasant property of extended bundle event structures is that 'local' transformation rules, i.e., transformation rules involving part of an event structure, can be defined. We summarize some basic transformation rules that preserve the semantics in terms of lposets. A denotational semantics of PA in terms of extended bundle event structures is defined. This semantics provides the basis for extensions of PA that are treated later on in this thesis. In addition, an operational semantics for PA is presented and shown to be consistent with the denotational semantics.

2.2 The realm of event structures

This section presents an overview of a prominent subset of event structure models as described in the literature. The presentation of these models is in chronological order of their conception, ranging from the well-known prime event structures to bundle event structures. The treatment of other types of event structures (or alikes) such as the free event structures of Darondeau & Degano [37], prioritized event structures of Degano *et al.* [43], event automata of Pinna & Poigné [118] and local event structures of Hoogers *et al.* [75, 76] is outside the scope of this chapter.

2.1. NOTATION. For X a set of events, predicate $\mathsf{CF}(X)$ is true iff X is conflict-free, that is, $\mathsf{CF}(X) \triangleq (\forall\, e, e' \in X : \neg\,(e \,\#\, e'))$.

For finite sequences $\sigma = x_1 \ldots x_n$, let $\overline{\sigma}$ denote the set of elements in σ, that is, $\overline{\sigma} \triangleq \{\, x_1, \ldots, x_n \,\}$, and let σ_i denote the prefix of σ up to the $(i{-}1)$-th element, that is, $\sigma_i \triangleq x_1 \ldots x_{i-1}$, for $0 < i \leqslant n{+}1$. □

2.2.1 Prime event structures

Originally, event structures were introduced as a vehicle to relate subclasses of Petri nets, such as occurrence nets and causal nets, and Scott's domain theory [114]. Event structures were introduced as 'net-alikes with the places removed'. To relate occurrence nets to domains the notion of *prime event structures* was introduced. Their labelled variants, associating with each event an action from a set $\mathcal{A}$ of actions, are defined as follows.

2.2. DEFINITION. (*Prime event structure*)

A *(labelled) prime event structure* $\mathcal{E}$ is a quadruple $(E, \#, \leqslant, l)$ with

- E, a set of *events*
- $\# \subseteq E \times E$, the (irreflexive and symmetric) *conflict* relation
- $\leqslant \subseteq E \times E$, a partial order, the *causality* relation
- $l : E \longrightarrow \mathcal{A}$, the *action-labelling* function

such that for all $e \in E$

1. $\{\, e' \in E \mid e' \leqslant e \,\}$ is finite

2. $\forall\, e', e'' \in E : (e \,\#\, e' \;\wedge\; e' \leqslant e'') \;\Rightarrow\; e \,\#\, e''$.

□

The first condition states that the number of causes of any event should be finite. The second condition, known as the *conflict inheritance* property, states that if an event e is in conflict with some event e', then it is in conflict with all causal successors of e'.

A configuration is a set of events that have happened during a specific run of the event structure. Conceptually a configuration C can also be viewed as a global state, namely the state of a system where all events in C have occurred.

2.3. DEFINITION. (*Configuration of a prime event structure*)

For prime event structure $\mathcal{E} = (E, \#, \leqslant, l)$, set $C \subseteq E$ is a *configuration* of $\mathcal{E}$ iff $\mathsf{CF}(C)$ holds and $\forall\, e \in C, e' \in E : e' \leqslant e \;\Rightarrow\; e' \in C$. □

A configuration should be conflict-free since conflicting events can never happen in a system run. In addition, all causal predecessors of e in C must be contained in C, i.e., C is downwards closed (w.r.t. $\leqslant$), as otherwise e could not have happened at all. The semantics of a prime event structure is defined as the family of its configurations, ordered by set inclusion. The resulting domain is a so-called prime algebraic coherent partial order [114]; this explains the name *prime* event structures.

2.4. EXAMPLE. An example of a prime event structure and its semantics in terms of families of configurations is given in Figure 2.1(a) and (b), respectively. In this structure we have $e_a \,\#\, e_b$, $e_b \leqslant e_d$, $e_c \leqslant e_d$, and $e_a \,\#\, e_d$ (due to conflict inheritance). □

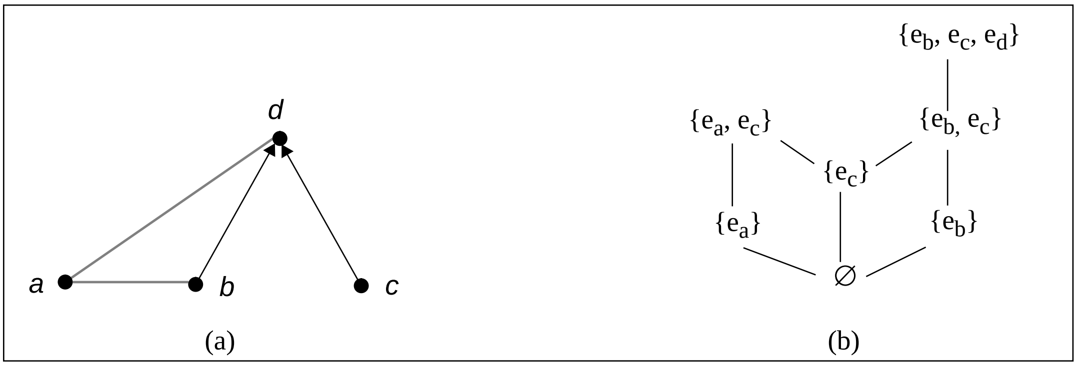


Figure 2.1: A prime event structure (a) and its family of configurations (b).

Prime event structures are simple mathematical structures: labelled partial orders extended with a conflict relation. The main limitation of prime event structures is the conflict inheritance property. Due to this property all causal successors of two conflicting events are in mutual conflict. This implies that each event can be enabled only in one way, thus disallowing an event to have alternative enablings. For describing a single system run—the original goal of occurrence nets, and thus of prime event structures—this does not bother, but for specifying behaviours this is undesirable. The fact that an event can have alternative causes can only be modelled by introducing one event for each possible enabling. This is unattractive—events usually have different alternative enablings by nature; modelling each alternative enabling of an event by a separate event would lead to an explosion of the number of events.

For similar reasons, prime event structures are unattractive as a semantical model for process algebras. Especially the semantics of the parallel composition operator is considerably complex, despite attempts to simplify it by Loogen & Goltz [95] and Vaandrager [144].

2.2.2 Stable event structures

Stable event structures were introduced by Winskel to overcome the unique enabling problem of prime event structures [153, 154]. In contrast with the binary causality relation $\leqslant$, stable event structures have an enabling relation, denoted $\vdash$, relating a (usually finite) set of events to a single event[1]. The interpretation of $X \vdash e$ for a set X of events and an event e is that e is enabled if *all* events in X have occurred.

2.5. DEFINITION. (*Stable event structure*)

A *(labelled) stable event structure* $\mathcal{E}$ is a quadruple $(E, \#, \vdash, l)$ with

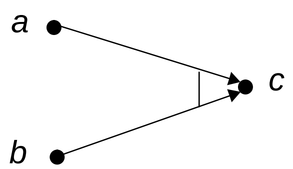
- E, a set of *events*
- $\# \subseteq E \times E$, the (irreflexive and symmetric) *conflict* relation
- $\vdash \subseteq \mathcal{P}(E) \times E$, the *enabling* relation
- $l : E \longrightarrow \mathcal{A}$, the *action-labelling* function

such that

1. $\forall X \subseteq E, e \in E : X \vdash e \;\Rightarrow\; \mathsf{CF}(X \cup \{e\})$
2. $\forall X, Y \subseteq E, e \in E : (X \vdash e \;\wedge\; Y \vdash e) \;\Rightarrow\; (\neg\,\mathsf{CF}(X \cup Y) \;\vee\; X = Y)$.

□

Enabling $X \vdash e$ is represented by drawing an arrow from each event in X to e and connecting all arrows by a small line. For instance, $X \vdash e_c$ with $X = \{e_a, e_b\}$ is depicted as



The first constraint of Definition 2.5, referred to as the *consistency constraint*, states that

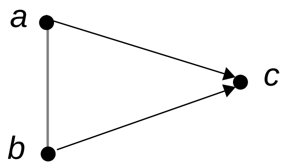
[1]For technical convenience we consider a *minimal* enabling relation. For X a set of events and event e, the minimal enabling relation $\vdash_{min}$ is defined as:

$$X \vdash_{min} e \triangleq X \vdash e \;\wedge\; (\forall Y \subseteq X : Y \vdash e \;\Rightarrow\; X = Y) \quad .$$

Since we only consider minimal enablings of stable event structures we write simply $\vdash$ rather than $\vdash_{min}$.

for enabling $X \vdash e$, all events in $X \cup \{ e \}$ should be conflict-free. The second constraint, called the *stability constraint*, ensures that an event in a system run is always enabled in a unique way. So, if there are two enablings $X \vdash e$ and $Y \vdash e$ then either they are equal $(X = Y)$ or there exists a conflict between X and Y such that they cannot both cause e. As a result, when event e occurs in a system run the events that have caused its occurrence can be unambiguously determined. This property is referred to as *absence of causal ambiguity.*

In contrast with prime event structures, events in stable event structures can have different enablings. For instance, in



event e_c can either be enabled by e_a or by e_b. This possibility leads to a more involved definition of a configuration. For that purpose the intermediate concept of *event trace* (also called proving sequence) is used. For technical convenience we define the set of events that are in conflict with some event in σ.

2.6. DEFINITION. For σ a sequence of events let $\mathsf{cfl}(\sigma) \triangleq \{ e \in E \mid \exists\, e_i \in \overline{\sigma} : e_i \# e \}$. □

2.7. DEFINITION. (*Configuration of a stable event structure*)

An *event trace* σ of stable event structure $\mathcal{E} = (E, \#, \vdash, l)$ is a sequence of events $e_1 \ldots e_n$ with $e_i \in E$ such that for all i, $0 < i \leqslant n$

$$(e_i \notin \mathsf{cfl}(\sigma_i) \cup \overline{\sigma_i}) \;\wedge\; (\exists\, X \subseteq \overline{\sigma_i} : X \vdash e_i) \quad .$$

A set $C \subseteq E$ is a *configuration* iff there is an event trace σ such that $C = \overline{\sigma}$. □

An event trace is conflict-free since all events in an event trace should be able to happen in a system run. In addition, for each event e_i in σ, at least one of the enablings X of e_i must be satisfied.

2.2.3 Flow event structures

An alternative model to overcome the unique enabling problem of prime event structures, called *flow event structures*, was developed by Boudol and Castellani [24, 26]. In flow event structures the causality relation $\leqslant$ of prime event structures is replaced by an (irreflexive) flow relation, similar to the flow relation in Petri nets which is defined by the existence of a place between two transitions. In addition, there is no requirement on the relationship between flow relations and conflicts, like the stability and consistency constraint in stable event structures, and the conflict relation is not required to be irreflexive. Whereas prime event structures correspond to occurrence nets, flow event structures correspond to a richer subclass of safe Petri nets, called flow nets [26].

2.8. DEFINITION. (*Flow event structure*)

A *(labelled) flow event structure* $\mathcal{E}$ is a quadruple $(E, \#, \prec, l)$ with

- E, a set of *events*
- $\# \subseteq E \times E$, the (symmetric) *conflict* relation
- $\prec \subseteq E \times E$, the (irreflexive) *flow* relation
- $l : E \longrightarrow \mathcal{A}$, the *action-labelling* function.

$\square$

Since $\#$ is not required to be irreflexive, *self-conflicting* events, that is, e such that $e \# e$, are allowed. Such events seem not very useful from a specifier's point of view as they will never occur, but turn out to be essential for defining operations (like parallel composition) on flow event structures. It should also be noted that the conflict and flow relations are not required to be disjoint as opposed to stable event structures. Thus, a structure with two events e and e' with $e \prec e'$ and $e \# e'$ is a legitimate flow event structure.

Like for prime and stable event structures, the semantics of a flow event structure is determined by its family of configurations, ordered by set inclusion.

2.9. DEFINITION. (*Configuration of a flow event structure*)

An *event trace* σ of flow event structure $\mathcal{E} = (E, \#, \prec, l)$ is a sequence of events $e_1 \ldots e_n$ with $e_i \in E$ such that for all i, $0 < i \leqslant n$

$$(e_i \notin \mathsf{cfl}(\sigma_i\, e_i) \cup \overline{\sigma_i}) \;\wedge\; (\forall\, e : e \prec e_i \;\Rightarrow\; (\exists\, e' \in \overline{\sigma_i} : e' \prec e_i \;\wedge\; (e' = e \;\vee\; e' \# e))).$$

A set $C \subseteq E$ is a *configuration* iff there is an event trace σ such that $C = \overline{\sigma}$. $\square$

The constraint $e_i \notin \mathsf{cfl}(\sigma_i\, e_i)$ guarantees that self-conflicting events can never occur. As pointed out in [26] self-conflicting events cannot in general be removed from a flow event structure without changing its set of configurations. So, given a flow event structure $\mathcal{E}$ containing some self-conflicting events, it is impossible to construct a flow event structure without self-conflicting events that has the same family of configurations as $\mathcal{E}$. This is a rather awkward property for specifying behaviours—it is rather unnatural to be forced to introduce impossible events in order to be able to specify some desired behaviour. Impossible events might be useful, but it should always be possible to safely remove them.

2.2.4 Bundle event structures

Langerak studied the suitability of prime, flow, and stable event structures as a noninterleaving semantic model for the process algebra LOTOS [89, 90]. He concludes that all these event structures have some drawbacks, making them unattractive for this purpose. As an alternative he proposes *bundle event structures.* Bundle event structures share some advantages of the aforementioned models while avoiding some of their drawbacks. For example, in bundle event structures events can have different enablings, and self-conflicting events are not allowed.

Causality is represented by a relation $\mapsto$ between a set X of events, which are pairwise in conflict, and an event e. The interpretation is that if e happens in a system run, exactly one event in X has happened before (and caused e). This enables us to uniquely define a causal ordering between the events in a system run (i.e., absence of causal ambiguity, like in stable event structures). Pairs (X, e), such that $X \mapsto e$, are called *bundles*; X is also called a *bundle set.*

2.10. Definition. (*Bundle event structure*)

A *bundle event structure* $\mathcal{E}$ is a quadruple $(E, \#, \mapsto, l)$ with

- E, a set of *events*
- $\# \subseteq E \times E$, the (irreflexive and symmetric) *conflict* relation
- $\mapsto \subseteq \mathcal{P}(E) \times E$, the *bundle* relation
- $l : E \longrightarrow \mathcal{A}$, the *action-labelling* function

such that $\forall X \subseteq E, e \in E : X \mapsto e \;\Rightarrow\; (\forall e', e'' \in X : e' \neq e'' \;\Rightarrow\; e' \# e'')$. □

The constraint, referred to as the *stability constraint*, says that all events in a bundle set should be in conflict. Let BES denote the class of bundle event structures.

A bundle (X, e) is indicated by drawing an arrow from each element of X to e and connecting all arrows by small lines, analogous to the representation of enablings in stable event structures[2].

The above definition allows an empty bundle, $\varnothing \mapsto e$, to be defined. The interpretation of such bundle is that e can never happen; e is an *impossible event.* Events pointed to by empty bundles are comparable with self-conflicting events in flow event structures, but have—as opposed to self-conflicting events—the pleasant property that they can always be eliminated while preserving the semantics (in terms of configurations). An alternative way to specify impossible events is by $\{ e \} \mapsto e$. Also these bundles can always be eliminated while preserving the semantics.

2.11. Definition. For σ a sequence of events let

$$\mathsf{sat}(\sigma) \triangleq \{ e \in E \mid \forall X \subseteq E : X \mapsto e \;\Rightarrow\; X \cap \overline{\sigma} \neq \varnothing \}.$$

□

Stated in words, $\mathsf{sat}(\sigma)$ is the set of events that have a causal predecessor in σ for all bundles pointing to them. That is, for events in $\mathsf{sat}(\sigma)$ all bundles are 'satisfied' by σ.

[2]It should be recalled, however, that $X \vdash e$ in stable event structures means that when e happens *all* events in X have happened before, whereas $X \mapsto e$, although represented in the same way as $X \vdash e$, means that when e happens precisely *one* event in X has happened before.

2.12. DEFINITION. (*Event trace of a bundle event structure*)

An *event trace* σ of bundle event structure $\mathcal{E} = (E, \#, \mapsto, l)$ is a sequence of events $e_1 \dots e_n$ with $e_i \in E$, satisfying for all i, $0 < i \leqslant n$

$$e_i \in \mathsf{sat}(\sigma_i) \setminus (\mathsf{cfl}(\sigma_i) \cup \overline{\sigma_i}) \quad .$$

A set $C \subseteq E$ is a *configuration* iff there is an event trace σ such that $C = \overline{\sigma}$. □

Event traces are conflict-free, as expected, and each event in the event trace is preceded in the sequence by a causal predecessor for each bundle pointing to it. Event traces correspond to linearizations of system runs.

2.13. EXAMPLE. Some bundle event structures are depicted in Figure 2.2. Event structure (c) has bundles $\{ e_a, e_b \} \mapsto e_c$, $\{ e_b \} \mapsto e_d$, and $\{ e_f \} \mapsto e_d$, and a conflict between e_a and e_b. Example event traces of this event structure are $e_a\, e_f\, e_c$, $e_b\, e_c$, and $e_f\, e_b\, e_d\, e_c$. □

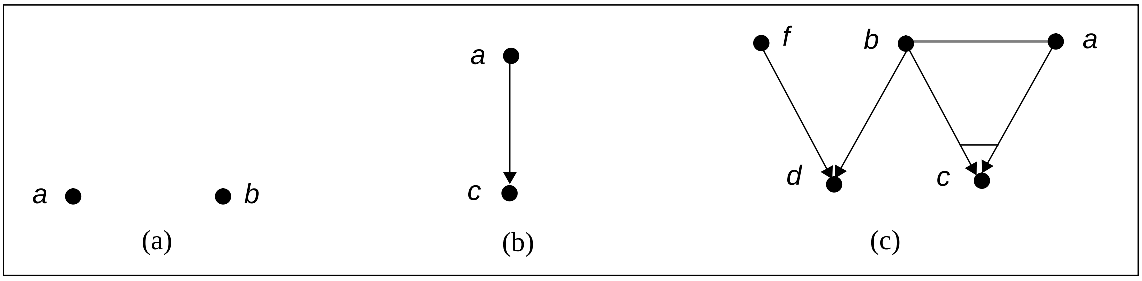


Figure 2.2: Example bundle event structures.

The semantics of bundle event structures is defined using labelled partially ordered sets (lposets), cf. Definition 1.8. We will not consider how these lposets can be generated, as this procedure is analogous to that of extended bundle event structures, see Section 2.3.2. An lposet keeps track of the causal dependencies between events. An event trace abstracts from these dependencies and is a linearization of an lposet, since it keeps track of the ordering of events. A configuration abstracts from the ordering of events, and thus it is the less discriminating (and simplest) notion of these three.

2.14. NOTATION. For an event structure $\mathcal{E}$ let $T(\mathcal{E})$ denote the set of event traces of $\mathcal{E}$, $C(\mathcal{E})$ the set of configurations of $\mathcal{E}$, and $L(\mathcal{E})$ the family of lposets of $\mathcal{E}$. □

For bundle event structures, having the same set of configurations is equivalent to having the same set of lposets. This result indicates that it suffices to use families of configurations as an underlying semantical model for bundle event structures.

2.15. THEOREM. $\forall\, \mathcal{E}, \mathcal{E}' \in \mathsf{BES} : C(\mathcal{E}) = C(\mathcal{E}') \iff L(\mathcal{E}) = L(\mathcal{E}')$.

PROOF. See [89, Theorem 5.4.2]. □

Since families of configurations can be used as an underlying semantical model for prime, flow, stable, and bundle event structures, the expressivity of these models can be compared in terms of configurations. From [89] we recall that, using $\sqsubset$ for denoting 'is strictly less expressive than', prime $\sqsubset$ bundle $\sqsubset$ flow $\sqsubset$ stable.

2.3 Extended bundle event structures

This section discusses extended bundle event structures. Section 2.3.1 introduces this type of event structures. Section 2.3.2 presents two recipes to deduce lposets from such event structures. Section 2.3.3 defines the status of an extended bundle event structure after the occurrence of a sequence of events, and Section 2.3.4 presents some simple, though useful transformation rules.

2.3.1 What are extended bundle event structures?

In order to model the disrupt operator $[>$ the symmetric conflict relation in bundle event structures is replaced by an *asymmetric conflict* relation, denoted by $\rightsquigarrow$.[3] The intuitive meaning of $e \rightsquigarrow e'$ is that (i) if e' occurs it disables the occurrence of e, and (ii) if e and e' both occur in a single system run then e causally precedes e'.

2.16. Definition. (*Extended bundle event structure*)

An *extended bundle event structure* $\mathcal{E}$ is a quadruple $(E, \rightsquigarrow, \mapsto, l)$ with

- E, a set of *events*
- $\rightsquigarrow \subseteq E \times E$, the (irreflexive) *asymmetric conflict* relation
- $\mapsto \subseteq \mathcal{P}(E) \times E$, the *bundle* relation
- $l : E \longrightarrow \mathcal{A}$, the *action-labelling* function

such that $\forall\, X \subseteq E, e \in E : X \mapsto e \;\Rightarrow\; (\forall\, e', e'' \in X : e' \neq e'' \;\Rightarrow\; e' \rightsquigarrow e'')$. □

In the rest of this dissertation we assume extended bundle event structures to have a *finite* set of events, unless stated otherwise.

The constraint above is a straightforward generalization of the stability constraint in Definition 2.10. $e \rightsquigarrow e'$ is represented as a dotted arrow pointing from e to e', thus reflecting that in case both e and e' happen in a single system run, there is a causal relation between the two (as if it were the case that $\{\, e \,\} \mapsto e'$). If $e \rightsquigarrow e'$ and $e' \rightsquigarrow e$ this is indicated using the same representation for $e \,\#\, e'$ in (bundle) event structures, i.e., a dotted line.

In the rest of this thesis EBES denotes the class of extended bundle event structures and we use $\mathcal{E}$, possibly subscripted and/or primed, to denote members of this class.

The definitions of configuration and event trace are a straightforward adaptation of the same notions for bundle event structures (cf. Definition 2.12). For technical convenience we introduce:

[3] The term asymmetric conflict does not mean that $e \rightsquigarrow e' \;\Rightarrow\; e' \not\rightsquigarrow e$ as it might suggest. $e \rightsquigarrow e'$ and $e' \rightsquigarrow e$ is allowed and is equivalent with $e \,\#\, e'$. The terminology 'asymmetric' is adopted from Langerak [89] and Pinna & Poigné [118].

2.17. Definition. (*Enabled events after σ*)

The set $\mathsf{en}(\sigma)$ of events that are enabled after σ is defined as

$$\mathsf{en}(\sigma) \triangleq \{ e \mid e \in \mathsf{sat}(\sigma) \setminus \overline{\sigma} \ \wedge \ \neg (\exists\, e_i \in \overline{\sigma} : e \rightsquigarrow e_i) \}.$$

□

e is enabled after σ if it does not occur in σ, if all bundles pointing to it are satisfied by events in σ, and if there is no event in σ that disables e.

2.18. Definition. (*Event trace of an extended bundle event structure*)

An *event trace* σ of extended bundle event structure $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$ is a sequence of events $e_1 \dots e_n$ with $e_i \in E$, satisfying $e_i \in \mathsf{en}(\sigma_i)$, for all i, $0 < i \leqslant n$.

A set $C \subseteq E$ is a *configuration* iff there is an event trace σ such that $C = \overline{\sigma}$. □

2.3.2 Families of lposets

The semantics of EBES is given by sets of lposets ordered under the prefix relation on lposets. This is convenient since all other semantics, such as pomset, multiset and interleaving semantics, can be defined on top of an lposet semantics.

We present two possible ways in which lposets can be generated. The first recipe is an *intensional* one in the sense that the bundles and asymmetric conflicts in $\mathcal{E}$ are used to deduce the causal dependencies. The second recipe is an *operational* (or, observational) one. This recipe allows to generate lposets from the event traces of $\mathcal{E}$ without referring to the structure of $\mathcal{E}$. Both recipes will be used in this dissertation. The lposets generated according to the intensional scheme are denoted by L°, the ones according to the operational scheme by $L^{\bullet}$.

2.19. Definition. For $C \in C(\mathcal{E})$ let $\prec_C \subseteq C \times C$ be the smallest relation satisfying, for all $e_i, e_j \in C$:

1. $(\exists\, X \subseteq E : e_i \in X \ \wedge \ X \mapsto e_j) \ \Rightarrow \ e_i \prec_C e_j$
2. $e_i \rightsquigarrow e_j \ \Rightarrow \ e_i \prec_C e_j \quad .$

□

Let $\prec_C^*$ be the reflexive and transitive closure of $\prec_C$, and let $<_\sigma$ be the precedence relation of events in event trace σ, that is, for $\sigma = e_1 \dots e_n$ we have $e_1 <_\sigma e_2 <_\sigma \dots <_\sigma e_n$.

2.20. Lemma. $\forall\, \sigma \in T(\mathcal{E}) : \prec_{\overline{\sigma}}^* \subseteq <_\sigma^*$.

Proof. See [89, Lemma 6.3.6]. □

Given this lemma it is now easy to verify that $\prec_C^*$ is a partial order on C.

2.21. Corollary. $\langle C, \prec_C^* \rangle$ is a poset.

Proof. $\prec_C^*$ is reflexive and transitive by definition. It remains to prove antisymmetry, i.e., for $e, e' \in C : e \prec_C^* e' \ \wedge \ e' \prec_C^* e \ \Rightarrow \ e = e'$. Let σ be an event trace such that $C = \overline{\sigma}$. Then, according to Lemma 2.20 we have $e \prec_C^* e' \ \wedge \ e' \prec_C^* e \ \Rightarrow \ e <_\sigma^* e' \ \wedge \ e' <_\sigma^* e$. Since $<_\sigma^*$ is a partial order it follows $e = e'$. □

The family of intensional lposets of $\mathcal{E}$, denoted $L^\circ(\mathcal{E})$, is defined as the set of all lposets corresponding to its configurations.

2.22. Definition. (*Intensional lposets of an extended bundle event structure*)

For $\mathcal{E} \in \mathsf{EBES} : L^\circ(\mathcal{E}) \ \triangleq \ \{ \langle C, \prec_C^*, l \restriction C \rangle \mid C \in C(\mathcal{E}) \}$. □

As a prerequisite to defining how to obtain lposets from $\mathcal{E}$ in an operational way we define

2.23. Definition. $\sigma, \sigma' \in T(\mathcal{E})$ are *configuration equivalent*, denoted $\sigma \sim \sigma'$, iff $\overline{\sigma} = \overline{\sigma}'$.[4] □

Lposets of $\mathcal{E}$ are now determined in an operational way as follows. Consider all $\sigma \in T(\mathcal{E})$ and consider its class of configuration-equivalent traces, $[\sigma]_\sim$. For each $\sigma' \in [\sigma]_\sim$ we take the reflexive and transitive closure of the precedence relation of events in σ', $<_{\sigma'}$, and consider all common orderings for any $\sigma' \in [\sigma]_\sim$.

2.24. Definition. (*Operational lposets of an extended bundle event structure*)

For $\mathcal{E} \in \mathsf{EBES} : L^\bullet(\mathcal{E}) \ \triangleq \ \{ \langle \overline{\sigma}, \bigcap_{\sigma' \in [\sigma]_\sim} <_{\sigma'}^*, l \restriction \overline{\sigma} \rangle \mid \sigma \in T(\mathcal{E}) \}$. □

It is easy to verify that $\bigcap_{\sigma' \in [\sigma]_\sim} <_{\sigma'}^*$ is a partial order on $\overline{\sigma}$. We sometimes let $L(\sigma)$ denote $\langle \overline{\sigma}, \bigcap_{\sigma' \in [\sigma]_\sim} <_{\sigma'}^*, l \restriction \overline{\sigma} \rangle$.

It turns out that the operational and intensional characterizations of lposets coincide. This means that all causal dependencies between events can be deduced from the sequential observations.

2.25. Theorem. $\forall \, \mathcal{E} \in \mathsf{EBES} : L^\circ(\mathcal{E}) = L^\bullet(\mathcal{E})$.

Proof. This follows from $\bigcap_{\sigma' \in [\sigma]_\sim} <_{\sigma'}^* = \prec_{\overline{\sigma}}^*$ for $\sigma \in T(\mathcal{E})$; see [89, Chapter 7]. □

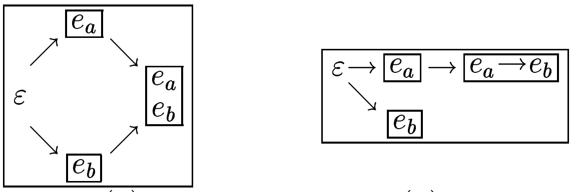
2.26. Example. Consider

(a)

(b)

The lposets (ordered under prefix) of these extended bundle event structures are

[4] Configuration equivalence is similar to the equivalence relation on sequential observations as defined by Mazurkiewicz [101]. He defines a binary independence relation on actions and considers sequential observations to be equivalent iff they contain the same actions with independent actions possibly swapped. Mazurkiewicz calls the equivalence classes traces, rather than their elements. An important distinction is that he considers actions whereas we consider events; for instance, $a\,;a$ and $a \,|||\, a$ cannot be distinguished by considering Mazurkiewicz' traces, but they can in our setting.



(a) (b)

□

The following theorem states that $\mathcal{E}$ and $\mathcal{E}'$ have identical event traces iff they have identical lposets. This is a nice result since it simplifies proof obligations—rather than proving that $\mathcal{E}$ and $\mathcal{E}'$ have identical lposets (i.e., are *lposet equivalent*) it suffices to prove their event trace equivalence.

2.27. Theorem. $\forall\, \mathcal{E}, \mathcal{E}' \in \mathsf{EBES} : L(\mathcal{E}) = L(\mathcal{E}') \iff T(\mathcal{E}) = T(\mathcal{E}')$.

Proof. See [89, Theorem 6.3.12]. □

For bundle event structures having the same configurations is equivalent to having the same lposets (see Theorem 2.15). This result does not hold for extended bundle event structures. For example, both event structures in Example 2.26 have family of configuration $\{ \varnothing, \{ e_a \}, \{ e_b \}, \{ e_a, e_b \} \}$. So, families of configurations cannot distinguish between these—from a causality point of view—different elements of EBES.[5]

We conclude this section by addressing the expressiveness of extended bundle event structures. Even on the level of families of configurations extended bundle event structures are strictly more expressive than bundle event structures, and consequently, than prime event structures. The relation with stable and flow event structures is less clear—there exist extended bundle event structures with a set of configurations that cannot be induced by any flow or stable event structure, and vice versa.

2.3.3 Remainder

During a run, or computation, of the system it is convenient to know the status or remaining behaviour. To that purpose we define the status of $\mathcal{E}$ after the occurrence of an event trace.

2.28. Definition. (*Remainder of an extended bundle event structure*)

$\mathcal{E}' = (E', \leadsto', \mapsto', l')$ is a *remainder* of $\mathcal{E}$ after $\sigma \in T(\mathcal{E})$, denoted $\mathcal{E}' = \mathcal{E}[\sigma]$, iff

- $E' = E \setminus \overline{\sigma}$
- $\leadsto' = \leadsto \cap (E' \times E')$
- $\mapsto' = (\mapsto \setminus \{ (X, e) \mid X \mapsto e \ \wedge \ X \cap \overline{\sigma} \neq \varnothing \}) \cup \{ (\varnothing, e) \mid \exists\, e' \in \overline{\sigma}, e \in E' : e \leadsto e' \}$
- $l' = l \restriction E'$.

□

[5] This also means that it is impossible to model asymmetric conflicts without copying events in prime, stable, or flow event structures. This impossibility has been argued in [89, Chapter 6].

It follows that for each bundle $(X, e) \in \mapsto'$ that $X \subseteq E'$ and $e \in E'$, because if $X \not\subseteq E'$ then $X \cap \overline{\sigma} \neq \varnothing$ so $(X, e) \notin \mapsto'$, and if $e \notin E'$ then $e \in \overline{\sigma}$, say $e = e_i$, but then $X \cap \overline{\sigma_i} \neq \varnothing$, so $(X, e) \notin \mapsto'$. It is not difficult to check that the remainder is an extended bundle event structure.

The intuitive interpretation of the above definition is as follows. First, all events in σ are removed from E and the conflicts between the remaining events are retained. Then, each event $e \in E$ that is disabled by some event in σ cannot happen anymore, and is made impossible by introducing an empty bundle pointing to it. Each bundle $X \mapsto e$ such that $X \cap \overline{\sigma} \neq \varnothing$ is removed, because the condition that this bundle poses, namely some event in X should have happened before e can happen, has now been satisfied.

2.29. Example. The remainder of an extended bundle event structure is exemplified in Figure 2.3. □

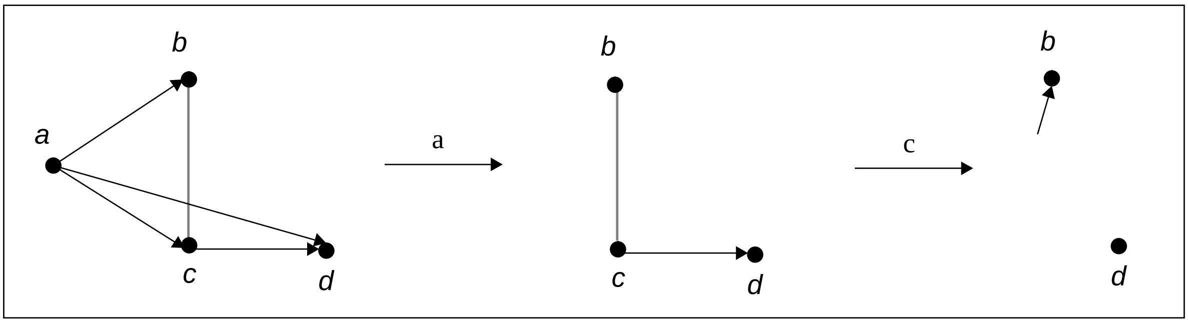


Figure 2.3: Example remainder of an extended bundle event structure.

We have the following correctness result concerning the remainder. It says that if $\mathcal{E}$ can evolve into $\mathcal{E}'$ by executing σ then σ' is a trace of $\mathcal{E}'$ iff $\sigma\,\sigma'$ is a trace of $\mathcal{E}$. This implies that $\mathcal{E}$ after σ does not allow evolutions that are disallowed by $\mathcal{E}$. In addition, it states that the lposet induced by $\sigma\,\sigma'$ is an extension of the lposet induced by σ.

2.30. Theorem. *Correctness of remainder*

For $\sigma \in T(\mathcal{E})$ and σ' a sequence of events:

1. $\sigma' \in T(\mathcal{E}[\sigma]) \Longleftrightarrow \sigma\,\sigma' \in T(\mathcal{E})$
2. $\sigma' \in T(\mathcal{E}[\sigma]) \;\Rightarrow\; L(\sigma)$ is a prefix of $L(\sigma\,\sigma')$.

Proof. Follows directly from [89, Theorem 6.3.9]. □

2.3.4 Transformation rules

This section presents some transformation rules for extended bundle event structures that can be used to transform $\mathcal{E}$ into $\mathcal{E}'$ such that $L(\mathcal{E})$ equals $L(\mathcal{E}')$. The rules involve only part of the event structure at hand. Each rule is defined in a pictorial form; the formalization and correctness proofs of these rules is not relevant here and can be found in [89]. Each picture shows only the relevant part of an event structure, that is, the part that is not depicted does not affect the validity of the presented rule. For the representation of the transformation rules we use the following notation.

2.31. NOTATION. Sets of events are represented by circles. A bundle $X \mapsto e$ is represented by a directed arrow from the circle representing X to event e. In addition, for $X, Y \subseteq E$, and $X, Y \neq \varnothing$ we have:

- $X \rightsquigarrow Y \triangleq (\forall\, e \in X, e' \in Y : e \rightsquigarrow e')$. This is represented by a dotted arrow from X to Y.
- $X \mapsto Y \triangleq (\forall\, e \in Y : X \mapsto e)$. This is represented by an arrow from X to Y.

□

The transformation rules are depicted in Figure 2.4. The sub-bundle removal rule is not an elementary rule, but can be derived from the symmetric conflict inheritance and bundle redundancy I rules. The rules facilitate the separation of all impossible events in an extended bundle event structure. The following theorem shows that all the isolated impossible events can be safely eliminated.

2.32. THEOREM. *Removal of impossible events*

Let $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$ with $e \notin E$, and let $a \in \mathcal{A}$. Then $\mathcal{E}$ is lposet equivalent with $(E \cup \{\, e \,\}, \rightsquigarrow, \mapsto \cup \{\, (\varnothing, e) \,\}, l \cup \{\, (e, a) \,\})$.

PROOF. Straightforward and omitted. □

Although the transformation rules are not complete they are useful to remove the major undesirable aspects from event structures, such as impossible events (e with $\varnothing \mapsto e$), and cyclic bundles ($X \mapsto \ldots \mapsto X$). Redundancy in bundles can also always be eliminated.

2.4 Causality-based semantics of PA

In this section we show how extended bundle event structures can be used to provide a causality-based semantics to PA in a compositional way. We define a function, denoted $\mathcal{E}[\![\ ]\!]$, that associates to each term $B \in$ PA an element of EBES. This mapping is adopted from [89, Chapter 6]. The set $\mathcal{A}$ of action labels of events equals $\mathsf{Act}^{\tau,\delta}$.

The initial events and successful termination events of an extended bundle event structure are defined as follows. The initial events are those events that have no bundle pointing to them. Let $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$.

2.33. DEFINITION. (*Initial events*)

The set of initial events of $\mathcal{E}$ is defined by $\mathit{init}(\mathcal{E}) \triangleq \{\, e \in E \mid \neg\,(\exists\, X \subseteq E : X \mapsto e) \,\}$.
□

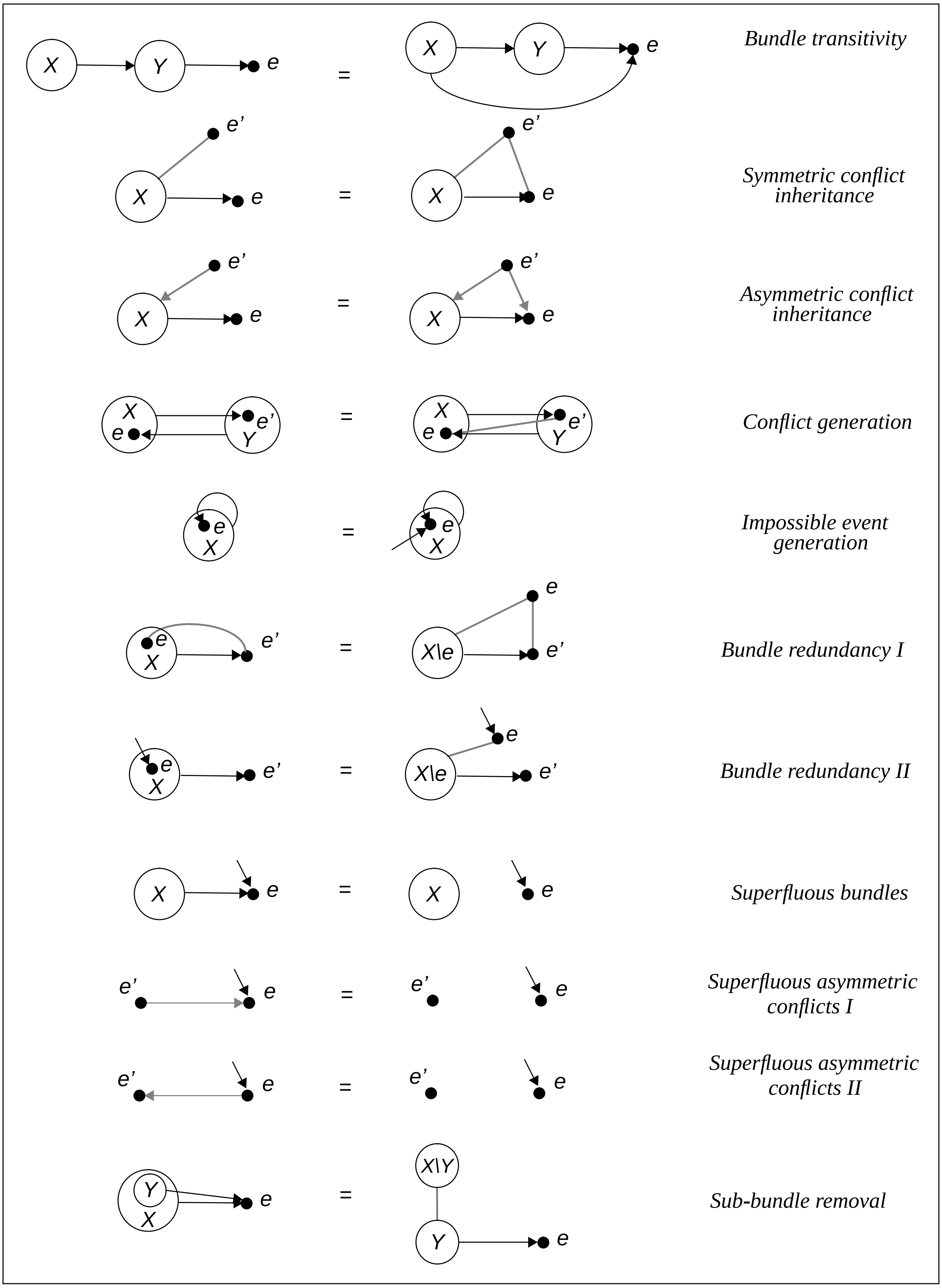


Figure 2.4: Transformation rules for extended bundle event structures.

Notice that $init(\mathcal{E})$ equals the set of enabled events after the empty trace, i.e., $\mathsf{en}(\varepsilon)$. Successful termination events are events that are labelled with δ, the successful termination action.

2.34. Definition. (*Successful termination events*)

The set of successful termination events of $\mathcal{E}$ is defined by $exit(\mathcal{E}) \triangleq \{ e \in E \mid l(e) = \delta \}$. □

$\mathcal{E}[\![\ ]\!]$ is defined recursively according to the following definitions. We suppose there is an infinite universe E_U of events. In the rest of this section let $\mathcal{E}[\![B_i]\!] = \mathcal{E}_i = (E_i, \leadsto_i, \mapsto_i, l_i)$, for i=1, 2 with $E_1 \cap E_2 = \varnothing$. (If $E_1 \cap E_2 \neq \varnothing$ then a suitable event renaming can be applied extended to $\leadsto$, $\mapsto$ and l.)

2.35. Definition. (*Semantics of* $\mathbf{0}$, $\surd$, $a\,;$, *and* $+$)

$$\begin{array}{rcl}
\mathcal{E}[\![\mathbf{0}]\!] & \triangleq & (\varnothing, \varnothing, \varnothing, \varnothing) \\
\mathcal{E}[\![\surd]\!] & \triangleq & (\{ e_\delta \}, \varnothing, \varnothing, \{ (e_\delta, \delta) \}) \text{ for some } e_\delta \in E_U \\
\mathcal{E}[\![a\,;\, B_1]\!] & \triangleq & (E, \leadsto_1, \mapsto, l_1 \cup \{ (e_a, a) \}) \text{ where} \\
E & = & E_1 \cup \{ e_a \} \text{ for some } e_a \in E_U \setminus E_1 \\
\mapsto & = & \mapsto_1 \cup (\{ \{ e_a \} \} \times init(\mathcal{E}_1)) \\
\mathcal{E}[\![B_1 + B_2]\!] & \triangleq & (E_1 \cup E_2, \leadsto, \mapsto_1 \cup \mapsto_2, l_1 \cup l_2) \text{ where} \\
\leadsto & = & \leadsto_1 \cup \leadsto_2 \cup (init(\mathcal{E}_1) \times init(\mathcal{E}_2)) \cup (init(\mathcal{E}_2) \times init(\mathcal{E}_1)).
\end{array}$$

□

The semantics of $\mathbf{0}$ and $\surd$ is self-explanatory. In $\mathcal{E}[\![a\,;\, B_1]\!]$ a bundle is introduced from the new event e_a (labelled a) to all initial events in $\mathcal{E}_1$ as e_a causally precedes these events. $\mathcal{E}[\![B_1 + B_2]\!]$ is equal to $\mathcal{E}_1 \cup \mathcal{E}_2$ extended with mutual conflicts between all initial events of $\mathcal{E}_1$ and $\mathcal{E}_2$ such that in the resulting structure only either B_1 or B_2 can happen.

2.36. Example. Let Figure 2.5 (a) through (c) depict the event structures corresponding to B_1 through B_3, respectively. Then Figure 2.5 (d) and (e) depict $\mathcal{E}[\![a\,;\, B_1]\!]$ and $\mathcal{E}[\![B_2 + B_3]\!]$, respectively. □

2.37. Definition. (*Semantics of* $\setminus$, $[\,]$, $>>$ *and* $[>$)

$$\begin{array}{rcl}
\mathcal{E}[\![B_1 \setminus G]\!] & \triangleq & (E_1, \leadsto_1, \mapsto_1, l) \text{ where} \\
 & & (l_1(e) \in G \Rightarrow l(e) = \tau) \wedge (l_1(e) \notin G \Rightarrow l(e) = l_1(e)) \\
\mathcal{E}[\![B_1[H]]\!] & \triangleq & (E_1, \leadsto_1, \mapsto_1, H \circ l_1) \\
\mathcal{E}[\![B_1 >> B_2]\!] & \triangleq & (E_1 \cup E_2, \leadsto, \mapsto, l) \text{ where} \\
\leadsto & = & \leadsto_1 \cup \leadsto_2 \cup \{ (e, e') \mid e, e' \in exit(\mathcal{E}_1) \wedge e \neq e' \} \\
\mapsto & = & \mapsto_1 \cup \mapsto_2 \cup (\{ exit(\mathcal{E}_1) \} \times init(\mathcal{E}_2)) \\
l & = & ((l_1 \cup l_2) \setminus (exit(\mathcal{E}_1) \times \{ \delta \})) \cup (exit(\mathcal{E}_1) \times \{ \tau \}) \\
\mathcal{E}[\![B_1 [> B_2]\!] & \triangleq & (E_1 \cup E_2, \leadsto, \mapsto_1 \cup \mapsto_2, l_1 \cup l_2) \text{ where} \\
\leadsto & = & \leadsto_1 \cup \leadsto_2 \cup (E_1 \times init(\mathcal{E}_2)) \cup (init(\mathcal{E}_2) \times exit(\mathcal{E}_1)).
\end{array}$$

□

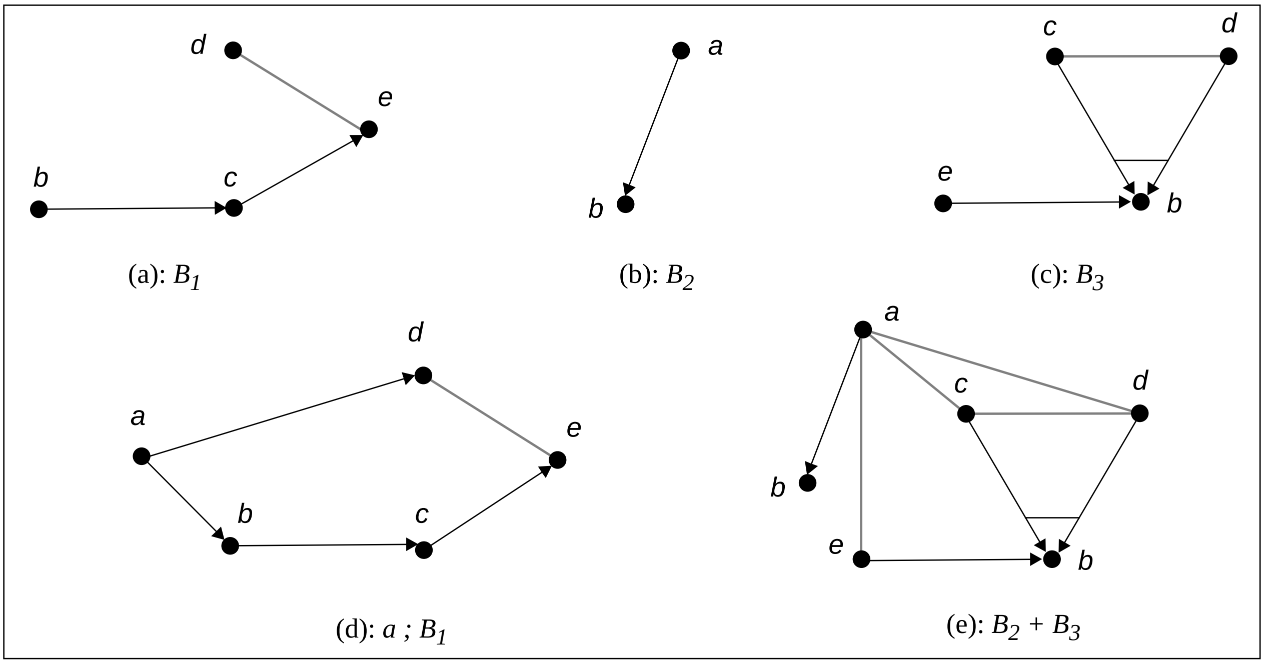


Figure 2.5: Examples of the semantics of action-prefix and choice.

$\mathcal{E}[\![\, B_1 \setminus G \,]\!]$ is identical to $\mathcal{E}_1$ except that events labelled with a label in G are now labelled with τ turning those events into internal ones. $\mathcal{E}[\![\, B_1[H] \,]\!]$ is defined similarly where events are relabelled according to H ($\circ$ denotes usual function composition).

$\mathcal{E}[\![\, B_1 >> B_2 \,]\!]$ is equal to $\mathcal{E}_1 \cup \mathcal{E}_2$ where bundles are introduced from the successful termination events of $\mathcal{E}_1$ to the initial events of $\mathcal{E}_2$. (To create bundles, mutual conflicts are introduced between the successful termination events of $\mathcal{E}_1$.) This corresponds with the fact that these initial events can only occur if B_1 has successfully terminated. The successful termination events of $\mathcal{E}_1$ are relabelled into internal events.

$\mathcal{E}[\![\, B_1 \,[> B_2 \,]\!]$ is equal to $\mathcal{E}_1 \cup \mathcal{E}_2$ extended with some additional asymmetric conflicts. First, each event in $\mathcal{E}_1$ may be disabled by an initial event of $\mathcal{E}_2$. This models that B_1 is disrupted once an initial event of B_2 happens. In addition, after the occurrence of a successful termination event in $\mathcal{E}_1$ no initial event of $\mathcal{E}_2$ can happen anymore.

2.38. EXAMPLE. Let Figure 2.6 (a) and (b) depict $\mathcal{E}[\![\, B_1 \,]\!]$ and $\mathcal{E}[\![\, B_2 \,]\!]$, respectively. $\mathcal{E}[\![\, B_1 >> B_2 \,]\!]$ and $\mathcal{E}[\![\, B_1 \,[> B_2 \,]\!]$ are depicted in Figure 2.6 (c) and (d), respectively. $\square$

We finally consider parallel composition. The events of $\mathcal{E}[\![\, B_1 \,||_G\, B_2 \,]\!]$ are constructed in the following way: an event e of $\mathcal{E}_1$ or $\mathcal{E}_2$ that does not need to synchronize is paired with the auxiliary symbol $*$, and an event which is labelled with an action in G^δ is paired with all events (if any) in the other process that are equally labelled. Thus events are pairs of events of $\mathcal{E}_1$ and $\mathcal{E}_2$, or with one component equal to $*$. Two events are now put in conflict if any of their components are in conflict, or if different events have a common component different from $*$ (such events appear if two or more events in one process synchronize with the same event in the other process). A bundle is introduced such that if we take the projection on the i-th component (i=1, 2) of all events in the bundle we obtain a bundle in $\mathcal{E}[\![\, B_i \,]\!]$.

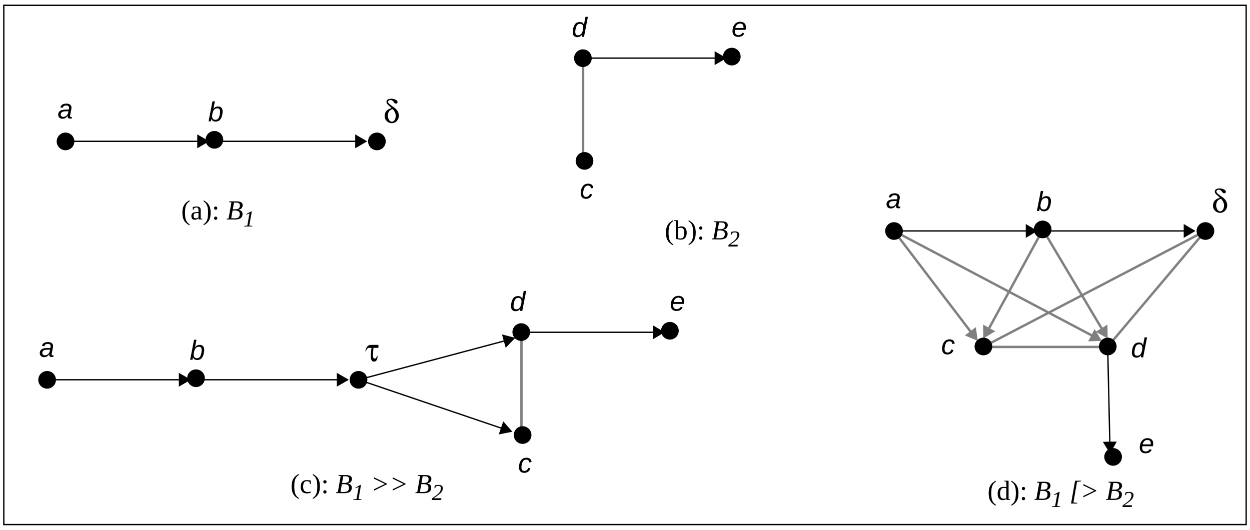


Figure 2.6: Examples of the semantics for enable and disrupt.

For $G \subseteq \mathsf{Act}$, $E_i^s \triangleq \{\, e \in E_i \mid l_i(e) \in G^\delta \,\}$ is the set of *synchronization* events and $E_i^f \triangleq E_i \backslash E_i^s$ the set of *non-synchronizing* events.

2.39. Definition. (*Semantics of* $||_G$)

$$
\begin{array}{rcl}
\mathcal{E}[\![\, B_1 \,||_G\, B_2 \,]\!] & \triangleq & (E, \leadsto, \mapsto, l) \text{ where} \\
E & = & (E_1^f \times \{\, * \,\}) \cup (\{\, * \,\} \times E_2^f) \,\cup \\
 & & \{\, (e_1, e_2) \in E_1^s \times E_2^s \mid l_1(e_1) = l_2(e_2) \,\} \\
(e_1, e_2) \leadsto (e_1', e_2') & \Leftrightarrow & (e_1 \leadsto_1 e_1') \ \vee\ (e_2 \leadsto_2 e_2') \ \vee \\
 & & (e_1 = e_1' \neq * \ \wedge\ e_2 \neq e_2') \vee (e_2 = e_2' \neq * \ \wedge\ e_1 \neq e_1') \\
X \mapsto (e_1, e_2) & \Leftrightarrow & \ \ (\exists\, X_1 \subseteq E_1 : X_1 \mapsto_1 e_1 \ \wedge\ X = \{\, (e, e') \in E \mid e \in X_1 \,\}) \\
 & & \vee\, (\exists\, X_2 \subseteq E_2 : X_2 \mapsto_2 e_2 \ \wedge\ X = \{\, (e, e') \in E \mid e' \in X_2 \,\}) \\
l((e_1, e_2)) & = & \mathsf{if}\ e_1 = * \ \mathsf{then}\ l_2(e_2)\ \mathsf{else}\ l_1(e_1).
\end{array}
$$

□

Note that $X \mapsto (e_1, e_2)$ is indeed a bundle, because, for instance, for $X = \{\, (e, e') \mid e \in X_1 \,\}$, it follows $\forall\, (e, e'), (e, e'') \in X : e' \neq e'' \ \Rightarrow\ (e, e') \leadsto (e, e'')$. By symmetry, a similar argument holds for bundles satisfying $X = \{\, (e, e') \mid e' \in X_2 \,\}$.

2.40. Example. The definition of $\mathcal{E}[\![\ ]\!]$ for parallel composition is exemplified in Figure 2.7. □

The semantics of $\mathcal{E}[\![\, P \,]\!]$ where $P := B$ is treated in Chapter 10.

2.41. Theorem. $\forall\, B \in \mathsf{PA} : \mathcal{E}[\![\, B \,]\!] \in \mathsf{EBES}$.

Proof. By induction on the structure of B. Routine and omitted. □

It appears that events in bundle sets of $\mathcal{E}[\![\, B \,]\!]$ are always equally labelled.

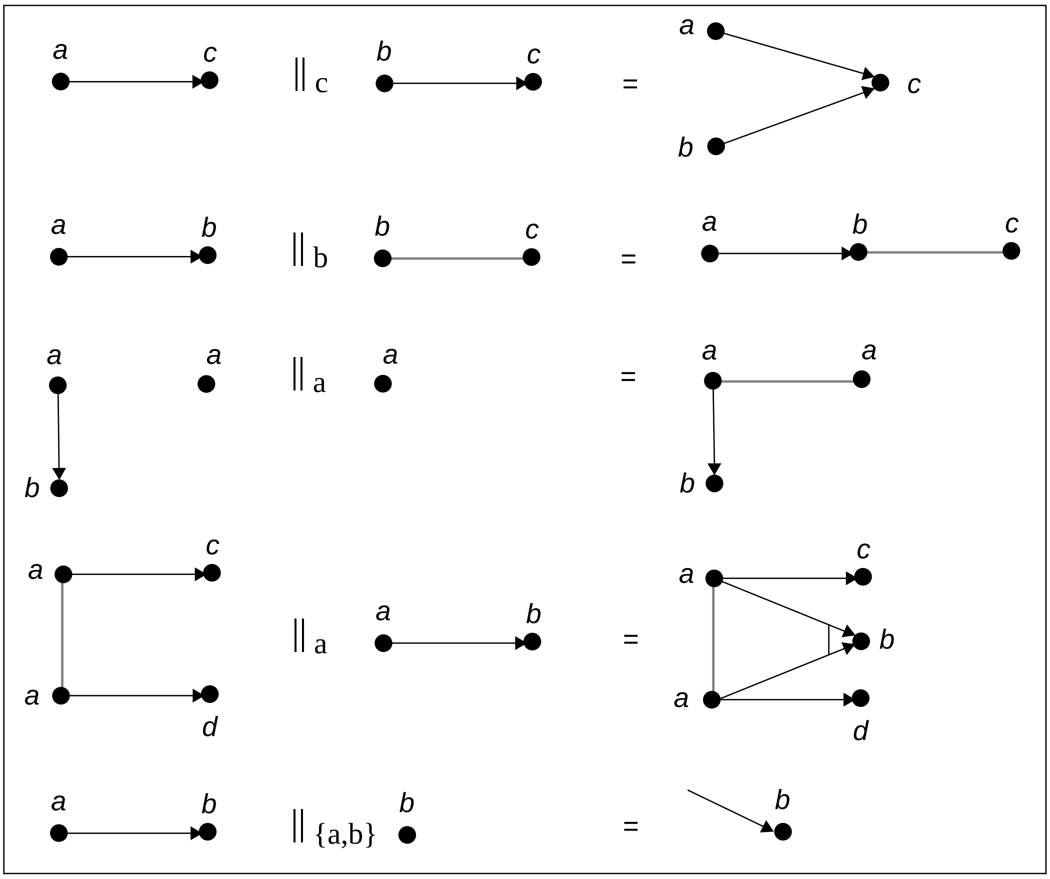


Figure 2.7: Examples of the semantics for parallel composition.

2.42. LEMMA. For $B \in \mathsf{PA}$ let $\mathcal{E}[\![B]\!] = (E, \rightsquigarrow, \mapsto, l)$. Then

$$\forall X \subseteq E, e, e', e'' \in E : (X \mapsto e \;\wedge\; e' \in X \;\wedge\; e'' \in X) \;\Rightarrow\; l(e') = l(e'') \quad .$$

PROOF. Straightforward by induction on the structure of B. □

In Chapters 4, 6, 7, and 8 we use a slightly different version of $\mathcal{E}[\![\ ]\!]$, denoted $\mathcal{E}'[\![\ ]\!]$. The need for this slight adaptation is explained in these chapters. Below we present the definition of $\mathcal{E}'[\![\ ]\!]$ and prove that any function that satisfies this definition is equivalent to $\mathcal{E}[\![\ ]\!]$ in the sense of having identical sets of lposets. (The differences with $\mathcal{E}[\![\ ]\!]$ are underlined.)

2.43. DEFINITION. (*Alternative semantics*)

$\mathcal{E}'[\![\ ]\!]$ is a function that satisfies

$$\begin{array}{rcl}
\mathcal{E}'[\![a\,;\, B_1]\!] & \triangleq & (E, \rightsquigarrow_1, \mapsto, l_1 \cup \{\, (e_a, a) \,\}) \text{ where} \\
E & = & E_1 \cup \{\, e_a \,\} \text{ for some } e_a \in E_U \setminus E_1 \\
\mapsto & = & \mapsto_1 \;\cup\; (\{\, \{\, e_a \,\} \,\} \times \underline{E') \text{ where } \mathit{init}(\mathcal{E}_1) \subseteq E' \subseteq E_1} \\
\mathcal{E}'[\![B_1 >> B_2]\!] & \triangleq & (E_1 \cup E_2, \rightsquigarrow, \mapsto, l) \text{ where} \\
\rightsquigarrow & = & \rightsquigarrow_1 \;\cup\; \rightsquigarrow_2 \;\cup \{\, (e, e') \mid e, e' \in \mathit{exit}(\mathcal{E}_1) \;\wedge\; e \neq e' \,\}
\end{array}$$

$$\begin{array}{rcl} \mapsto & = & \mapsto_1 \cup \mapsto_2 \cup \underline{(\{ exit(\mathcal{E}_1) \} \times E') \text{ where}} \\ & & \underline{init(\mathcal{E}_2) \subseteq E' \subseteq E_2} \\ l & = & ((l_1 \cup l_2) \setminus (exit(\mathcal{E}_1) \times \{ \delta \})) \cup (exit(\mathcal{E}_1) \times \{ \tau \}). \end{array}$$

For all other syntactic constructs let $\mathcal{E}'[\![B]\!] \triangleq \mathcal{E}[\![B]\!]$. □

The only difference between $\mathcal{E}'[\![\ ]\!]$ and $\mathcal{E}[\![\ ]\!]$ concerns the definition of the bundles for action-prefix and sequential composition. For instance, $\mathcal{E}'[\![B_1 >> B_2]\!]$ introduces a new bundle from $exit(\mathcal{E}_1)$ to all events in a set E', $init(\mathcal{E}_2) \subseteq E' \subseteq E_2$, whereas $\mathcal{E}[\![\ ]\!]$ introduces such bundles only to the initial events of $\mathcal{E}_2$. From the following theorem it follows that this is equivalent in terms of families of lposets.

2.44. THEOREM. $\forall B \in \mathsf{PA} : L(\mathcal{E}[\![B]\!]) = L(\mathcal{E}'[\![B]\!])$ for any $\mathcal{E}'[\![\ ]\!]$ satisfying Definition 2.43.

PROOF. The proof is by induction on the structure of B.
Base: For $\mathbf{0}$ and $\surd$ we have that $\mathcal{E}[\![B]\!] = \mathcal{E}'[\![B]\!]$ which proves the theorem.
Induction Step: By definition of $\mathcal{E}'[\![\ ]\!]$ it suffices to only consider action-prefix and enabling. We only provide the proof for action-prefix; the proof for enabling is similar and omitted. Suppose the theorem holds for B_1. Let $\mathcal{E} = \mathcal{E}[\![a ; B_1]\!]$, $\mathcal{E}_1 = \mathcal{E}[\![B_1]\!]$, $\mathcal{E}' = \mathcal{E}'[\![a ; B_1]\!]$, and $\mathcal{E}_1' = \mathcal{E}'[\![B_1]\!]$. For $init(\mathcal{E}_1) = E_1$ the theorem trivially holds since $\mathcal{E}[\![\ ]\!] = \mathcal{E}'[\![\ ]\!]$ in this case. Assume $init(\mathcal{E}_1) \neq E_1$. Consider $\mathcal{E}$, and introduce a new bundle $\{ e_a \} \mapsto e$ with e a non-initial event in $\mathcal{E}_1$ pointed to by X where X consists of initial events only. (Since $init(\mathcal{E}_1) \neq E_1$ it is easy to see that such event must exist.) According to the bundle transitivity rule (see Section 2.4) the resulting event structure is lposet equivalent to $\mathcal{E}$. This procedure is repeated by each time introducing a bundle $\{ e_a \} \mapsto e$ where e is an event in $E_1 \setminus A$ where A is the set of events in E_1 to which a bundle originating from e_a already exists. Obviously, such a procedure terminates when all events in E_1 have been 'visited' resulting in an event structure with $\{ e_a \} \mapsto e$ for all events $e \in E_1$. As all intermediate structures are lposet equivalent, this shows that introducing an additional bundle from e_a to each event in E' ($init(\mathcal{E}_1) \subseteq E' \subseteq E_1$) results in an event structure which is lposet-equivalent to $\mathcal{E}$. Together with the induction hypothesis this proves $L(\mathcal{E}') = L(\mathcal{E})$. □

As a result of this theorem we may safely interchange the use of $\mathcal{E}[\![\ ]\!]$ and any $\mathcal{E}'[\![\ ]\!]$ that satisfies Definition 2.43 whenever appropriate.

2.5 Event-based operational semantics for PA

In this section we define a transition system (in the sense of [120]) in which we keep track of the occurrence of actions, that is, events, in an expression of PA. This results in an *event transition system*. In order to define an event transition system we decorate each occurrence of an action-prefix or $\surd$ with an arbitrary but unique event occurrence identifier, denoted by a Greek letter. These occurrence identifiers play the role of event names. For instance, an expression like $a ; b + b$ becomes $a_\chi ; b_\psi + b_\xi$ and expression $a ; \surd >> b$ becomes $a_\chi ; \surd_\psi >> b_\xi$.

For parallel composition new event names can be created. If e is an event name of B and e' an event name in B', then possible new names for events in $B \,||_G\, B'$ are $(e, *)$ and $(*, e')$

for unsynchronized events where e (e') is independently performed by B (B') and (e, e') for synchronized events. The actual event names of the newly created events are in fact irrelevant (though technically convenient); the important aspect is that they are *unique*.

2.45. DEFINITION. (*Occurrence identifiers*)

Let Occ be an infinite set of occurrence identifiers. The set of events Ev is now defined as the smallest set satisfying

- $Occ \subseteq Ev$
- $\forall\, e \in Ev : (e, *) \in Ev \ \wedge \ (*, e) \in Ev$
- $\forall\, e, e' \in Ev : (e, e') \in Ev$.

□

We present a set of inference rules that define set of transition relations $\xrightarrow{(e,a)} \subseteq \mathsf{PA}^+ \times Ev \times \mathsf{Act}^{\tau,\delta} \times \mathsf{PA}^+$, where PA^+ denotes PA augmented with occurrence identifiers. $B \xrightarrow{(e,a)} B'$ denotes that behaviour B can perform event $e \in Ev$, labelled with action $a \in \mathsf{Act}^{\tau,\delta}$, and subsequently evolve into B'. The transition relation $\xrightarrow{(e,a)}$ is defined as the smallest relation closed under all inference rules defined in Table 2.1.

Notice that by omitting the occurrence identifiers from the expressions and the transition labels we obtain the standard interleaving inference rules of PA as presented in Chapter 1.

The relationship between the denotational semantics of PA in terms of event structures and the event-based operational semantics is as follows. Let $\mathsf{TS}(B)$ be the transition system obtained by applying the inference rules of Table 2.1 to B. We can also construct a transition system for $\mathcal{E}[\![B]\!]$ by having elements of EBES as states ($\mathcal{E}[\![B]\!]$ being the initial state) and having a transition from $\mathcal{E}$ to $\mathcal{E}'$ if $\mathcal{E}' = \mathcal{E}[\sigma]$ with $|\,\sigma\,| = 1$. The transitions are labelled with the event in σ and its action label. Let this transition system be called $\mathsf{ETS}(\mathcal{E}[\![B]\!])$. (For brevity, we do not elaborate on the formal definitions of these transition systems; these definitions are quite straightforward.)

2.46. THEOREM. $\forall\, B \in \mathsf{PA} : \mathsf{TS}(B) \sim \mathsf{ETS}(\mathcal{E}[\![B]\!])$.

PROOF. From [89, Theorem 7.5.3] it follows that $\mathsf{TS}(B)$ and $\mathsf{ETS}(\mathcal{E}[\![B]\!])$ are event trace equivalent. Since for each transition $B \xrightarrow{(e,a)} B'$ there is a unique way in which this transition is derived from the inference rules it follows that $\mathsf{TS}(B)$ and $\mathsf{ETS}(\mathcal{E}[\![B]\!])$ are strong bisimulation equivalent, see [89, Theorem 7.3.2]. □

A similar result has been obtained by Baier & Majster-Cederbaum [10] in the context of theoretical CSP (TCSP) and prime event structures. Due to the external choice operator in TCSP they obtain weak bisimulation equivalence rather than strong bisimulation equivalence.

$$\frac{}{\surd_\xi \xrightarrow{(\xi,\delta)} \mathbf{0}} \qquad \frac{}{a_\xi \,;\, B \xrightarrow{(\xi,a)} B}$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1'}{B_1 + B_2 \xrightarrow{(\xi,a)} B_1'} \qquad \frac{B_2 \xrightarrow{(\xi,a)} B_2'}{B_1 + B_2 \xrightarrow{(\xi,a)} B_2'}$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1'}{B_1 >> B_2 \xrightarrow{(\xi,a)} B_1' >> B_2} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{(\xi,\delta)} B_1'}{B_1 >> B_2 \xrightarrow{(\xi,\tau)} B_2}$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1'}{B_1 \,[> B_2 \xrightarrow{(\xi,a)} B_1' \,[> B_2} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{(\xi,\delta)} B_1'}{B_1 \,[> B_2 \xrightarrow{(\xi,\delta)} B_1'}$$

$$\frac{B_2 \xrightarrow{(\xi,a)} B_2'}{B_1 \,[> B_2 \xrightarrow{(\xi,a)} B_2'}$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,*),a)} B_1' \,||_G\, B_2} \quad (a \notin G^\delta) \qquad \frac{B_2 \xrightarrow{(\xi,a)} B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((*,\xi),a)} B_1 \,||_G\, B_2'} \quad (a \notin G^\delta)$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1' \ \wedge \ B_2 \xrightarrow{(\psi,a)} B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,\psi),a)} B_1' \,||_G\, B_2'} \quad (a \in G^\delta)$$

$$\frac{B \xrightarrow{(\xi,a)} B'}{B \setminus G \xrightarrow{(\xi,a)} B' \setminus G} \quad (a \notin G) \qquad \frac{B \xrightarrow{(\xi,a)} B'}{B \setminus G \xrightarrow{(\xi,\tau)} B' \setminus G} \quad (a \in G)$$

$$\frac{B \xrightarrow{(\xi,a)} B'}{B[H] \xrightarrow{(\xi,H(a))} B'[H]}$$

Table 2.1: Event-based operational semantics for PA.

3 Disjunctive causality and interleaving

This chapter discusses two qualitative extensions of extended bundle event structures. In the first extension the stability constraint on bundles is dropped. The resulting model, called dual event structures, incorporates conjunctive causality—like all other event structures—and disjunctive causality—unlike most other event structures. The second extension comprehends the incorporation of an (irreflexive and symmetric) interleaving relation between events. We investigate for both models how lposets can be deduced and what transformation rules are supported. The expressiveness of the models is compared with the event structures of Chapter 2.

3.1 Introduction

Causal dependencies between events can be of different nature. The most basic notion is a binary relation, $<$ say, between events, where $e_a < e_c$ means that e_a enables e_c (in process algebra we would write $a\,;c$). When, in addition, we have $e_b < e_c$ event e_c is enabled once both e_a and e_b have occurred (i.e., $(a\,|||\,b) >> c$). This type of causality is referred to as *conjunctive* causality: an event is enabled once all of its causal predecessors have occurred. Conjunctive causality is (in one form or the other) present in all types of event structures of Chapter 2. The natural complementary construct, called *disjunctive* causality, is that e_c is enabled once either e_a or e_b has occurred (similar to $(a+b) >> c$). Using disjunctive causality it can be expressed that an event is enabled once some event out of a number of potential causal predecessors has happened. A similar terminology is adopted by Gunawardena [60]. He refers to conjunctive and disjunctive causality as AND and OR causality, respectively.

Extended bundle event structures support the following types of causalities (see Figure 3.1):

- conjunctive causality—if $X \mapsto e$ and $Y \mapsto e$ then $(X$ 'and' $Y) \mapsto e$;
- (exclusive) disjunctive causality—if $\{\,e, e'\,\} \mapsto e''$ then e'' is enabled once either e or e' has occurred with the restriction that either e or e' can occur but not both (this is due to the stability constraint).

Since the combination of bundles pointing to the same event leads to a conjunction of enabling constraints we might say that extended bundle event structures require the specification of causalities in conjunctive normal form. For instance, if $X = \{\,e_a, e_b\,\}$ and $Y = \{\,e_c\,\}$ we have that $X \mapsto e$ and $Y \mapsto e$ is an enabling condition which denotes $((e_a$ 'or' $e_b)$ 'and' $e_c)$. An overview of the types of causalities in event structure models is given in Table 3.1.

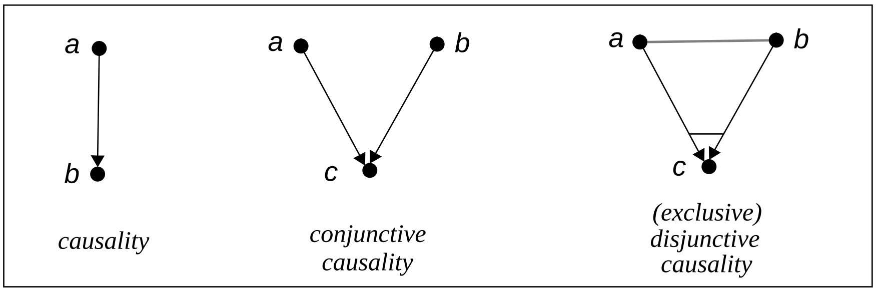


Figure 3.1: Types of causalities in extended bundle event structures.

	types of causality			
event structure	basic	'and'	'or'	normal form
prime	$e \leqslant e'$	+	-	C
stable	$X \vdash e$	+	exclusive	D
flow	$e \prec e'$	+	exclusive	C
(extended) bundle	$X \mapsto e$	+	exclusive	C
dual	$X \mapsto e$	+	+	C

('+' is present, '-' is absent,
C = conjunctive, and D = disjunctive)

Table 3.1: Types of causalities in event structures.

As already pointed out by Gunawardena [60] it is interesting to observe that formal models for concurrency mostly focus on conjunctive causality, while disjunctive causality has received only scant attention. This does, for instance, also hold for Petri nets where the incorporation of disjunctive causalities gives rise to unsafe nets (and the modelling of disablings $\rightsquigarrow$ gives rise to self-loops or inhibitor arcs), see e.g., Katoen [81]. Due to the conflict inheritance property prime event structures do not support disjunctive causality at all; other event structure models do allow alternative enablings of events, but do not support disjunctive causality in its full flavour—due to stability-like constraints alternative enablings are required to be in mutual conflict, such that in a system run only one of these alternative enablings can happen. Using Gunawardena's jargon this is best described as XOR causality.

Besides this observation the relevance of disjunctive causality has been argued in different application fields, such as the design of distributed systems [17, 46, 145], the design and analysis of speed-independent circuits [157], and the specification of business processes such as workflow management systems [41]. In the first (and main) part of this chapter we therefore drop the stability constraint from extended bundle event structures such that disjunctive causality is no longer restricted to be exclusive. Since the resulting model supports the dual conjunctive and disjunctive causalities we baptized this model *dual event structures.* We investigate for this new type of event structures how notions like event trace, remainder and families of lposets are defined and consider several transformation rules that preserve correctness in terms of lposets. The expressiveness of this model is compared to the other event structure models of Chapter 2.

Ferreira Pires *et al.* [46, 145] use a notation based on different types of causality to support the design of distributed systems. They include a mechanism to express the interleaving of events which is used to model that events can happen in any order but are not independent, i.e., they should not occur at the same time. Such scenarios typically appear in mutual exclusion situations. Inspired by this work we equip in the second part of this chapter dual event structures with a symmetric (irreflexive) *interleaving relation* between events, denoted by $\rightleftharpoons$. The intuitive interpretation of $e \rightleftharpoons e'$ is that e and e' are interleaved, e being caused by e' when e' occurs before e, and vice versa when e occurs before e'. Using such relation interleaving of events can be represented in dual event structures without having the need for copying events while retaining the symmetric nature of interleaving. A similar concept is presented by Zwiers and Janssen [159] and Wehrheim [151] who use a global symmetric dependency relation on actions (rather than events).

3.2 Disjunctive causality

The principle of (and need for) disjunctive causality can best be illustrated by means of a simple example. We consider a system called *one-all* (adopted from Verhoeff [146]), with two inputs e_a and e_b and two outputs e_c and e_d, see Figure 3.2(a). In this system output e_c happens when *one* input has been received, whereas output e_d occurs when *all* inputs are received. Phrased otherwise, if e_d happens then both e_a and e_b should have occurred, whereas if e_c happens either e_a or e_b or both have occurred. The obvious representation of the enabling

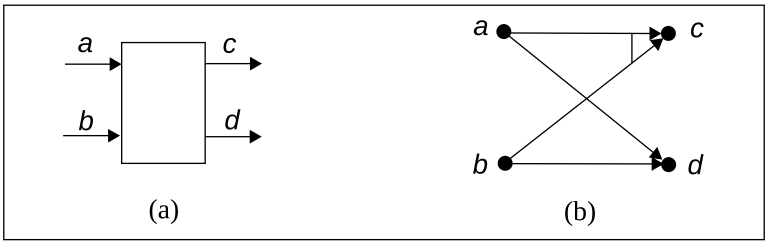


Figure 3.2: A simple system requiring disjunctive causality.

of e_c in extended bundle event structures, $\{ e_a, e_b \} \mapsto e_c$, requires e_a and e_b to be in mutual conflict, which is obviously not the case. This problem can be solved by copying event e_c, one copy for each alternative enabling, and putting these copies in mutual conflict. A drawback of this solution is that it requires copying of events and leads to an explosion of the number of events in general—if there are N alternative enablings of e we need N copies of e, all mutually in conflict. At a conceptual level we also prefer the representation of e_c by a single event, not distinguishing between whether it is enabled by e_a or e_b.

Dropping the stability constraint enables an event structure as depicted in Figure 3.2(b). This entails that events in a bundle set X are no longer required to be in mutual conflict. The intuitive interpretation of $X \mapsto e$ now becomes: when event e happens, *at least* one event in

X has occurred. So, an event e is enabled when for each bundle $X \mapsto e$ some event in X has happened.

3.2.1 What are dual event structures?

In this section we formally define dual event structures, the type of event structures one obtains by dropping the stability constraint from extended bundle event structures. All other ingredients remain as they were:

3.1. Definition. (*Dual event structure*)

A *dual event structure* Δ is a quadruple $(E, \rightsquigarrow, \mapsto, l)$ with

- E, a set of *events*
- $\rightsquigarrow \subseteq E \times E$, the (irreflexive) *asymmetric conflict* relation
- $\mapsto \subseteq \mathcal{P}(E) \times E$, the *bundle* relation
- $l : E \longrightarrow \mathcal{A}$, the *action-labelling* function.

□

Dual event structures are represented in the same way as extended bundle event structures. DES denotes the class of dual event structures and we use Δ, possibly subscripted and/or primed, to denote members of this class.

The notions of event trace and configuration are defined in an analogous way as for extended bundle event structures (see Definition 2.18). For convenience we recall this definition. Let $T(\Delta)$ denote the set of event traces of Δ.

3.2. Definition. $\mathsf{en}(\sigma) \triangleq \{ e \mid e \in \mathsf{sat}(\sigma) \setminus \overline{\sigma} \ \wedge \ \neg (\exists\, e_i \in \overline{\sigma} : e \rightsquigarrow e_i) \}$. □

3.3. Definition. (*Event trace of a dual event structure*)

An *event trace* σ of dual event structure $\Delta = (E, \rightsquigarrow, \mapsto, l)$ is a sequence of events $e_1 \dots e_n$ with $e_i \in E$ satisfying $e_i \in \mathsf{en}(\sigma_i)$, for all $0 < i \leqslant n$.

A set $C \subseteq E$ is a *configuration* iff there is an event trace σ such that $C = \overline{\sigma}$. □

There is an important difference with extended bundle event structures that we like to point out. Due to the stability constraint for extended bundle event structures the following holds for each event trace $\sigma = e_1 \dots e_n$ and bundle $X \mapsto e_i$:

$$X \cap \overline{\sigma_i} \neq \varnothing \ \Rightarrow \ |\, X \cap \overline{\sigma_i} \,| = 1.$$

Stated in words, if there is some event in X present in σ_i then there is precisely one such event. A technically pleasant consequence of this property is that one can uniquely determine from σ the direct causal predecessors of each event in σ. This absence of causal ambiguity property is lost for dual event structures, as shown in the following example.

3.4. EXAMPLE. Two example dual event structures are depicted in Figure 3.3. Figure 3.3(a) has maximal configuration $\{e_a, e_b, e_c\}$ where e_c is enabled by e_a or e_b. Some of its event traces are $e_a, e_a\, e_c, e_b\, e_c, e_a\, e_b\, e_c, e_b\, e_a\, e_c$. Event trace $\sigma = e_a\, e_b\, e_c$ contains causal ambiguity since for $X = \{e_a, e_b\}$, $X \cap \overline{e_a\, e_b} = \{e_a, e_b\}$. As a result one cannot uniquely determine from σ whether event e_c causally depends on e_a or on e_b. Figure 3.3(a) is known in the literature as Winskel's switch [155].

In Figure 3.3(b) two bundles $\{e_a, e_b\} \mapsto e_d$ and $\{e_b, e_c\} \mapsto e_d$ determine the enablings of e_d. Possible event traces of this dual event structure are: $e_b\, e_d$, $e_a\, e_c\, e_d$, $e_c\, e_b\, e_d\, e_a$. $\square$

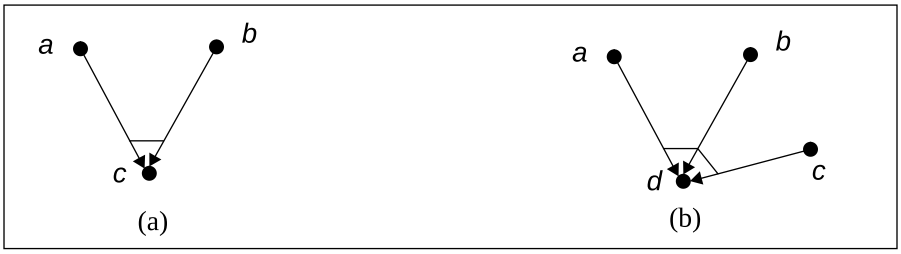


Figure 3.3: Two example dual event structures.

3.2.2 Families of lposets

The semantics of dual event structures is defined using families of lposets, non-empty sets of finite lposets ordered under the prefix relation. Like in Chapter 2 for extended bundle event structures we provide two views on lposets: an intensional one, denoted L°, which is determined by considering $\rightsquigarrow$ and $\mapsto$, and an operational one, denoted $L^\bullet$, which is derived from system observations, i.e., event traces. We first consider L° and start with some observations.

Consider Figure 3.3(a). For this dual event structure we would expect e_c to be causally dependent on either e_a or e_b. So, we consider $\boxed{\begin{array}{l} e_a \to e_c \\ e_b \end{array}}$ and $\boxed{\begin{array}{l} e_a \\ e_b \to e_c \end{array}}$ to be legitimate lposets. The reader might argue that it should also be possible for e_c to be causally dependent on both e_a and e_b, taking into account the lposet $\boxed{\begin{array}{lll} e_a & & \\ & \searrow & \\ e_b & \to & e_c \end{array}}$. We abandon this possibility because the occurrence of only e_a or e_b enables the occurrence of e_c. When we would incorporate this possibility it is not clear (to us) whether e_c being dependent on either e_a or e_b, and e_c being dependent on both e_a and e_b, should be modelled by the same event, or not.

The general idea for the definition of L° is that for each bundle pointing to some event e there must be precisely one event which is responsible for the satisfaction of this bundle.

In the rest of this section let $\Delta = (E, \rightsquigarrow, \mapsto, l)$.

3.5. DEFINITION. (*Intensional lposets of a dual event structure*)

The intensional lposets of Δ, denoted $L^\circ(\Delta)$, is the family of lposets $\langle C, \prec_C^*, l \restriction C \rangle$ where $\prec_C \subseteq C \times C$ is an acyclic relation[1] and $C \subseteq E$ is conflict-free (i.e., $\mathsf{CF}(C)$ holds), satisfying for all $e \in C$:

1. $\forall\, e' \in C : e' \rightsquigarrow e \;\Rightarrow\; e' \prec_C e$, and
2. $\exists\, F_e : \{\, X \mid X \mapsto e \,\} \longrightarrow \{\, e' \mid e' \prec_C e \,\}$ such that
   (a) $\{\, e' \mid e' \prec_C e \,\} \subseteq (\{\, F_e(X) \mid X \in \mathit{dom}(F_e) \,\} \cup \{\, e' \mid e' \rightsquigarrow e \,\})$, and
   (b) $\forall\, X \in \mathit{dom}(F_e) : F_e(X) \in X$.

□

The first constraint requires that conflicting events are ordered in the right way; this is identical to the case for extended bundle event structures (cf. Definition 2.19). Remark that, since C is conflict-free, it cannot appear that $e \rightsquigarrow e'$ and $e' \rightsquigarrow e$.

The second constraint ensures that for any bundle pointing to e there is precisely one event in that bundle (set) that is responsible for the satisfaction of this bundle. It requires for each e in C the existence of a (possibly empty) function F_e, the *bundle assignment* function of e, that associates with each bundle X pointing to e an event e' in X such that e' precedes e. Constraint 2.(a) ensures a kind of minimality: event e' can only precede e (under $\prec_C$) if $e' \rightsquigarrow e$, or if e' is responsible for the satisfaction of a bundle pointing to e. Constraint 2.(b) is a consistency constraint saying that only events can be responsible for the satisfaction of X if they are member of X.[2]

Two remarks are in order. First, it should be observed that it is not required for F_e to be injective, i.e., it is allowed for $X \mapsto e$ and $Y \mapsto e$ with $X \neq Y$ that $F_e(X) = F_e(Y) = e'$. In this case e' is an event that belongs to both X and Y, and that is responsible for the satisfaction of both bundles. Secondly, if $X \mapsto e$ and $X \mapsto e'$ it is not required that $F_e(X)$ equals $F_{e'}(X)$. This means that e and e' may be caused by different events in X.

The second constraint requires the existence of a function for each e in C that satisfies some conditions. The following lemma shows that such function always exists.

3.6. LEMMA. For $C \subseteq E$ with $\mathsf{CF}(C)$ and $\prec_C$ an acyclic relation satisfying constraint 1. of Definition 3.5 there exists for any $e \in C$ a function $F_e : \{\, X \mid X \mapsto e \,\} \longrightarrow \{\, e' \mid e' \prec_C e \,\}$ such that

1. $\{\, e' \mid e' \prec_C e \,\} \subseteq (\{\, F_e(X) \mid X \in \mathit{dom}(F_e) \,\} \cup \{\, e' \mid e' \rightsquigarrow e \,\})$, and
2. $\forall\, X \in \mathit{dom}(F_e) : F_e(X) \in X$.

PROOF. The proof is by contradiction. Let $C \subseteq E$ with $\mathsf{CF}(C)$ and $\prec_C$ an acyclic relation satisfying constraint 1. of Definition 3.5. Assume that for $e \in C$ all functions $F_e : \{\, X \mid X \mapsto e \,\} \longrightarrow \{\, e' \mid e' \prec_C e \,\}$ do not satisfy the second constraint of Definition 3.5. This could only be because:

[1] A relation is acyclic if its transitive closure is irreflexive.

[2] Remark that the second constraint of Definition 3.5 implies that $X \mapsto e \;\Rightarrow\; (\exists\, e' \in X \cap C : e' \prec_C e)$ as required for the lposets of extended bundle event structures; see also Definition 2.19.

1. $\{e' \mid e' \prec_C e\} \not\subseteq (\{F_e(X) \mid X \in dom(F_e)\} \cup \{e' \mid e' \rightsquigarrow e\})$. Then there exists an event e', say, $e' \prec_C e$ but $e' \not\rightsquigarrow e$ and $e' \notin \{F_e(X) \mid X \in dom(F_e)\}$, for all functions F_e. This means that there exists no bundle $X \mapsto e$ with $e' \in X$. But then $e' \prec_C e$ can only follow from $e' \rightsquigarrow e$, according to the first constraint of Definition 3.5. Contradiction.

2. $\exists X \in dom(F_e) : F_e(X) \notin X$, for all functions F_e. Then there is a bundle $X \mapsto e$ such that $X \cap \{e' \mid e' \prec_C e\} = \varnothing$. This contradicts with constraint 2.(a) of Definition 3.5.

$\square$

The next lemma shows that all elements in $L^\circ(\Delta)$ are lposets.

3.7. LEMMA. $\forall p \in L^\circ(\Delta) : p$ is an lposet.

PROOF. Let $p = \langle E_p, \leqslant_p, l_p \rangle$ be an element in $L^\circ(\Delta)$. It suffices to check whether $\leqslant_p$ is a partial order. Since $\leqslant_p$ is the reflexive and transitive closure of $\prec_p$ (i.e., $\prec_{E_p}$) it remains to check antisymmetry. Suppose $e, e' \in E_p$ such that $e \leqslant_p e'$ and $e' \leqslant_p e$. If $e \neq e'$ then we would have $e \prec_p^+ e'$ and $e' \prec_p^+ e$, where $\prec_p^+$ denotes the transitive closure of $\prec_p$. But then $\prec_p$ would be acyclic. Contradiction, so $e = e'$. $\square$

3.8. EXAMPLE. The maximal intensional lposets of Figure 3.4(a) are $\boxed{\begin{array}{l} e_a \rightarrow e_c \\ e_b \end{array}}$ and $\boxed{\begin{array}{l} e_a \\ e_b \rightarrow e_c \end{array}}$.

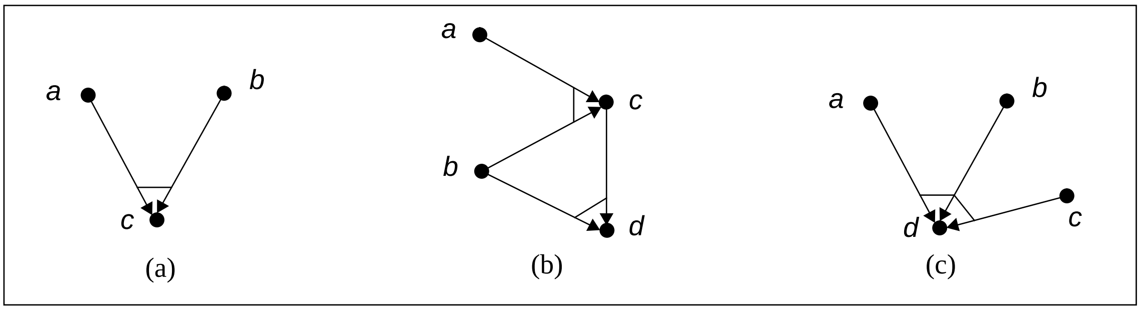


Figure 3.4: Three dual event structures.

Figure 3.4(b) has the following maximal intensional lposets:

$$\boxed{\begin{array}{l} e_a \rightarrow e_c \rightarrow e_d \\ e_b \end{array}},\ \boxed{\begin{array}{l} e_a \\ e_b \rightarrow e_c \rightarrow e_d \end{array}},\ \boxed{\begin{array}{l} e_a \rightarrow e_c \\ e_b \rightarrow e_d \end{array}},\ \text{and}\ \boxed{\begin{array}{lcl} e_a & & e_c \\ & \nearrow & \\ e_b & \rightarrow & e_d \end{array}}.$$

Finally, Figure 3.4(c) has the following maximal intensional lposets:

$$\boxed{\begin{array}{lcl} e_a & & \\ e_b & & \\ & \searrow & \\ e_c & \rightarrow & e_d \end{array}},\ \boxed{\begin{array}{lcl} e_b & & \\ e_a & & \\ & \searrow & \\ e_c & \rightarrow & e_d \end{array}},\ \boxed{\begin{array}{lcl} e_c & & \\ e_a & & \\ & \searrow & \\ e_b & \rightarrow & e_d \end{array}},\ \text{and}\ \boxed{\begin{array}{l} e_a \\ e_b \rightarrow e_d \\ e_c \end{array}}.$$

$\square$

It is hot hard to check that for each event trace σ of Δ there exists a (set of) corresponding lposet(s) that orders only the (possible) causal dependencies between events in σ. Moreover, each linearization of an lposet of Δ is an event trace of Δ. So,

3.9. LEMMA. $\forall\, \sigma \in E^* : (\exists\, p \in L^\circ(\Delta) : E_p = \overline{\sigma} \;\wedge\; \leqslant_p \subseteq <^*_\sigma) \iff \sigma \in T(\Delta)$.

PROOF. '$\Rightarrow$': Let $\sigma = e_1 \dots e_n$ and $\overline{\sigma} = E_p$ such that $\leqslant_p \subseteq <^*_\sigma$. The proof is by contradiction. Suppose $\sigma \notin T(\Delta)$. This could only be because one of the following reasons:

1. $e_i \rightsquigarrow e_j$ and $e_j <^*_\sigma e_i$. But $e_i \rightsquigarrow e_j$ implies $e_i \prec_p e_j$, and so $e_i \leqslant_p e_j$. Since $\leqslant_p \subseteq <^*_\sigma$ we have $e_i <^*_\sigma e_j$. From the antisymmetry of $<^*_\sigma$ it follows $e_i = e_j$. But $\rightsquigarrow$ is irreflexive. Contradiction.
2. $X \mapsto e_i$ and $X \cap \overline{\sigma_i} = \varnothing$. From $X \mapsto e_i$ it follows that $(\exists\, e \in X : e \prec_p e_i)$, and so $(\exists\, e \in X : e \leqslant_p e_i)$. Since $\leqslant_p \subseteq <^*_\sigma$ then $e <^*_\sigma e_i$, and so $X \cap \overline{\sigma_i} \neq \varnothing$. Contradiction.

'$\Leftarrow$': Straightforward and omitted. □

We sometimes let $L^\circ(\overline{\sigma})$ denote the set of lposets corresponding to $\overline{\sigma}$.

Dual event structures that have the same sets of intensional lposets have the same set of event traces.

3.10. THEOREM. $\forall\, \Delta, \Delta' \in \mathsf{DES} : L^\circ(\Delta) = L^\circ(\Delta') \;\Rightarrow\; T(\Delta) = T(\Delta')$.

PROOF. Assume $L^\circ(\Delta) = L^\circ(\Delta')$. Let $\sigma \in T(\Delta)$. From Lemma 3.9 it follows that for all $\sigma \in T(\Delta)$ there exists an lposet $\langle \overline{\sigma}, \leqslant, l \rangle$ in $L^\circ(\Delta)$ with $\leqslant \subseteq <^*_\sigma$. Since $L^\circ(\Delta) = L^\circ(\Delta')$ it follows that $\langle \overline{\sigma}, \leqslant, l \rangle$ in $L^\circ(\Delta')$, and according to Lemma 3.9 we have $\sigma \in T(\Delta')$. The proof for the opposite direction, i.e., $\sigma \in T(\Delta') \;\Rightarrow\; \sigma \in T(\Delta)$, is obtained by reversing the arguments Δ and Δ' in the above reasoning. □

The reverse implication does not hold. A counterexample is provided by

These two dual event structures are event trace equivalent, but, for instance, $\boxed{\begin{smallmatrix} e_a \rightarrow e_d \\ \nearrow \\ e_b \rightarrow e_c \end{smallmatrix}}$ is a maximal intensional lposet of the right-hand dual event structure, but not of the other. This entails that event traces are not sufficiently expressive as an underlying semantical model for dual event structures.

We now concentrate on the definition of $L^\bullet(\Delta)$, the operational characterization of the lposets of dual event structure Δ. Due to the possibility of causal ambiguity it is not possible to generate the lposets of a dual event structure according to the same operational procedure as for extended bundle event structures. For instance, the completed event traces for Figure 3.4(a) are $e_a\, e_b\, e_c, e_b\, e_a\, e_c, e_a\, e_c\, e_b$, and $e_b\, e_c\, e_a$. When we would follow the same procedure as in Definition 2.24 we obtain a single lposet in which e_c is completely causally independent. This is undesirable.

Observe that the sets of events preceding e_c in these event traces are $\{\, e_a \,\}$, $\{\, e_b \,\}$ and $\{\, e_a, e_b \,\}$. The minimal sets under $\subseteq$ represent the alternative enablings of e_c, and are called the *minimal enablings* of e.

3.11. DEFINITION. (*Minimal enablings of e in $[\sigma]_\sim$*)

For $\sigma \in T(\Delta)$ and $e \in E$, the minimal enablings of e in $[\sigma]_\sim$ are defined as

$$\mathsf{men}([\sigma]_\sim, e) \triangleq \{ \overline{\sigma_1} \mid \exists\, \sigma_2 : \sigma_1\, e\, \sigma_2 \in [\sigma]_\sim \;\wedge\; \neg\, (\exists\, \sigma_1'\, e\, \sigma_2' \in [\sigma]_\sim : \overline{\sigma_1}' \subset \overline{\sigma_1}) \}.$$

□

The lposets of Δ are now constructed in the following way. For each event trace σ of Δ we construct lposets of the form $\langle \overline{\sigma}, \leqslant, l \restriction \overline{\sigma} \rangle$, where $\leqslant$ is determined as follows. For each event e in $\overline{\sigma}$ we select a minimal enabling $\{ e_1', \ldots, e_k' \}$ from the set of minimal enablings $\mathsf{men}([\sigma]_\sim, e)$. Since all events in such minimal enabling must precede e in order to enable it, these events precede e under $\leqslant$: $e_1' \leqslant e, \ldots, e_k' \leqslant e$. In order to ensure transitivity we require that if e is part of a minimal enabling of e', say, and e' is part of a minimal enabling of e'' then e' is also part of the minimal enabling of e''.

For technical convenience we introduce:

3.12. DEFINITION. For $< \subseteq E \times E$ and $e \in E$ let $\downarrow_< e \triangleq \{ e' \in E \mid e' < e \}$. □

3.13. DEFINITION. (*Operational lposets of a dual event structure*)

The operational lposets of Δ, denoted $L^\bullet(\Delta)$, is the family of lposets

$$\bigcup_{\sigma \in T(\Delta)} \{ \langle \overline{\sigma}, <^\circ, l \restriction \overline{\sigma} \rangle \mid \forall\, e, e' \in \overline{\sigma} : \downarrow_< e \in \mathsf{men}(\overline{\sigma}, e) \;\wedge\; (e' \in \downarrow_< e \Rightarrow \downarrow_< e' \subseteq \downarrow_< e) \}.$$

□

$<^\circ$ denotes the reflexive closure of $<$.

3.14. LEMMA. $\forall\, p \in L^\bullet(\Delta) : p$ is an lposet.

PROOF. Let $p = \langle E_p, \leqslant_p, l_p \rangle$ an element of $L^\bullet(\Delta)$. We prove that $\leqslant_p$ is a partial order. From the previous definition it follows that $\leqslant_p = <^\circ$, where $<^\circ$ is the reflexive closure of $<$. It remains to check antisymmetry and transitivity.

1. To prove antisymmetry we derive:

$$\begin{array}{ll}
& e <^\circ e' \;\wedge\; e' <^\circ e \\
\Leftrightarrow & \{ \text{Definition 3.12} \} \\
& e \in \downarrow_{<^\circ} e' \;\wedge\; e' \in \downarrow_{<^\circ} e \\
\Leftrightarrow & \{ <^\circ \text{ is the reflexive closure of } < \} \\
& (e \in \downarrow_< e' \;\wedge\; e' \in \downarrow_< e) \vee e = e' \\
\Rightarrow & \{ \text{Definition 3.13} \} \\
& (e \in \downarrow_< e' \;\wedge\; e' \in \downarrow_< e \;\wedge\; \downarrow_< e \subseteq \downarrow_< e' \;\wedge\; \downarrow_< e' \subseteq \downarrow_< e) \vee e = e' \\
\Leftrightarrow & \{ \text{calculus} \} \\
& (e \in \downarrow_< e' \;\wedge\; e' \in \downarrow_< e \;\wedge\; \downarrow_< e = \downarrow_< e') \vee e = e'
\end{array}$$

$$
\begin{array}{ll}
\Rightarrow & \{ \text{ calculus } \} \\
& (e \in \downarrow_< e \ \wedge \ e' \in \downarrow_< e') \ \vee \ e = e' \\
\Rightarrow & \{ \ \downarrow_< e \in \mathsf{men}([\sigma]_\sim, e) \ \Rightarrow \ e \notin \downarrow_< e \ \} \\
& e = e' \quad .
\end{array}
$$

2. We prove that $<$ is transitive; this implies that $<^\circ$ is transitive.

$$
\begin{array}{ll}
& e < e' \ \wedge \ e' < e'' \\
\Leftrightarrow & \{ \text{ Definition 3.12 } \} \\
& e \in \downarrow_< e' \ \wedge \ e' \in \downarrow_< e'' \\
\Rightarrow & \{ \text{ Definition 3.13 } \} \\
& e \in \downarrow_< e' \ \wedge \ \ \downarrow_< e' \subseteq \downarrow_< e'' \\
\Rightarrow & \{ \text{ calculus } \} \\
& e \in \downarrow_< e'' \\
\Leftrightarrow & \{ \text{ Definition 3.12 } \} \\
& e < e'' \quad .
\end{array}
$$

□

3.15. EXAMPLE. Consider again the dual event structures of Figure 3.4. The maximal operational lposets of Figure 3.4(a) are $\boxed{\begin{array}{l} e_a \rightarrow e_c \\ e_b \end{array}}$ and $\boxed{\begin{array}{l} e_a \\ e_b \rightarrow e_c \end{array}}$. Figure 3.4(b) has the following maximal operational lposets:

$$
\boxed{\begin{array}{l} e_a \rightarrow e_c \rightarrow e_d \\ e_b \end{array}}, \ \boxed{\begin{array}{l} e_a \rightarrow e_c \\ e_b \rightarrow e_d \end{array}}, \text{ and } \boxed{\begin{array}{lll} e_a & & e_c \\ & \nearrow & \\ e_b & \rightarrow & e_d \end{array}}.
$$

Note that $\boxed{\begin{array}{l} e_a \\ e_b \rightarrow e_c \rightarrow e_d \end{array}}$ is not obtained as an operational lposet while it is an intensional lposet. Finally, Figure 3.4(c) has the following maximal operational lposets:

$$
\boxed{\begin{array}{lll} e_b & & \\ e_a & & \\ & \searrow & \\ e_c & \rightarrow & e_d \end{array}} \text{ and } \boxed{\begin{array}{l} e_a \\ e_b \rightarrow e_d \\ e_c \end{array}}.
$$

Also for this dual event structure some intensional lposets are not obtained operationally. □

The previous example shows that the operational and intensional characterizations of lposets do not have to coincide. This is not that surprising, since the operational perspective is constructed from event traces and we know from the above that having the same set of event traces does not imply having the same set of intensional lposets for dual event structures. The lposets that we do construct operationally are, however, correct lposets:

3.16. LEMMA. $\forall \Delta \in \mathsf{DES} : L^\bullet(\Delta) \subseteq L^\circ(\Delta)$.

PROOF. Let $p = \langle E_p, \leqslant_p, l_p \rangle$ an lposet of $L^\bullet(\Delta)$. We prove that p is an element of $L^\circ(C)$ by contradiction. It can only not be an lposet in $L^\circ(C)$ because either

1. $\exists\, e, e' \in E_p : e \rightsquigarrow e'$ and $e \not\leqslant_p e'$. If $e \rightsquigarrow e'$ then e should precede e' in each event trace σ with $\overline{\sigma} = E_p$. So, e belongs to each minimal enabling of e in $[\sigma]_\sim$. But then $e \in \downarrow_< e'$, and so $e \leqslant_p e'$. Contradiction.

2. There exists an event e for which one of the constraints for F_e is not fulfilled.

   (a) $\{\, e' \mid e' \leqslant_p e \,\} \not\subseteq (\{\, F_e(X) \mid X \in \mathit{dom}(F_e) \,\} \cup \{\, e' \mid e' \rightsquigarrow e \,\})$. Let $e' \leqslant_p e$, $e' \neq e$, and $e' \not\rightsquigarrow e$. The above inequality of sets for all functions F_e implies that there exists no bundle $X \mapsto e$ with $e' \in X$. Since $e' \leqslant_p e$ and $e \neq e'$ there is a minimal enabling of e. Because there exists no bundle $X \mapsto e$ with e, this can only be because $e' \rightsquigarrow e$. Contradiction.

   (b) $\exists\, X \in \mathit{dom}(F_e) : F_e(X) \notin X$, for all functions F_e. Then there is a bundle $X \mapsto e$ such that $X \cap \{\, e' \mid e' \leqslant_p e \,\} = \varnothing$. But if $X \mapsto e$ then each minimal enabling of e should contain some event in X. So, $X \cap \{\, e' \mid e' \leqslant_p e \,\} \neq \varnothing$. Contradiction.

$\square$

The relationship between L° and $L^\bullet$ can be identified in more detail. For deriving the operational lposets we have taken the minimal enablings of an event e as a starting-point. This reflects the idea that any event in a minimal enabling should causally precede e in order to let e happen. This perspective prevents, however, the generation of lposets with a bit more ordering than strictly necessary. E.g., for Figure 3.4(b) the lposet $\boxed{\begin{array}{l} e_a \\ e_b \to e_c \to e_d \end{array}}$ is not obtained since there exists an lposet $\boxed{\begin{array}{ccc} e_a & & e_c \\ & \nearrow & \\ e_b & \to & e_d \end{array}}$ that is less ordered.

3.17. DEFINITION. (*Smoothening*)
$\langle E, \leqslant, l \rangle$ is *smoother* than $\langle E', \leqslant', l' \rangle$ iff $E = E'$, $l = l'$ and $\leqslant' \subseteq \leqslant$. $\square$

That is, q is smoother than p if it has the same labelled events as p, but contains more ordering among the events; i.e., q is closer to being linear. Evidently, 'smoother than' is a partial order on lposets.

The operational lposets are the intentional ones that are minimal under the 'smoother than' relation.

3.18. THEOREM. $\forall\, \Delta \in \mathsf{DES} : L^\bullet(\Delta) = \{\, p \in L^\circ(\Delta) \mid \neg\, (\exists\, q \in L^\circ(\Delta) : p \text{ is smoother than } q) \,\}$.

PROOF. '$\subseteq$': Let $p \in L^\bullet(\Delta)$. From Lemma 3.16 it follows that $p \in L^\circ(\Delta)$. The proof is by contradiction. Let $p = \langle E_p, \leqslant_p, l_p \rangle$ and $q = \langle E_q, \leqslant_q, l_q \rangle$. Assume $q \in L^\circ(\Delta)$ such that p is smoother than q, i.e., p contains more ordering than q. Suppose $e \leqslant_p e'$, but $e \not\leqslant_q e'$. If $e \not\leqslant_q e'$ then we can construct an event trace σ with $\overline{\sigma} = E_p = E_q$ where e' precedes e. But then $e \notin \mathsf{men}([\sigma]_\sim, e')$ and so $e \not\leqslant_p e'$. Contradiction.

'$\supseteq$': Let $p \in L^\circ(\Delta)$ such that p is minimal under smoothening. Let $p = \langle E_p, \leqslant_p, l_p \rangle$ and $E_p = \{\, e_1, \ldots, e_n \,\}$. For each $e_i \in E_p$ we construct a sequence σ^i such that e_i is only preceded in σ^i by those events in p that precede e_i under $\leqslant_p$. Lemma 3.9 guarantees that $\sigma^i \in T(\Delta)$. Repeating this process for each $e_i \in E_p$ thus results in a set of event traces $\sigma^1, \ldots, \sigma^n$ with $\overline{\sigma_i} = E_p$, and thus $\sigma^i \sim \sigma^j$ for all $0 < i, j \leqslant n$. From Definition 3.11 and the construction of σ^i it follows immediately for each

$e_i \in E_p$ that $\mathsf{men}([\sigma^i]_\sim, e_i)$ consists of the set of events that precede e_i under $\leqslant_p$. Since $\sigma^i \in T(\Delta)$ this implies that $p \in L^\bullet(\Delta)$. □

3.19. THEOREM. $\forall\, \Delta, \Delta' \in \mathsf{DES} : L^\bullet(\Delta) = L^\bullet(\Delta') \Longleftrightarrow T(\Delta) = T(\Delta')$.

PROOF. '$\Rightarrow$': It follows from Lemma 3.9 that every linearization of an lposet of Δ and Δ' is an event trace. In a similar way as in the proof of Theorem 3.10 it can be proven that $L^\bullet(\Delta) = L^\bullet(\Delta') \Rightarrow T(\Delta) = T(\Delta')$.

'$\Leftarrow$': If $T(\Delta) = T(\Delta')$ it means that the minimal enablings of all events are identical, and consequently that all operational lposets are identical. □

3.2.3 Remainder

The remainder of a dual event structure after the execution of a sequence of events is defined analogously as for extended bundle event structures.

3.20. DEFINITION. (*Remainder of a dual event structure*)

$\Delta' = (E', \rightsquigarrow', \mapsto', l')$ is a *remainder* of Δ after $\sigma \in T(\Delta)$, denoted $\Delta' = \Delta[\sigma]$, iff

- $E' = E \setminus \overline{\sigma}$
- $\rightsquigarrow' = \rightsquigarrow \cap (E' \times E')$
- $\mapsto' = (\mapsto \setminus \{ (X, e) \mid X \mapsto e \;\wedge\; X \cap \overline{\sigma} \neq \varnothing \}) \cup \{ (\varnothing, e) \mid \exists\, e' \in \overline{\sigma}, e \in E' : e \rightsquigarrow e' \}$
- $l' = l \restriction E'$.

□

Each bundle $X \mapsto e$ such that $X \cap \overline{\sigma} \neq \varnothing$ is removed, because the enabling condition that this bundle poses, namely that some event in X should have happened before e can happen, is now fulfilled. This is according to the principle that the first possible cause of an event e that happens will cause e.

3.21. EXAMPLE. Let dual event structure Δ be depicted in Figure 3.5(a). Figure 3.5(b) depicts $\Delta[e_a]$ and Figure 3.5(c) depicts $\Delta[e_b]$. Remark that e_c is enabled once e_a occurs. Similarly, e_c and e_d are enabled once e_b occurs. □

As a prerequisite for the next theorem we need to lift the notion of prefix on lposets to families of lposets. This is done in the following way:

3.22. DEFINITION. For $\mathcal{P}$ and $\mathcal{Q}$ families of lposets let

$$\mathcal{P} \text{ is a prefix of } \mathcal{Q} \Leftrightarrow (\forall\, p \in \mathcal{P} : (\exists\, q \in \mathcal{Q} : p \text{ is a prefix of } q)) \quad .$$

□

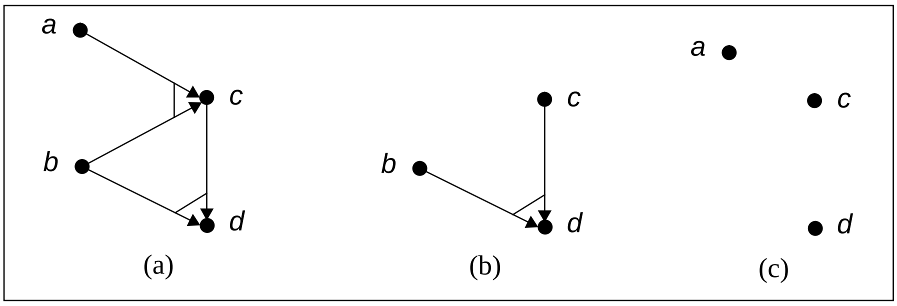


Figure 3.5: Remainders of dual event structures.

Phrased in words, $\mathcal{P}$ is considered to be a prefix of $\mathcal{Q}$ iff for each lposet $p \in \mathcal{P}$ there exists an lposet $q \in \mathcal{Q}$ such that p is a prefix (in the sense of lposets) of q. Note that we do not require the reverse, i.e., that for each $q \in \mathcal{Q}$ there exists a $p \in \mathcal{P}$ such that p is a prefix of q. So, $\mathcal{Q}$ may contain lposets that have no prefix in $\mathcal{P}$.

We now have the following correctness result for the remainder of Δ. (This seems identical to Theorem 2.30 but it should be reminded that $L^\circ(\overline{\sigma})$ is now a set of lposets rather than a single lposet, and that the notion of prefix is generalized to sets of lposets.)

3.23. THEOREM. *Correctness of remainder*

For $\sigma \in T(\Delta)$ and σ' a sequence of events:

1. $\sigma' \in T(\Delta[\sigma]) \iff \sigma\,\sigma' \in T(\Delta)$
2. $\sigma' \in T(\Delta[\sigma]) \ \Rightarrow\ L^\circ(\overline{\sigma})$ is a prefix of $L^\circ(\overline{\sigma\,\sigma'})$.

PROOF. Since the definitions of event trace and remainder for a dual event structure are identical to that of an extended bundle event structure, the first theorem follows directly from Theorem 2.30. We prove that $L^\circ(\overline{\sigma})$ is a prefix of $L^\circ(\overline{\sigma\,\sigma'})$ given that $\sigma \in T(\Delta)$ and $\sigma' \in T(\Delta[\sigma])$ with $\overline{\sigma} \cap \overline{\sigma}' = \varnothing$. Let $\Delta[\sigma] = (E', \rightsquigarrow', \mapsto', l')$. Let $p = \langle E_p, \leqslant_p, l_p\rangle$ be an lposet in $L^\circ(\overline{\sigma})$ and $r = \langle E_r, \leqslant_r, l_r\rangle$ be an lposet in $L^\circ(\overline{\sigma}')$ of $\Delta[\sigma]$. We prove that there exists an lposet $q \in L^\circ(\overline{\sigma\,\sigma'})$ such that p is a prefix of q by constructing an lposet $q = \langle E_q, \leqslant_q, l_q\rangle$ and then show that (i) p is a prefix of q and that (ii) $q \in L^\circ(\overline{\sigma\,\sigma'})$. Let $E_q = E_p \cup E_r$, $l_q = l_p \cup l_r$ and $\leqslant_q = (\leqslant_p\ \cup\ \leqslant_r\ \cup\ <)^*$ where $<$ is an acyclic relation satisfying:

1. $\forall\, e \in E_p, e' \in E_r : e \rightsquigarrow e' \ \Rightarrow\ e < e'$
2. $\forall\, X \subseteq E, e' \in E_r : X \mapsto e' \ \wedge\ \neg(X \mapsto' e') \ \Rightarrow\ (\exists\, e \in X \cap E_p : e < e')$

where the constraints on the bundle assignment functions are respected. The fact that p is a prefix of q follows immediately from the fact that no event in $E_q \setminus E_p$, i.e., E_r, precedes under $\leqslant_q$ an event in E_p. The proof that $q \in L^\circ(\overline{\sigma\,\sigma'})$ is rather straightforward (but elaborate) by checking the conditions of Definition 3.5 and is omitted here. □

A few remarks are in order. To define the notion of remainder for dual event structures we have adopted the principle that in case of disjunctive causality the first possible cause of e that happens will actually cause e. For instance, in Figure 3.5(b) the possibility that e_c causally depends on e_b is lost, since bundle $\{e_a, e_b\} \mapsto e_c$ is satisfied as soon as e_a has occurred. When considering bundles as enablings this is a defensible decision: once an event in a bundle occurs, the event pointed to by this bundle is enabled, and so can occur. The same choice is made by Gunawardena in his timed $\{\mathsf{AND}, \mathsf{OR}\}$ automata when relating temporal and causal ordering in case of OR causality [61, 62]. Another justifiable perspective is, referring again to Figure 3.5(b), that when e_a has occurred there is still a possibility for e_c to be causally dependent on e_b (if e_b happens). This requires a more involved notion of remainder, since we need to keep track of events that have occurred. The study of this alternative notion of remainder is left for further study.

For dual event structures we have, according to Theorem 3.10, that event traces are not sufficiently expressive as an underlying semantical model, unlike extended bundle event structures. It would therefore be interesting to consider remainders after lposets, rather than event traces, like we did in this section.

3.2.4 Transformation rules

This section presents some transformation rules for dual event structures that can be used to transform Δ into Δ' such that $L^\circ(\Delta) = L^\circ(\Delta')$. We take the same approach as in Section 2.3.4. Each rule is presented in pictorial form and in formal terms. To illustrate how correctness proofs of rules are conducted we provide the proofs of some non-trivial rules.

3.24. Theorem. $(E, \rightsquigarrow, \mapsto, l)$ is lposet equivalent with

1. $(E, \rightsquigarrow, \mapsto \setminus \{(X, e)\}, l)$ if $X \mapsto e \ \wedge\ Y \mapsto e \ \wedge\ X \mapsto Y$.
2. $(E, \rightsquigarrow \setminus \{(e, e')\}, \mapsto, l)$ if $X \rightsquigarrow e' \ \wedge\ e' \rightsquigarrow X \ \wedge\ X \mapsto e \ \wedge\ e \rightsquigarrow e'$.
3. $(E, \rightsquigarrow \setminus \{(e', e)\}, \mapsto, l)$ if $e' \rightsquigarrow X \ \wedge\ X \mapsto e \ \wedge\ e' \rightsquigarrow e$.
4. $(E, \rightsquigarrow, (\mapsto \setminus \{(X, e)\}) \cup \{(X \setminus e, e)\}, l)$ if $e \in X \ \wedge\ X \mapsto e$.
5. $(E, \rightsquigarrow, (\mapsto \setminus \{(X, e')\}) \cup \{(X \setminus e, e')\}, l)$ if $e' \rightsquigarrow e \ \wedge\ e \in X \ \wedge\ X \mapsto e'$.
6. $(E, \rightsquigarrow, (\mapsto \setminus \{(X, e')\}) \cup \{(X \setminus e, e')\}, l)$ if $X \mapsto e' \ \wedge\ e \in X \ \wedge\ \varnothing \mapsto e$.
7. $(E, \rightsquigarrow, \mapsto \setminus \{(X, e)\}, l)$ if $\varnothing \mapsto e \ \wedge\ X \mapsto e$.
8. $(E, \rightsquigarrow \setminus \{(e', e)\}, \mapsto, l)$ if $\varnothing \mapsto e \ \wedge\ e' \rightsquigarrow e$.
9. $(E, \rightsquigarrow \setminus \{(e, e')\}, \mapsto, l)$ if $\varnothing \mapsto e \ \wedge\ e \rightsquigarrow e'$.

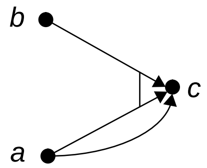
Proof. We only provide the proofs for the first two rules. The proofs for the other rules are similar and omitted. For each rule let Δ_l and Δ_r denote the left-hand and right-hand dual event structure, respectively.

1. The only difference between these two dual event structures is that Δ_r does require e to be preceded in any lposet by some event in X. The proof is by contradiction. Suppose Δ_l has an lposet p that contains e but where e is not preceded (under $\leqslant_p$) by an event in X. By definition, e is preceded by event e', say, in Y. So, $e' \leqslant_p e$. For e' we have that $X \mapsto e'$ and so there should be some event e'' in X with $e'' \leqslant_p e'$. But then, by transitivity of $\leqslant_p$ we have $e'' \leqslant e$. Contradiction. So, both dual event structures have the same set of lposets.

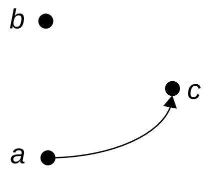
2. The only difference between these two dual event structures is that Δ_l does not allow an lposet in which e' is preceded by e. The proof is by contradiction. Suppose Δ_r has an lposet p for which e' is preceded by e, i.e., $e \leqslant_p e'$. If $e \in E_p$ then there is some event e'', say, in X such that $e'' \leqslant_p e$. But, since $e'' \# e'$ this means that both e'' and e' occur in a system run. Contradiction. So, both dual event structures have the same set of lposets.

$\square$

The transformation rules of Theorem 3.24 are pictorially represented in Figure 3.6. The first three rules and last three rules do also hold for extended bundle event structures, see Figure 2.4. Remark that there is no transformation rule that allows for the removal of sub-bundles, like we had for extended bundle event structures. For instance,



cannot be simplified because removal of $\{ e_a \} \mapsto e_c$ would lead to a dual event structure in which the lposet $\boxed{\begin{smallmatrix} e_a & \searrow & \\ e_b & \rightarrow & e_c \end{smallmatrix}}$ becomes impossible. The same applies for the removal of $\{ e_a, e_b \} \mapsto e_c$. It is interesting to observe that for the operational characterization of lposets we have that the above dual event structure can be simplified to



since these two dual event structures are event trace equivalent.

Impossible events in extended bundle event structures have the pleasant property that they can always be eliminated while preserving the underlying lposet semantics. Fortunately, such a result also holds for dual event structures as shown below. The rules of Theorem 3.24 facilitate the separation of all impossible events in a dual event structure. The following theorem shows that all the isolated impossible events can be safely eliminated.

3.25. THEOREM. *Removal of impossible events*

Let $\Delta = (E, \rightsquigarrow, \mapsto, l)$ with $e \notin E$, and let $a \in \mathcal{A}$. Then Δ is lposet equivalent with $(E \cup \{ e \}, \rightsquigarrow, \mapsto \cup \{ (\varnothing, e) \}, l \cup \{ (e, a) \})$.

PROOF. Straightforward and omitted. $\square$

The following theorem shows that impossible events do not extend the expressiveness of dual event structures. This is opposed to flow event structures where self-conflicting events, which are also impossible, cannot always be removed without affecting the underlying semantics.

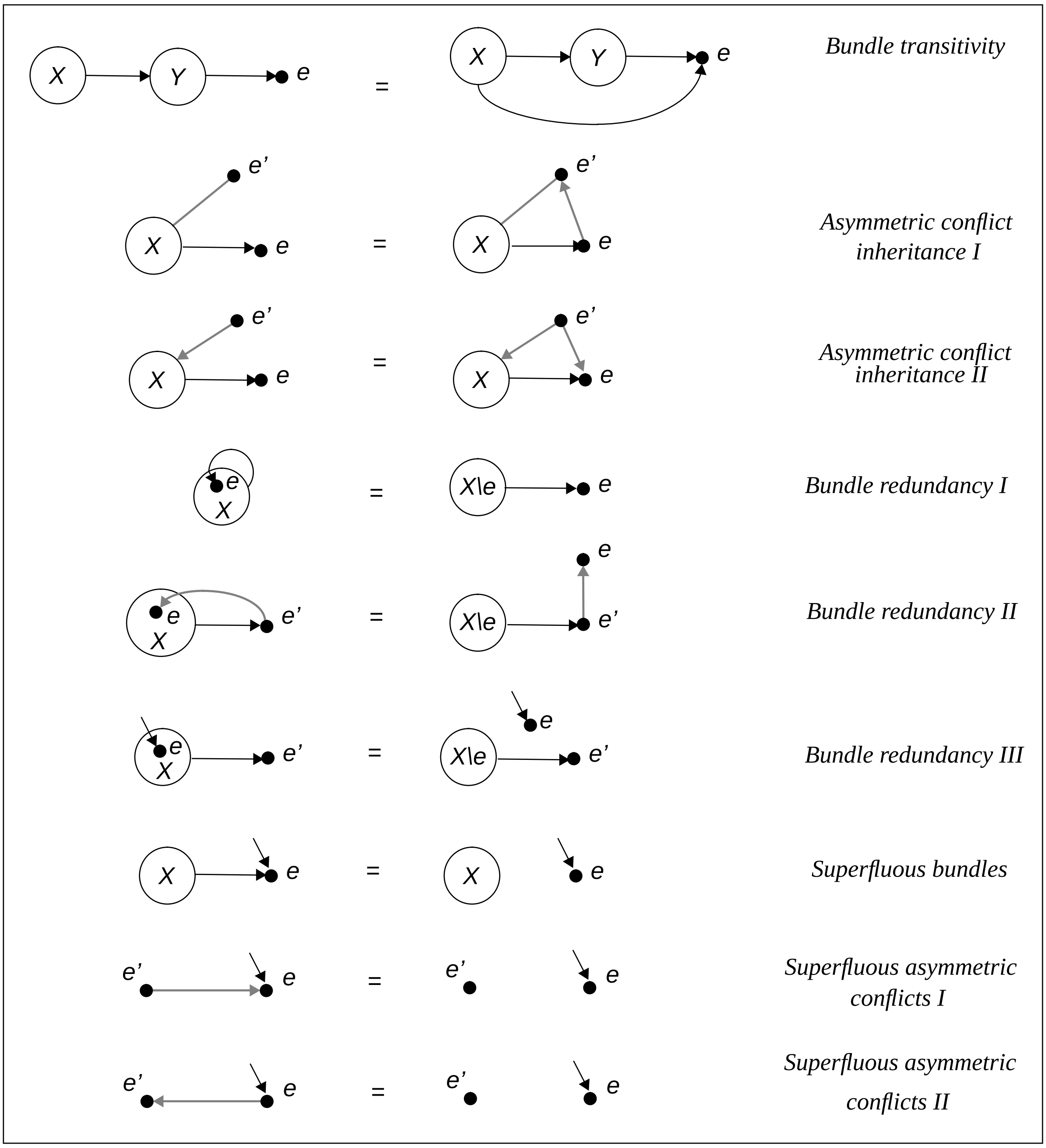


Figure 3.6: Transformation rules for dual event structures.

3.26. THEOREM. $\Delta \in \mathsf{DES}$ can be transformed into $\Delta' = (E', \rightsquigarrow', \mapsto', l')$ such that $L^\circ(\Delta) = L^\circ(\Delta')$ and $(\forall\, (X, e) \in \mapsto' : X \neq \varnothing)$.

PROOF. Analogous to [89, Theorem 5.5.4]. □

3.27. EXAMPLE. The transformation rules of this section can, for instance, be used to eliminate cyclic bundles such as $X \mapsto \ldots \mapsto X$. Consider, for example, the dual event structure depicted in Figure 3.7(a). By applying the rule of bundle transitivity Figure 3.7(b) is obtained which can be proven to be lposet equivalent with Figure 3.7(c) by applying the rule bundle redundancy I. By the rule superfluous bundles we obtain Figure 3.7(d). Finally, using Theorem 3.25 this dual event structure is proven to be lposet equivalent with the empty dual event structure. □

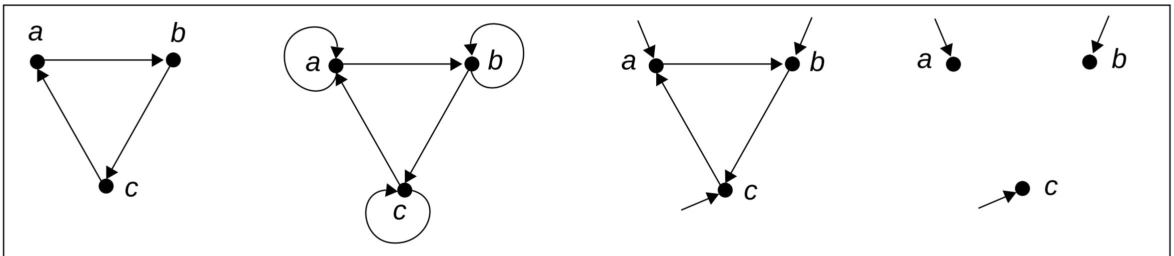


Figure 3.7: Transformations of a cyclic event structure.

We conclude this section by stating that redundant bundles can always be simplified, i.e., for $X \mapsto e$ impossible events in X can be removed from X (bundle redundancy III), events in X in conflict with e can be removed from X (bundle redundancy II), and in case $e \in X$, e can also be removed from X (bundle redundancy I). To our opinion this proves that the transformation rules, although not complete, are useful to eliminate undesired phenomena from dual event structures.

3.2.5 Expressiveness of dual event structures

By definition dual event structures are strictly more expressive than extended bundle event structures, and thus than prime event structures. This also holds at the level of sets of configurations, since, for example, there does not exist an extended bundle event structure with the same set of configurations as the dual event structure of Figure 3.4(a).

On the level of sets of configurations extended bundle event structures are incomparable with stable and flow event structures. That is, there is an extended bundle event structure with a set of configurations that cannot be generated by any flow or stable event structure, and vice versa [89, Chapter 6]. This section shows that on the level of sets of configurations dual event structures are strictly more expressive than stable event structures and flow event structures.

We provide a recipe for transforming a (labelled) stable event structure $\mathcal{S}$ into a corresponding dual event structure $\Delta(\mathcal{S})$. This recipe is proven to be correct on the level of event traces and indicates that dual event structures are *at least as* expressive as stable event structures on the level of event traces, and thus on the level of sets of configurations. By providing a dual event structure for which it is impossible to construct a corresponding stable event structure with the same set of configurations it is shown that dual event structures are *strictly more* expressive than stable event structures.

(Labelled) stable event structure $\mathcal{S} = (E, \#, \vdash, l)$ is transformed into a dual event structure $\Delta(\mathcal{S})$ in the following way. The symmetric conflict relation between events e and e' is turned into the equivalent asymmetric conflicts $e \rightsquigarrow e'$ and $e' \rightsquigarrow e$. As a result $e \rightsquigarrow e'$ in $\Delta(\mathcal{S})$ iff $e' \rightsquigarrow e$ in $\Delta(\mathcal{S})$. The definition of the bundle relation $\mapsto$ is somewhat more complex. Consider event e with enablings $X_1 \vdash e$ and $X_2 \vdash e$ in $\mathcal{S}$. Thus, if e happens either all events in X_1 or in X_2 have happened. We now obtain $\mapsto$ by taking all pairs of events (e_1, e_2) with $e_1 \in X_1$ and $e_2 \in X_2$ and introduce a bundle $\{ e_1, e_2 \} \mapsto e$ for all such pairs. Using this construction it is ensured that enabling $X_i \vdash e$ (for i=1, 2) is satisfied iff all bundles in $\Delta(\mathcal{S})$ are satisfied (see proof of Theorem 3.32). Generalizing this approach to the case of k bundles ($k \geqslant 0$) results in the following construction.

3.28. DEFINITION. (*From stable to dual event structures*)

Let $\mathcal{S} = (E, \#, \vdash, l)$ be a (labelled) stable event structure. $\Delta(\mathcal{S}) \triangleq (E, \rightsquigarrow, \mapsto, l)$ with

- $e \rightsquigarrow e' \;\wedge\; e' \rightsquigarrow e \Leftrightarrow e \# e'$
- $\{ e_1, \ldots, e_k \} \mapsto e \Leftrightarrow \forall\, 0 < i \leqslant k : e_i \in X_i \;\wedge\; X_i \vdash e$,
  where k is the number of (non-empty) enablings of e in $\mathcal{S}$.

$\square$

3.29. EXAMPLE. Let $\mathcal{S}$ be a stable event structure with set of events $\{ e_a, e_b, e_x, e_y, e_f \}$, $e_b \# e_y$, $\{ e_a, e_b \} \vdash e_f$ and $\{ e_x, e_y \} \vdash e_f$ and the other events having empty enablings, see Figure 3.8(a). The corresponding dual event structure $\Delta(\mathcal{S})$ is depicted in Figure 3.8(b). Conform Definition 3.28 this dual event structure has bundles $\{ e_a, e_x \} \mapsto e_f$, $\{ e_b, e_x \} \mapsto e_f$, $\{ e_a, e_y \} \mapsto e_f$ and $\{ e_b, e_y \} \mapsto e_f$. $\square$

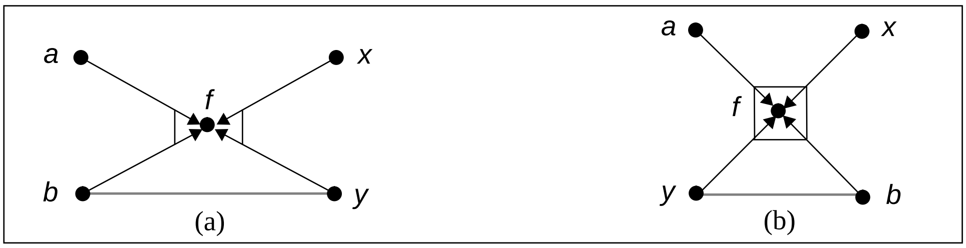


Figure 3.8: A stable event structure (a) and its corresponding dual event structure (b).

In order to prove the correctness of Definition 3.28 we need to define the lposets of a stable event structure. This can be done in the following way.

3.30. DEFINITION. (*Lposets of a stable event structure*)

The lposets of labelled stable event structure $\mathcal{S}$, denoted $L(\mathcal{S})$, is the family of lposets $\langle C, \prec_C^*, l \upharpoonright C \rangle$ where $\prec_C \subseteq C \times C$ is a minimal acyclic relation and $C \subseteq E$ is conflict-free, satisfying for all $e \in C$:

$$\exists\, X \subseteq C : X \vdash e \;\wedge\; (\forall\, e' \in X : e' \prec_C e) \enspace .$$

□

It directly follows that each $p \in L(\mathcal{S})$ is an lposet. Moreover,

3.31. LEMMA. For all stable event structures $\mathcal{S}$, $\mathcal{S}'$: $L(\mathcal{S}) = L(\mathcal{S}') \iff T(\mathcal{S}) = T(\mathcal{S}')$.

PROOF. Straightforward and omitted. □

We now have the following correctness result:

3.32. THEOREM. For all stable event structures $\mathcal{S}$: $L(\mathcal{S}) = L^{\circ}(\Delta(\mathcal{S}))$.

PROOF. '$\subseteq$': Let $p = \langle E_p, \leqslant_p, l_p \rangle$ be an lposet in $L(\mathcal{S})$. Suppose $X_1 \vdash e, \ldots, X_k \vdash e$ in $\mathcal{S}$. Since $p \in L(\mathcal{S})$ it follows from Definition 3.30 that there exists an X_m, for $0 < m \leqslant k$, such that $X_m \subseteq E_p$ and $(\forall\, e' \in X_m : e' \leqslant_p e)$. According to Definition 3.28 all bundles in $\Delta(\mathcal{S})$ are of the form $\{ e'_1, \ldots, e'_k \} \mapsto e$ with $e'_j \in X_j$, for $0 < j \leqslant k$. Since all events in X_m precede (under $\leqslant_p$) event e we take for each bundle Y pointing to e the event in X_m, i.e., $F_e(Y) = e'_m$. In this way each bundle in $\Delta(\mathcal{S})$ pointing to e is satisfied by precisely one event. This implies that p satisfies the constraints of Definition 3.5 and is an (intensional) lposet of $\Delta(\mathcal{S})$.

'$\supseteq$': Let $p = \langle E_p, \leqslant_p, l_p \rangle$ be an lposet in $L^{\circ}(\Delta(\mathcal{S}))$. The proof that $p \in L(\mathcal{S})$ is by contradiction. Suppose $p \notin L(\mathcal{S})$. This can only be because no enabling X_j of e in $\mathcal{S}$ satisfies $(\forall\, e_j \in X_j : e_j \leqslant_p e)$. Assume $X_1 \vdash e, \ldots, X_k \vdash e$ in $\mathcal{S}$. Since $p \notin L(\mathcal{S})$ it means that for all j we have $(\exists\, e'_j \in X_j : e'_j \not\leqslant_p e)$. From Definition 3.28 it now follows that for bundle $\{ e'_1, \ldots, e'_k \} \mapsto e$ there is no event preceding e under $\leqslant_p$. But then there is no bundle assignment function for e satisfying the constraints of Definition 3.5, so $p \notin L^{\circ}(\Delta(\mathcal{S}))$. Contradiction. □

The following example shows that dual event structures are strictly more expressive than stable event structures.

3.33. EXAMPLE. Consider the dual event structure with events e_a, e_b, and e_c with $\{ e_a, e_b \} \mapsto e_c$ (i.e., Winskel's switch [155]). This event structure has the following set of configurations $\varnothing, \{ e_a \}, \{ e_b \}$, $\{ e_a, e_c \}, \{ e_b, e_c \}, \{ e_a, e_b \}$, and $\{ e_a, e_b, e_c \}$. In a corresponding stable event structure there should be an enabling $\{ e_a \} \vdash e_c$ and $\{ e_b \} \vdash e_c$, but due to the stability constraint there should be a conflict between e_a and e_b, making the maximal configuration $\{ e_a, e_b, e_c \}$ impossible. So, it is impossible to construct a stable event structure with this set of configurations. □

So, on the level of sets of configurations dual event structures are strictly more expressive than stable event structures, and since stable event structures are strictly more expressive than flow event structures it follows that dual event structures are more expressive than flow event structures. The realm of event structures indicating the hierarchy at the level of sets of configurations is presented in Figure 3.9. (Expressiveness increases when going from left to right.) A similar hierarchy of event structure models has recently been published by Van Glabbeek & Plotkin [52]. It is an interesting result that dual event structures are an 'upper bound' of extended bundle and stable event structures. This is not to say that is the least upper bound; it would be interesting to consider stable event structures equipped with an asymmetric conflict relation for this purpose.

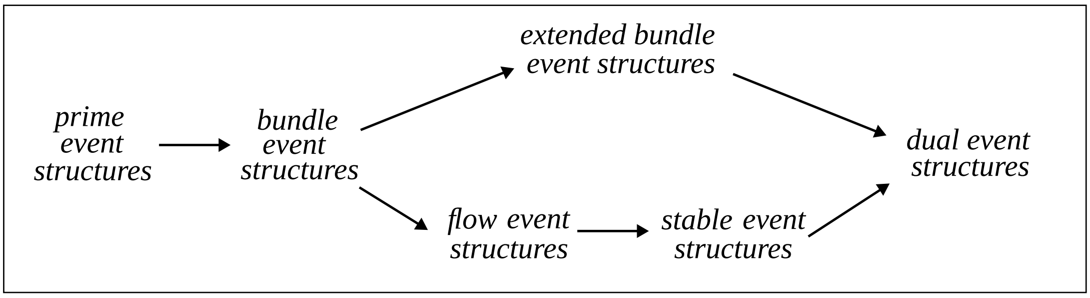


Figure 3.9: The realm of event structures.

The connection between dual and stable event structures has other important consequences. From Rensink [127] it is known that prime, bundle, flow and extended bundle event structures do respect a global relation $<_{\mathcal{P}} \subseteq E_{\mathcal{P}} \times E_{\mathcal{P}}$ (if it exists) on the level of a family $\mathcal{P}$ of lposets, called the *causal flow* relation. The existence of a causal flow relation is based on the intuition that there is a *fixed* cause-and-effect relation between the events. We recall from [127]:

3.34. DEFINITION. (*Causal flow relation*)

For $\mathcal{P}$ a family of lposets a binary relation $<_{\mathcal{P}}$ is a *(causal) flow* relation on $\mathcal{P}$ if it is irreflexive and for all $p \in \mathcal{P}$ and $e, e' \in E_p$:

$$e \leqslant_p e' \iff (e, e') \in (<_{\mathcal{P}} \cap (E_p \times E_p))^* \quad .$$

□

$\mathcal{P}$ is said to *reflect causal flow* if there exists a causal flow relation on $\mathcal{P}$. Events related under $<_{\mathcal{P}}$ should in every possible run of the system be causally related according to the causal flow relation—if $e, e' \in E_p$ for some $p \in \mathcal{P}$ such that $e <_{\mathcal{P}} e'$ then also $e \leqslant_p e'$. In addition, the ordering relations of the posets should be backed up by chains of causal relations: if $e \leqslant_p e'$ then $(e, e') \in (<_{\mathcal{P}} \cap (E_p \times E_p))^*$.

Stable event structures do *not* respect causal flow. The following example is taken from [127, Chapter 2]. Consider stable event structure $\mathcal{S}$ with events $\{ e_a, e_b, e_c, e_d \}$, $e_a \# e_b$ and enablings $\varnothing \vdash e_a, \varnothing \vdash e_b, \{ e_a \} \vdash e_c, \{ e_a, e_c \} \vdash e_d, \{ e_b \} \vdash e_d$ and $\{ e_b, e_d \} \vdash e_c$. The corresponding dual event structure $\Delta(\mathcal{S})$ consists of $e_a \rightsquigarrow e_b$, $e_b \rightsquigarrow e_a$ and bundles $\{ e_a, e_b \} \mapsto e_c$, $\{ e_a, e_d \} \mapsto e_c$, $\{ e_a, e_b \} \mapsto e_d$, and $\{ e_b, e_c \} \mapsto e_d$, see Figure 3.10. Two (operational) lposets of this dual event structure are $\boxed{e_b \rightarrow e_d \rightarrow e_c}$ and $\boxed{e_a \rightarrow e_c \rightarrow e_d}$. Here we see that e_d is enabled by e_c in one run of the system, while in another run it is just the other way around! This means that, in general, for dual event structures, there is no fixed cause-and-effect relation between events.

So, relaxing the stability constraint in extended bundle event structures, a model that respects causal flow, results in dual event structures, a model that does not respect causal flow.

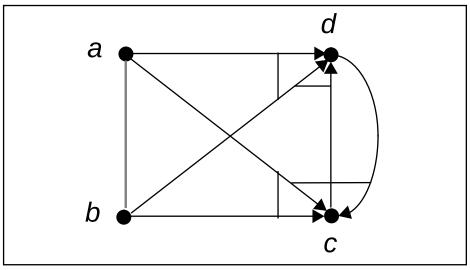


Figure 3.10: A dual event structure that does not respect causal flow.

3.3 Interleaving

Inspired by the work of Ferreira Pires *et al.* [46, 145] we equip in this section dual event structures with a symmetric *interleaving relation* between events. Such a relation can be used to model that events can happen in any order but are not independent, i.e., they may not occur at the same time. Such scenarios typically appear in mutual exclusion situations.[3]

Interleaving of events can be represented using the basic ingredients of event structures by explicitly modelling each possible interleaving. For instance, three events e_a, e_b, and e_c that are mutually interleaved can be modelled as depicted in Figure 3.11(a). The benefits of such a representation are that no extensions of the basic machinery of event structures are needed (conflict and causality suffice), and that the different causal orderings between events are explicitly shown. The main drawback of this representation is that it leads to an explosion of

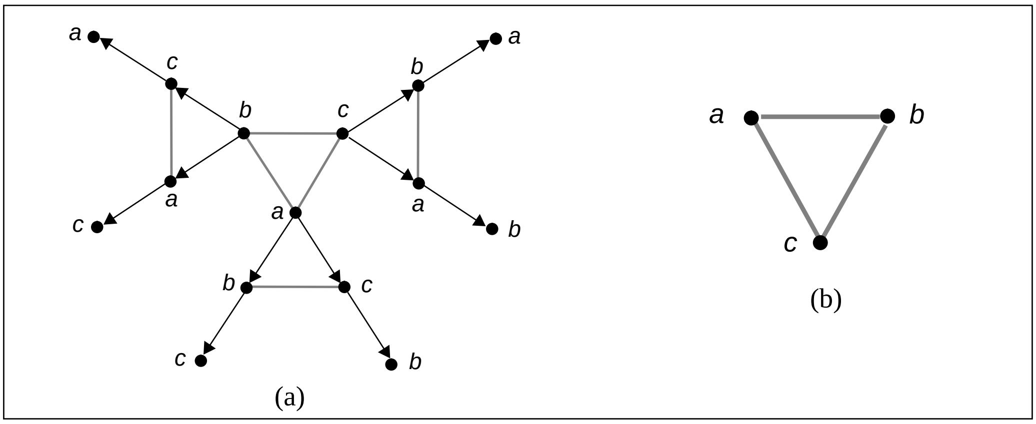


Figure 3.11: Modelling the interleaving of events.

the number of events.[4] In addition, the symmetric nature of interleaving—if e is interleaved with e', then e' is interleaved with e—is no longer explicitly represented as a symmetric relationship.

[3] We like to point out that in a process algebraic framework, which is not present in [46, 145], the interleaving of (observable) events of processes P and Q, say, can always be established by synchronizing P and Q with a third process, R say, that forces this interleaving explicitly.

[4] More precisely, if k_n denotes the number of copies of an event in case of n interleaved events it follows that $k_1 = 1$ and $k_{n+1} = n \cdot k_n + 1$, for $n \geqslant 0$.

We, therefore, propose a different route and introduce a (symmetric) interleaving relation, denoted $\rightleftharpoons$, between events. The interpretation of $e \rightleftharpoons e'$ is that e and e' are interleaved, e being caused by e' when e occurs before e', or vice versa. Using this relation we obtain for the above example the (concise) event structure as depicted in Figure 3.11(b) where the grey line between events means that the connected events are interleaved. $\rightleftharpoons$ strongly resembles the global dependency relation introduced in [159] and [151]; the main difference is that $\rightleftharpoons$ concerns events rather than actions.

3.35. Definition. (*Extended dual event structure*)

An *extended dual event structure* Θ is a tuple $\langle \Delta, \rightleftharpoons \rangle$ with

- Δ, a dual event structure $(E, \rightsquigarrow, \mapsto, l)$
- $\rightleftharpoons \subseteq E \times E$, the (irreflexive and symmetric) *interleaving* relation.

$\Box$

Since several mechanisms to model impossible events (either by $\varnothing \mapsto e$ or $\{ e \} \mapsto e$) exist we do not allow $\rightleftharpoons$ to be reflexive. $e \rightleftharpoons e'$ is represented by a thick solid grey line between e and e'. EDES denotes the class of extended dual event structures and we use Θ, possibly subscripted and/or primed, for elements of EDES.

The notions of event trace and configurations are identical to dual event structures. The lposets of an extended dual event structure are defined in an intensional way as follows:

3.36. Definition. (*Lposets of an extended dual event structure*)

The lposets of Θ, denoted $L(\Theta)$, is the family of lposets $\langle C, \prec_C^*, l \restriction C \rangle$ where $\prec_C \subseteq C \times C$ is an acyclic relation and $C \subseteq E$ is conflict-free, satisfying for all $e \in C$:

1. $\forall\, e' \in C : e' \rightsquigarrow e \;\Rightarrow\; e' \prec_C e$, and
2. $\forall\, e' \in C : e' \rightleftharpoons e \;\Rightarrow\; (e' \prec_C e \;\vee\; e \prec_C e')$
3. $\exists\, F_e : \{ X \mid X \mapsto e \} \longrightarrow \{ e' \mid e' \prec_C e \}$ such that
   (a) $\{ e' \mid e' \prec_C e \} \subseteq (\{ F_e(X) \mid X \in \mathit{dom}(F_e) \} \cup \{ e' \mid e' \rightsquigarrow e \;\vee\; e' \rightleftharpoons e \})$, and
   (b) $\forall\, X \in \mathit{dom}(F_e) : F_e(X) \in X$.

$\Box$

The difference with Definition 3.5 is the second constraint that takes care of interleaved events. In addition, constraint 3.(a) is adapted by the incorporation of interleaved events. It can be verified along the same lines as for dual event structures that for each event $e \in C$ a bundle assignment function F_e exists satisfying constraints 3.(a) and 3.(b). This is left to the diligent reader.

The following result indicates that all linearizations of an lposet of Θ that respect the ordering of the lposet are event traces of Θ.

3.37. LEMMA. $\forall\, \sigma \in E^* : (\exists\, p \in L(\Theta) : E_p = \overline{\sigma} \;\wedge\; \leqslant_p \subseteq <_\sigma^*) \iff \sigma \in T(\Theta)$.

PROOF. Similar to the proof of Lemma 3.9. $\Box$

3.38. LEMMA. $\forall\, \Theta, \Theta' \in \mathsf{EDES} : L(\Theta) = L(\Theta') \;\Rightarrow\; T(\Theta) = T(\Theta')$.

PROOF. Similar to the proof of Theorem 3.10 $\Box$

Notice that the reverse implication does not hold. A counterexample is provided by the extended dual event structures

(a) (b)

They have the same set of event traces, but (a) has one maximal lposet $\boxed{\begin{smallmatrix} e_a \\ e_b \end{smallmatrix}}$ whereas (b) has maximal lposets $\boxed{e_a \rightarrow e_b}$ and $\boxed{e_b \rightarrow e_a}$. This example also shows that it makes not much sense to deduce lposets for extended dual event structures in an operational way, i.e., from event traces, since interleaving and independence of events can never be distinguished.

3.39. DEFINITION. (*Remainder of an extended dual event structure*)

$\Theta' = (\Delta', \rightleftharpoons')$ is a *remainder* of Θ after $\sigma \in T(\Theta)$, denoted $\Theta' = \Theta[\sigma]$, iff $\Delta' = \Delta[\sigma] = (E', \rightsquigarrow', \mapsto', l')$ and $\rightleftharpoons' = \rightleftharpoons \cap\, (E' \times E')$. $\Box$

We have the following correctness result on remainders of extended dual event structures.

3.40. THEOREM. *Correctness of remainder*

For $\sigma \in T(\Theta)$ and σ' a sequence of events:

1. $\sigma' \in T(\Theta[\sigma]) \iff \sigma\, \sigma' \in T(\Theta)$
2. $\sigma' \in T(\Theta[\sigma]) \;\Rightarrow\; L(\overline{\sigma})$ is a prefix of $L(\overline{\sigma\, \sigma'})$.

PROOF. Similar to the proof of Theorem 3.23. $\Box$

Figure 3.12 presents some transformation rules on extended dual event structures. The transformation rules of Figure 3.6 do also apply in this setting. The first rule facilitates the isolation of impossible events in presence of interleavings. The second and third rule provide a means to remove redundant interleavings.

3.41. THEOREM. $\langle (E, \rightsquigarrow, \mapsto, l), \rightleftharpoons \rangle$ is lposet equivalent with $\langle (E, \rightsquigarrow, \mapsto, l), \rightleftharpoons \setminus \{\, (e, e') \,\} \rangle$ if $e \rightleftharpoons e' \;\wedge\; (\varnothing \mapsto e' \;\vee\; e \rightsquigarrow e' \;\vee\; \{\, e \,\} \mapsto e')$.

PROOF. Straightforward and omitted. $\Box$

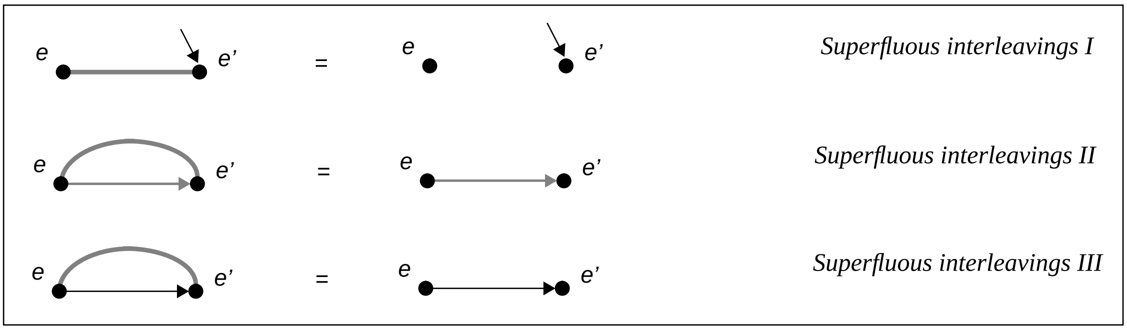


Figure 3.12: Transformation rules for eliminating interleavings.

3.4 Conclusions

In this chapter we have presented two qualitative extensions of extended bundle event structures. The main part of this chapter was devoted to a novel type of event structures, called *dual event structures*, which are obtained from extended bundle event structures by dropping the stability constraint. Dual event structures support *disjunctive causality*, i.e., they allow to express that an event is enabled once some causal predecessor has happened. The main consequences of dropping the stability constraint are that having the same set of lposets implies having the same set of event traces, but the reverse implication does no longer hold. This entails that event traces are not sufficiently expressive as an underlying semantical model for dual event structures—lposets can only be partly recovered from event traces; this is illustrated by presenting a novel recipe to generate lposets from event traces.

Dual event structures were shown to be strictly more expressive than stable event structures and, as a result, they do not respect a global causal flow relation between events (in contrast with prime, flow, bundle, and extended bundle event structures). This means that the causal dependencies between events in different runs of the system may be reversed. So, for dual event structures there does not need to be a fixed cause-and-effect relation between events.

In the same style as for extended bundle event structure transformation rules were presented that allow for the elimination of undesired phenomena in dual event structures, such as cyclic bundles, redundancy in bundles and impossible events. Due to the presence of disjunctive causality there is no rule for eliminating sub-bundles.

In the second part of this chapter we extended dual event structures with a symmetric (and irreflexive) *interleaving* relation between events. This relation provides an explicit mechanism to state that either e causes e' or e' causes e in a system run.

We consider the work presented in this chapter as a first investigation on the incorporation of disjunctive causality in event structures. Some issues deserve further attention. For instance, it would be interesting to see whether the recipe to generate lposets from event traces can be refined (without equipping traces with extra causality information) such that the intensional lposets can be better 'approximated', and to study other types of remainders, such as remainders after lposets, and remainders for which the principle that the first potential cause of an event that happens is the actual cause, is dropped.

4 A simple timing module

This chapter describes a simple timed variant of extended bundle event structures. We equip events and bundles with a time attribute. An event e with time t denotes that e is enabled from t time units on since the system is started, usually assumed to be time 0. t associated with bundle $X \mapsto e$ denotes that the time between the occurrence of an event in X and the appearance of e should be at least t time units. The result is a causality-based model allowing the specification of minimal time constraints. The timing extension is a conservative extension of the untimed causality-based model, is suitable for discrete and continuous time, and does not include notions to explicitly force the passage of time. A temporal process algebra is defined that includes a delay function which constrains the occurrence time of actions. The suitability of timed event structures for providing a compositional causality-based semantics to this algebra is studied.

4.1 Introduction

Extended bundle event structures allow for the modelling of systems by specifying their branching structure (conflicts) and causal ordering (bundles). This facilitates the specification of the relative ordering of events. The need for describing time constraints is well recognized. The specification of time-related properties is essential to describe, for instance, the time lapse between causally dependent events and to specify that a confirmation should be delivered within a certain time after issuing a request. In addition, the fact that events can only occur in a certain period of time cannot be described without information about time lapses.

This chapter considers a (simple) timed extension of extended bundle event structures. Section 4.2 introduces and justifies the timed causality-based model. Several notions that were defined for EBES are carried over to the timed case: timed event traces, timed remainders and the generation of (timed) lposets. The suitability of the resulting timed model for providing a causality-based semantics to a timed process algebra is investigated in Section 4.3. We prove that this semantics is a conservative extension of $\mathcal{E}[\![\ ]\!]$, the denotational semantics of PA. We investigate under which syntactical constraints the timed event structure model could be simplified. Finally, Section 4.4 draws some conclusions of this chapter.

4.2 Timed event structures

This section introduces our basic timed model, which we call timed event structures. Section 4.2.1 introduces the basic ideas and the notion of timed event structure. Section 4.2.2 deals with the notion of timed event trace, a generalization of event trace. A lattice of traces, in fact of equivalence classes of traces, is proposed in Section 4.2.3; this section is not essential for the rest of this chapter, and can be skipped if desired. Section 4.2.4 defines how to obtain lposets from timed event structures and relates this approach to the untimed case. The status of a timed event structure after the execution of a sequence of timed events is defined in Section 4.2.5. Finally, Section 4.2.6 presents some transformation rules.

4.2.1 What are timed event structures?

Let Time denote an arbitrary time domain with a total ordering relation $<$. We use t, possibly subscripted and/or primed, to range over Time.

The idea is to add time delays to event structures by associating time with bundles. Suppose we have an event e_b with a bundle $\{ e_a \} \mapsto e_b$ and we associate a time delay t to this bundle. The intuitive interpretation is that if e_a happens at a certain time, then e_b is enabled t time units later. That is, if e_a happens at time t_a, then e_b is enabled at time $t_a + t$. Event e_b does not have to happen immediately, so it may happen at any time from $t_a + t$ on. t is thus the *minimal* delay between e_a and e_b. In Chapter 7 we introduce a timed model which also supports the specification of time constraints that specify the last moment at which an event may happen.

The reason for not requiring what is often referred to as *maximal progress*, i.e., an event happens as soon as it is enabled, is that in general an event may be subject to interaction with the environment which may introduce further delays. Since we consider multi-party synchronization this also applies to events resulting from interaction between two components, unlike the case for binary synchronization (as in CCS) where synchronizations can be required to happen as soon as both (= all) participants are ready for it since no further interaction can take place.

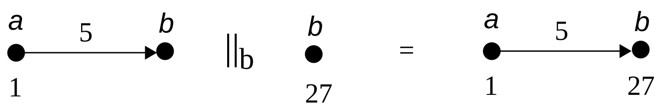
We assume function $\mathcal{T}$ to associate a value of Time, the time domain, to bundles. A bundle (X, e) with $\mathcal{T}((X, e)) = t$ is denoted by $X \overset{t}{\mapsto} e$.

Events may have several bundles pointing to them. Suppose we have an event e_c with bundles $\{ e_a \} \overset{t}{\mapsto} e_c$ and $\{ e_b \} \overset{t'}{\mapsto} e_c$. The interpretation that we choose for this construct is that an event can happen as soon as all timing constraints on it have been met. This means that a synchronization is enabled once all participants are ready to engage in it. For the above example, this means that if e_a happens at time t_a and e_b happens at time t_b, then e_c is enabled at time $max(t_a + t, t_b + t')$. So, in case $t_a + t < t_b + t'$ the component that performs e_a has to wait until the other component is ready for synchronization, after which it may continue (by performing e_c).

Summarizing, by associating time to bundles *relative* minimal time delays between events, or more precisely, between a set of events and an event, can be specified. We also would like to be able to specify time constraints for events that have no bundle pointing to them (i.e., the initial events). Such constraints specify the delay of an event with respect to the time at which the 'execution' began, normally assumed to be time 0. One might consider such constraints to be *absolute* time constraints.

There are basically two ways to support the specification of such time constraints: (i) associating time to events, or (ii) introducing a fictitious event, ω say, modelling the start of the system with a bundle pointing to the initial events equipped with the appropriate time delay. The second possibility, used in different contexts by, for instance, Murphy [106, 108] and Žic [158], has the main advantage that time is only associated to bundles, so—at first sight—keeping the model conceptually simple. The main drawback of this approach, however, is that definitions become more complex (event ω has to be treated quite differently from other 'normal' events; for instance, in the remainder of a timed event structure a new start event must be created in order to record the absolute time constraints of the remaining events) and proof obligations become more severe (e.g., one has to prove that bundles $X \mapsto e$ satisfy $e \neq \omega$ and $X = \{\, \omega \,\}$ or $\omega \notin X$, and that asymmetric conflicts $e \rightsquigarrow e'$ satisfy $e \neq \omega$ and $e' \neq \omega$).

In order not to complicate the theory, which could easily distract the reader from the essential points of the model, we consider possibility (i) of above, delays associated to events. We assume a function $\mathcal{D}$ that associates a value in Time to an event. Due to synchronization it does not suffice to only associate time values to initial events, but also non-initial events can be delayed. Consider, for example,



where the result specifies that if e_a occurs at t_a then e_b is enabled from $max(t_a{+}5, 27)$. The interpretation is that an event e with $\mathcal{D}(e) = t$ is enabled from t time units on since the start of the system.

Concluding, we propose the following notion of timed event structure.

4.1. DEFINITION. (*Timed event structure*)

A *timed* event structure is a triple $\langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ with

- $\mathcal{E}$, an (extended bundle) event structure $(E, \rightsquigarrow, \mapsto, l)$
- $\mathcal{D} : E \longrightarrow \mathsf{Time}$, the *event delay* function
- $\mathcal{T} : \mapsto \longrightarrow \mathsf{Time}$, the *bundle delay* function.

$\square$

For depicting timed event structures we use the following conventions. The time associated with a bundle or an event is a non-negative real number[1] and is depicted near to a bundle or an event, respectively. For convenience, we often omit delays equal to 0. We use Γ to denote

[1] This choice for Time allows for zero separation of time between causally dependent events. For instance $\{\, e_a \,\} \overset{0}{\mapsto} e_b$ allows e_a and e_b to occur at the same time instant. Other choices for Time could prevent this, if desired.

a timed event structure and EBES_T to denote the class of timed event structures. Recall that $\mathcal{E}$ is considered to have a finite number of events; infinite event structures are dealt with in Chapter 10.

4.2. EXAMPLE. Some example timed event structures are depicted in Figure 4.1. Figure 4.1(a) has bundles $\{e_a\} \overset{3}{\mapsto} e_c$, $\{e_b\} \overset{5}{\mapsto} e_c$, $\{e_b\} \overset{2}{\mapsto} e_d$, and a symmetric conflict between e_c and e_d. In Figure 4.1(b) we have $\mathcal{D}(e_a) = 2$, $\mathcal{D}(e_b) = 3$ and $\mathcal{D}(e_c) = 7$. Note that e_b is a non-initial event having a non-zero delay associated with it. □

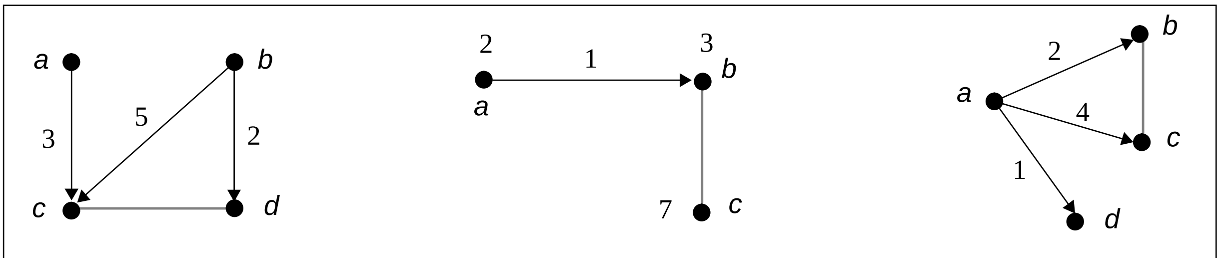


Figure 4.1: Some example timed event structures.

4.2.2 Timed event traces

We define the notion of *timed* event trace as a generalization of the notion of event trace. A timed event (e, t) denotes that e happened at time t.

4.3. NOTATION. For sequences of timed events $\sigma = (e_1, t_1) \ldots (e_n, t_n)$ let $\lfloor\sigma\rfloor$ denote the sequence of events of σ, i.e., $\lfloor\sigma\rfloor \triangleq e_1 \ldots e_n$ for $n \geqslant 1$ and $\lfloor\varepsilon\rfloor \triangleq \varepsilon$. Note that $\overline{\lfloor\sigma\rfloor}$ denotes the set of events in σ, while $\overline{\sigma}$ denotes the set of timed events in σ. □

Given a sequence σ of timed events $(e_1, t_1) \ldots (e_n, t_n)$ and an event e that is enabled after σ, that is $e \in \mathsf{en}(\lfloor\sigma\rfloor)$, let $\mathsf{time}(\sigma, e)$ denote the minimal time instant from which e can occur. Event e can occur if (i) its absolute delay $\mathcal{D}(e)$ is respected, (ii) the time relative to all its immediate causal predecessors is respected, and (iii) for each event e_j with $e_j \rightsquigarrow e$ we have that e occurs at at least t_j.

4.4. DEFINITION. For σ a sequence of timed events $(e_1, t_1) \ldots (e_n, t_n)$ with $e_i \in E$, $t_i \in \mathsf{Time}$ for $0 < i \leqslant n$, and $e \in \mathsf{en}(\lfloor\sigma\rfloor)$, let

$$\begin{aligned}\mathsf{time}(\sigma, e) \triangleq\, & Max(\{\mathcal{D}(e)\} \cup H_1 \cup H_2) \text{ where}\\ & H_1 = \{t + t_j \mid \exists X \subseteq E : X \overset{t}{\mapsto} e \ \wedge\ X \cap \overline{\lfloor\sigma\rfloor} = \{e_j\}\}\\ & H_2 = \{t_j \mid \exists e_j \in \overline{\lfloor\sigma\rfloor} : e_j \rightsquigarrow e\} \ .\end{aligned}$$

□

When $e_j \rightsquigarrow e$ and e_j has occurred, then e_j should temporally precede e. This is a natural extension of the untimed case in which e_j *causally* precedes e if both events occur. Since events cannot happen before their causes, causal ordering implies temporal ordering.

A timed sequential observation of the system is now defined as an untimed sequential observation where each event has a correct timing associated with it.

4.5. DEFINITION. (*Timed event trace*)

A *timed event trace* of timed event structure $\Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ is a sequence σ of timed events $(e_1, t_1) \ldots (e_n, t_n)$ with $e_i \in E$, $t_i \in \mathsf{Time}$, for all $0 < i \leqslant n$, satisfying

1. $e_1 \ldots e_n \in T(\mathcal{E})$
2. $\forall\, i : t_i \geqslant \mathsf{time}(\sigma_i, e_i)$.

□

Note that, according to the last constraint, an event can happen at any time from the moment it is enabled. Let $T_T(\Gamma)$ denote the set of timed event traces of Γ

4.6. EXAMPLE. For the following sequences of timed events we give the conditions under which they are timed event traces of Figure 4.1(a):

$$\begin{array}{lll} (e_a, t_a)\,(e_b, t_b)\,(e_d, t_d) & \text{if} & t_d \geqslant t_b{+}2, \text{ and} \\ (e_a, t_a)\,(e_b, t_b)\,(e_c, t_c) & \text{if} & t_c \geqslant max(t_a{+}3, t_b{+}5) \quad . \end{array}$$

□

4.7. DEFINITION. $\sigma, \sigma' \in T_T(\Gamma)$ are *timed configuration equivalent*, denoted $\sigma \sim_T \sigma'$, iff $\overline{\sigma} = \overline{\sigma}'$. □

Note that $\sim_T \subseteq \sim$, where $\sigma \sim \sigma'$ iff $\overline{[\sigma]} = \overline{[\sigma']}$.

Timed event traces do respect causality, but not necessarily time. That is, two (or more) independent events can occur in a trace in either order regardless of their timing. For example, $(e_b, 1)(e_a, 3)$ and $(e_a, 3)(e_b, 1)$ are timed event traces of Figure 4.1(a). The possible choices correspond to the possible interleavings of the causally independent events. Although it may at first sight be counterintuitive to allow traces that do not respect time, their appearance can be understood from the fact that event traces are linearizations of (timed) partial orders. Since the causal ordering between events implies their temporal ordering the causal ordering can never contradict the temporal order.

4.8. DEFINITION. Timed event trace σ is *time-consistent* iff $\forall\, i, j : i < j \;\Rightarrow\; t_i \leqslant t_j$. □

Predicate $tc(\sigma)$ is true iff σ is time-consistent. A timed event trace that is not time-consistent is called *ill-timed*. The fact that ill-timed traces can only appear due to the possible interleavings of independent events follows from the following result.

4.9. THEOREM. *Ill-timed theorem*

For $t' < t$: $\sigma\,(e, t)\,(e', t')\,\sigma' \in T_T(\Gamma) \;\Rightarrow\; \sigma\,(e', t')\,(e, t)\,\sigma' \in T_T(\Gamma)$.

PROOF. Let $\sigma^1 = \sigma\,(e, t)\,(e', t')\,\sigma'$ and $\sigma^2 = \sigma\,(e', t')\,(e, t)\,\sigma'$. Let $t' < t$ and $\sigma^1 \in T_T(\Gamma)$. The proof is by contradiction. Suppose $\sigma^2 \notin T_T(\Gamma)$. This can only be because one of the following reasons:

1. $[\sigma^2]$ is not an event trace of $\mathcal{E}$. This can only be because one of the following reasons:
   (a) There exists a bundle $X \mapsto e'$ with $e \in X$. But then, according to the second constraint of Definition 4.5, $t' \geqslant t$. Contradiction.
   (b) $e \rightsquigarrow e'$ (i.e., e' disables e). According to the second constraint of Definition 4.5 then $t' \geqslant t$. Contradiction.

   This proves that $[\sigma^2]$ is an event trace of $\mathcal{E}$.

2. $\exists\, j : t_j < \mathsf{time}(\sigma^2_j, e_j)$. This can only be because of one of the following reasons:
   (a) (i) $t < \mathcal{D}(e)$, or (i') $t' < \mathcal{D}(e')$. These cases contradict with the fact $\sigma^1 \in T_T(\Gamma)$.
   (b) (i) there exists a bundle $X \overset{t}{\mapsto} e'$ with $e_j \in X$ and $t' < t_j{+}t$. This contradicts with $\sigma^1 \in T_T(\Gamma)$. (i') there exists a bundle $X \overset{t}{\mapsto} e$ with $e_j \in X$ and $t < t_j{+}t$. For this case we distinguish between $e_j \neq e'$ and $e_j = e'$. For $e_j \neq e'$ we have that $t \geqslant t_j{+}t$, otherwise $\sigma^1 \notin T_T(\Gamma)$. Consider $e_j = e'$. This is impossible, since e precedes e' in σ^1 and $\sigma^1 \in T_T(\Gamma)$. Contradiction.
   (c) (i) $e \rightsquigarrow e'$ and $t > t'$, or (i') $e' \rightsquigarrow e$ and $t' > t$. Case (i) cannot occur (similar to case 1.(b)) and (i') contradicts with the assumption that $t' < t$.

□

This theorem implies that for any ill-timed event trace σ there exists a corresponding time-consistent event trace σ', that can be obtained from σ by swapping repeatedly ill-timed pairs of timed events, yielding $\overline{\sigma} = \overline{\sigma}'$. Note that the reverse implication of Theorem 4.9 does not hold; for instance, if e causally depends on e' then the order of events $e'\,e$ in a trace cannot be reversed since this would contradict their causal ordering.

For a more extensive discussion on ill-timed traces we refer to Aceto & Murphy [1, 2].

4.2.3 A lattice of timed traces

The timed model only allows for the specification of minimal time constraints. That is, only lower bounds on the occurrence time of events can be dealt with. In this section we show that all timed event traces that contain the same events but possibly with different timing can be considered as a lattice (ordered under a 'faster than' relation) with as a least element a trace in this class with the minimal correct timing.

σ is called a *fast trace* iff all events in σ have a minimal correct timing, i.e., they all occur as soon as possible. E.g. $(e_a, 0)(e_b, 0)$, $(e_b, 0)(e_a, 0)$, $(e_b, 0)(e_d, 2)$ and$(e_b, 0)(e_a, 0)(e_c, 5)$ are fast traces of Figure 4.1(a).

4.10. Definition. $\sigma \in T_T(\Gamma)$ is *fast* iff $\forall\, i : t_i = \mathsf{time}(\sigma_i, e_i)$. □

In the rest of this section we assume that σ and σ' are representers of equivalence classes under $\sim_T$, i.e., σ and σ' represent classes of timed traces that all have the same timed events, but possibly in a different order. The following is relative to Γ with $\sigma, \sigma' \in T_T(\Gamma)$. Let $\sigma = (e_1, t_1) \ldots (e_n, t_n)$ and $\sigma' = (e_1, t'_1) \ldots (e_n, t'_n)$.

4.11. DEFINITION. For σ, σ' with $\sigma \sim \sigma'$ let $\sigma \preccurlyeq \sigma'$ iff $\forall\, i : t_i \leqslant t'_i$. □

It can easily be verified that $\preccurlyeq$ (pronounced *faster than*) is a partial order on event equivalent classes of configuration equivalent timed event traces.

4.12. LEMMA. σ is a fast timed event trace iff $(\forall\, \sigma' \in [\sigma]_\sim : \sigma \preccurlyeq \sigma')$.

PROOF. '$\Rightarrow$': Let σ be a fast timed event trace of Γ. For $\sigma = \sigma'$ the lemma trivially holds. Consider $\sigma \neq \sigma'$ and $\sigma \sim \sigma'$. The proof for this case is by contradiction, distinguishing between (1) $\sigma' \preccurlyeq \sigma$, and (2) $\sigma \not\preccurlyeq \sigma' \ \wedge\ \sigma' \not\preccurlyeq \sigma$.

1. Suppose $\sigma' \preccurlyeq \sigma$. Then, there exists e_i, say, such that $t'_i < t_i$. Since σ is a fast timed trace we have $t_i = Max(\{\, \mathcal{D}(e_i) \,\} \cup H_1 \cup H_2)$, cf. Definition 4.10. Suppose there are K ($K \geqslant 0$) bundles $X_k \overset{t_k}{\mapsto} e_i$ in Γ ($0 < k \leqslant K$) and $X_k \cap \overline{[\sigma]} = \{\, e_{jk} \,\}$. Then we have $H_1 = \{\, t_{j1}{+}t_1, \ldots, t_{jK}{+}t_K \,\}$. For N ($N \geqslant 0$) events $e_{jn} \rightsquigarrow e_i$ in Γ with e_{jn} in $\overline{[\sigma]}$ we have $H_2 = \{\, t_{j1}, \ldots, t_{jN} \,\}$. Now consider $t'_i < t_i$. Then either (a) $t'_i < Max(H_1)$ or (b) $t'_i < Max(H_2)$.
   (a) Suppose $t'_i < Max(H_1)$. As $\sigma \sim \sigma'$ and for each bundle $X \mapsto e_i$ in Γ, e_i has a unique causal predecessor in σ, e_i is enabled in σ and σ' by the same set of events:
   $$X \mapsto e_i \ \wedge\ X \cap \overline{[\sigma]} = \{\, e_j \,\} \ \Rightarrow\ (\forall\, \sigma' \in [\sigma]_\sim : X \cap \overline{[\sigma']} = \{\, e_j \,\}) \quad .$$
   But then it immediately follows $t'_i \geqslant Max(H_1)$. Contradiction.
   (b) Suppose $t'_i < Max(H_2)$. As $\sigma \sim \sigma'$ we also have that
   $$\{\, e \in \overline{[\sigma]} \mid e \rightsquigarrow e_i \,\} = \{\, e \in \overline{[\sigma']} \mid e \rightsquigarrow e_i \,\} \quad .$$
   But then it immediately follows $t'_i \geqslant Max(H_2)$. Contradiction.
2. Suppose $\sigma \not\preccurlyeq \sigma' \ \wedge\ \sigma' \not\preccurlyeq \sigma$. By definition of $\preccurlyeq$ this equals $(\exists\, e_i : t_i > t'_i) \ \wedge\ (\exists\, e_i : t'_i > t_i)$. Using an analogous argument as for case 1. we can prove that the first conjunct does not hold.

'$\Leftarrow$': Straightforward by contradiction and omitted. □

4.13. LEMMA. $\langle [\sigma]_\sim, \preccurlyeq \rangle$ is a poset with a least element.

PROOF. Straightforward from the previous lemma and the fact that for each σ class $[\sigma]_\sim$ contains a fast timed event trace. □

4.14. EXAMPLE. Consider the timed event structure of Figure 4.1(c) and some of its timed event traces:

$$\begin{aligned} \sigma_1 &= (e_a, 0)\,(e_d, 3)\,(e_c, 4) \\ \sigma_2 &= (e_a, 0)\,(e_d, 10)\,(e_c, 4) \\ \sigma_3 &= (e_a, 3)\,(e_d, 12)\,(e_c, 7) \end{aligned} \qquad \begin{aligned} \sigma_4 &= (e_a, 3)\,(e_d, 5)\,(e_c, 7) \\ \sigma_5 &= (e_a, 0)\,(e_d, 1)\,(e_c, 4) \quad . \end{aligned}$$

σ_5 is a fast timed event trace. $\preccurlyeq$ is the reflexive and transitive closure of $\sigma_5 \preccurlyeq \sigma_1$, $\sigma_1 \preccurlyeq \sigma_4$, $\sigma_1 \preccurlyeq \sigma_2$, $\sigma_2 \preccurlyeq \sigma_3$, and $\sigma_4 \preccurlyeq \sigma_3$. □

For σ, σ' such that $\sigma \sim \sigma'$, let $lub(\sigma, \sigma')$ be the sequence of slowest timed events faster than both σ and σ'. Similarly, $glb(\sigma, \sigma')$ is defined as the sequence of fastest events slower than both σ and σ'.

4.15. DEFINITION. For $\sigma = (e_1, t_1) \ldots (e_n, t_n)$ and $\sigma' = (e_1, t_1') \ldots (e_n, t_n')$ let

- $lub(\sigma, \sigma') \triangleq (e_1, min(t_1, t_1')) \ldots (e_n, min(t_n, t_n'))$
- $glb(\sigma, \sigma') \triangleq (e_1, max(t_1, t_1')) \ldots (e_n, max(t_n, t_n'))$.

$\square$

4.16. LEMMA. $\forall\, \sigma, \sigma' \in T_T(\Gamma) : lub(\sigma, \sigma') \in T_T(\Gamma) \ \wedge \ glb(\sigma, \sigma') \in T_T(\Gamma)$.

PROOF. By contradiction. We provide the proof for *lub*, the proof for *glb* is similar. Let $\sigma, \sigma' \in T_T(\Gamma)$ and suppose $\sigma'' = lub(\sigma, \sigma') \notin T_T(\Gamma)$. This can only be because of one of the following reasons:

1. $[\sigma''] \notin T(\mathcal{E})$. Then $[\sigma], [\sigma'] \notin T(\mathcal{E})$. Contradiction.
2. $\exists\, e_i : t_i'' < \mathsf{time}(\sigma_i'', e_i)$. This can only be because one of the following reasons:
   (a) $t_i'' < \mathcal{D}(e_i)$, i.e., $min(t_i, t_i') < \mathcal{D}(e_i)$. Then $t_i < \mathcal{D}(e_i)$ or $t_i' < \mathcal{D}(e_i)$. Contradiction.
   (b) $e_i \rightsquigarrow e_j$ and $t_i'' > t_j''$. Then, by definition of *lub*, $min(t_i, t_i') > min(t_j, t_j')$.

$$
\begin{array}{ll}
& min(t_i, t_i') > min(t_j, t_j') \\
\Leftrightarrow & \{ \text{ definition of } min \ \} \\
& (t_i \leqslant t_i' \ \wedge \ t_i > min(t_j, t_j')) \ \vee \ (t_i' \leqslant t_i \ \wedge \ t_i' > min(t_j, t_j')) \\
\Leftrightarrow & \{ \text{ definition of } min \ \} \\
& (t_i \leqslant t_i' \ \wedge \ (t_i > t_j \ \vee \ t_i > t_j')) \ \vee \ (t_i' \leqslant t_i \ \wedge \ (t_i' > t_j \ \vee \ t_i' > t_j')) \\
\Leftrightarrow & \{ \ \sigma \in T_T(\Gamma) \ \Rightarrow \ t_i \leqslant t_j \text{ (idem for } \sigma') \ \} \\
& (t_i \leqslant t_i' \ \wedge \ t_i > t_j') \ \vee \ (t_i' \leqslant t_i \ \wedge \ t_i' > t_j) \\
\Leftrightarrow & \{ \ \sigma \in T_T(\Gamma) \ \Rightarrow \ t_i \leqslant t_j \text{ (idem for } \sigma') \ \} \\
& \text{false} \quad .
\end{array}
$$

   (c) $\exists\, X : X \overset{t}{\mapsto} e_i$ and $e_j \in X$ and $t_i'' < t_j'' + t$. Then by definition of *lub*, $min(t_i, t_i') < min(t_j, t_j') + t$. In a similar way as the previous case it can be proven that this leads to a contradiction.

$\square$

4.17. THEOREM. $\langle \langle [\sigma]_\sim, \preccurlyeq \rangle, lub, glb \rangle$ is a lattice with a least element.

PROOF. Directly from Lemma 4.13 and 4.16. $\square$

This lattice construction is possible since timed event structures allow for the specification of *minimal* time constraints only. Later on we will encounter models which do also allow the specification of *maximal* time constraints, and we will see that for those models the above lattice construction does not work.

4.2.4 Families of lposets

The semantics of a timed event structure is defined by means of its family of labelled partially ordered sets (lposets). In this section we define how to obtain these lposets and investigate the relation between the lposets of Γ and the lposets of its corresponding untimed counterpart $\mathcal{E}$. For simplicity we only define lposets in an operational way, i.e., starting from timed event traces. The intensional characterization as provided in Chapter 6 for urgent event structures can be applied in a similar way to the model of this chapter.

4.18. DEFINITION. (*Lposets of a timed event structure*)

For $\Gamma \in \mathsf{EBES}_T : L_T(\Gamma) \triangleq \{ \langle \overline{\sigma}, \bigcap_{\sigma' \in [\sigma]_{\sim_T}} <^*_{\sigma'}, l \restriction \overline{\sigma} \rangle \mid \sigma \in T_T(\Gamma) \}$. □

Here l is the labelling function of Γ and $l((e,t)) = l(e)$. We consider all $\sigma \in T_T(\Gamma)$ and consider its class of timed configuration-equivalent timed traces, $[\sigma]_{\sim_T}$. With each $\sigma' \in [\sigma]_{\sim_T}$ an ordering on timed events $<^*_{\sigma'}$ is associated which reflects the precedence of timed events in σ'. More specifically, if $\sigma' = (e'_1, t'_1) \ldots (e'_n, t'_n)$ then $<^*_{\sigma'}$ is defined as (the reflexive and transitive closure of) $(e'_1, t'_1) <_{\sigma'} (e'_2, t'_2) <_{\sigma'} \ldots <_{\sigma'} (e'_n, t'_n)$. It is easy to verify that $\bigcap_{\sigma' \in [\sigma]_{\sim_T}} <^*_{\sigma'}$ is a partial order on $\overline{\sigma}$.

For $\sigma \in T_T(\Gamma)$ we sometimes use $L_T(\sigma)$ as an abbreviation for $\langle \overline{\sigma}, \bigcap_{\sigma' \in [\sigma]_{\sim_T}} <^*_{\sigma'}, l \restriction \overline{\sigma} \rangle$.

4.19. THEOREM. $\forall \Gamma, \Gamma' \in \mathsf{EBES}_T : T_T(\Gamma) = T_T(\Gamma') \Longleftrightarrow L_T(\Gamma) = L_T(\Gamma')$.

PROOF. Similar to the untimed case [89, Theorem 6.3.12] and omitted here. □

The untimed lposets of Γ are now deduced from $L_T(\Gamma)$ as follows:

4.20. DEFINITION. For $\Gamma \in \mathsf{EBES}_T$ the untimed lposets of Γ are defined as

$$L(\Gamma) \triangleq \{ \langle [E], \leqslant \restriction [E], l \rangle \mid \langle E, \leqslant, l \rangle \in L_T(\Gamma) \}.$$

□

Here, $[E]$ denotes the set of events in E. That is, for $E = \{ (e_1, t_1), \ldots, (e_n, t_n) \}$ we have that $[E] \triangleq \{ e_1, \ldots, e_n \}$. $\leqslant \restriction [E]$ denotes $\{ (e, e') \in [E] \mid \exists\, t, t' : (e,t) \leqslant (e', t') \}$.

The following theorem shows that the inclusion of minimal time constraints into event structures retains the causal dependencies as present in the untimed case. Suppose we remove the timed components of Γ and determine the lposets of this untimed structure, then this yields the same result as if we would first calculate the lposets of Γ and then abstract from time. We denote the removal of timed components from Γ by φ. For $\Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ we simply have $\varphi(\Gamma) \triangleq \mathcal{E}$.

4.21. THEOREM. $\forall \Gamma \in \mathsf{EBES}_T : L(\Gamma) = L(\phi(\Gamma))$.

PROOF.

$L(\Gamma)$

$$
\begin{aligned}
&= \quad \{\text{ Definition 4.20 }\}\\
&\quad \{\, \langle [E], \leqslant\restriction [E], l\rangle \mid \langle E, \leqslant, l\rangle \in L_T(\Gamma)\,\}\\
&= \quad \{\text{ Definition 4.18 }\}\\
&\quad \{\, \langle [E], \leqslant\restriction [E], l\rangle \mid \langle E, \leqslant, l\rangle \in \{\, \langle \overline{\sigma}, \textstyle\bigcap_{\sigma' \in [\sigma]_{\sim_T}} <^*_{\sigma'}, l \restriction \overline{\sigma}\rangle \mid \sigma \in T_T(\Gamma)\,\}\,\}\\
&= \quad \{\ \}\\
&\quad \{\, \langle \overline{[\sigma]}, \textstyle\bigcap_{[\sigma'] \in [\sigma]_{\sim_T}} <^*_{[\sigma']}, l \restriction \overline{[\sigma]}\rangle \mid \sigma \in T_T(\Gamma)\,\}\\
&= \quad \{\ \sigma \in T_T(\Gamma) \Leftrightarrow [\sigma] \in T(\varphi(\Gamma))\ \}\\
&\quad \{\, \langle \overline{[\sigma]}, \textstyle\bigcap_{\sigma' \in [[\sigma]]_{\sim}} <^*_{\sigma'}, l \restriction \overline{[\sigma]}\rangle \mid [\sigma] \in T(\varphi(\Gamma))\,\}\\
&= \quad \{\text{ Definition 2.24 }\}\\
&\quad L(\varphi(\Gamma)) \quad .
\end{aligned}
$$

□

4.2.5 Timed remainder

Like for the untimed case we are interested in the status of a timed event structure after the execution of a sequence of timed events. In this section we define the notion of timed remainder and prove its correctness.

4.22. Definition. (*Timed remainder*)

The *timed remainder* of timed event structure $\Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}\rangle$ after timed event trace σ, is $\Gamma[\sigma] = \langle \mathcal{E}', \mathcal{D}', \mathcal{T}'\rangle$ where

- $\mathcal{E}' = \mathcal{E}[[\sigma]] = (E', \rightsquigarrow', \mapsto', l')$
- $\forall\, e \in E' : \mathcal{D}'(e) = Max(\{\, \mathcal{D}(e)\,\} \cup H_1 \cup H_2)$ with
  $H_1 = \{\, t + t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \ \wedge\ X \cap \overline{[\sigma]} = \{\, e_j\,\}\,\}$ and
  $H_2 = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow e\,\}$
- $\mathcal{T}' = (\mathcal{T} \restriction \mapsto') \cup \{\, ((\varnothing, e), \star) \mid \varnothing \mapsto' e\,\}$ for some $\star \in \mathsf{Time}$.

□

The first component is equal to the remainder of $\mathcal{E}$. The timings associated with the retained bundles are unaffected and since $\mathcal{T}'$ is a total function the introduced bundles (cf. Definition 2.28) are associated a time value. Since the events pointed to by these bundles will never happen, this time value is arbitrary.

In addition, the delay of an event e which has a bundle pointing to it originating from some event e_j in σ has to be checked: if t_j plus the required relative time, t say, between e_j and e is larger than the delay of e, e should be postponed to (at least) $t{+}t_j$. Because this should hold for all bundles pointing to e originating from some event in σ, the maximum is taken such that all required relative delays are satisfied.

Finally, in order to enforce that the causal relation between e_j and e induces a temporal precedence, the delay of e becomes at least t_j in case $e_j \rightsquigarrow e$. Again, this should hold for all asymmetric conflicts to e originating from some event in σ, resulting in the max-construction above.

It is quite straightforward to check that for all $\Gamma \in \mathsf{EBES}_T$ and $\sigma \in T_T(\Gamma)$ we have $\Gamma[\sigma] \in \mathsf{EBES}_T$, since $\mathcal{E}[[\sigma]] \in \mathsf{EBES}$ and all events and bundles in $\Gamma[\sigma]$ are assigned a time value.

4.23. Example. The remainder of a timed event structure is exemplified in Figure 4.2 and Figure 4.3. Figure 4.2 shows how event delays are updated due to the presence of bundles originating from events in the configuration, whereas Figure 4.3 shows how this procedure works when asymmetric conflicts cause the update. □

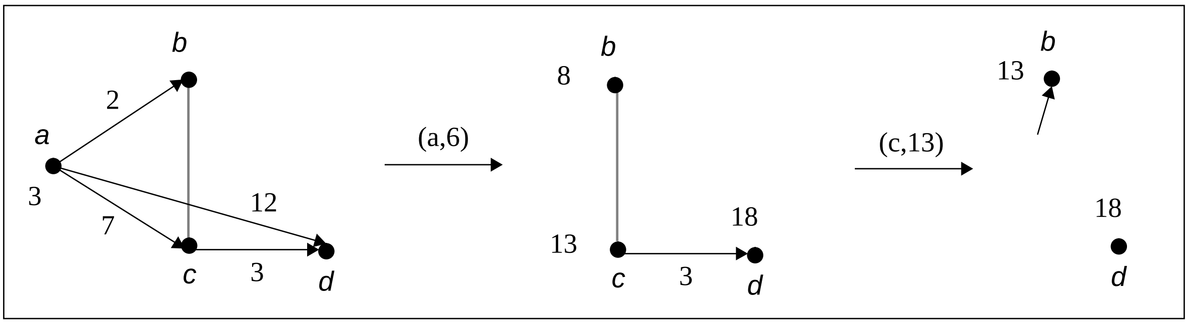


Figure 4.2: Example remainder of a timed event structure (I).

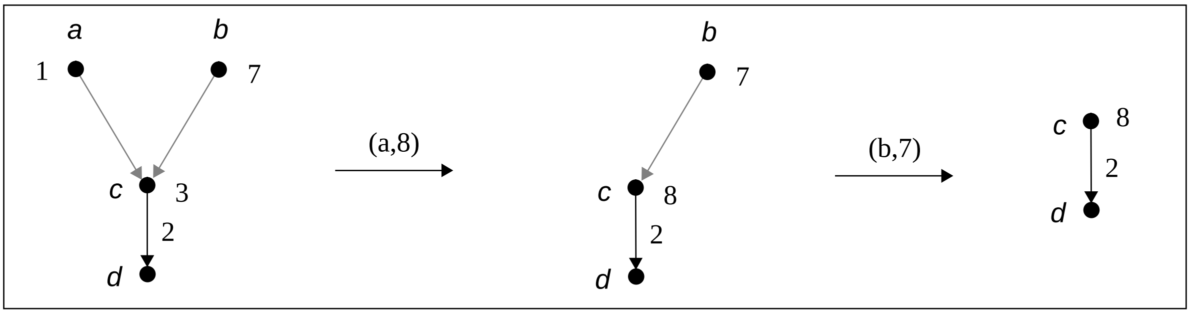


Figure 4.3: Example remainder of a timed event structure (II).

We have the following correctness result concerning the definition of timed remainder. It says that if Γ can evolve into Γ' by executing σ then σ' is a trace of Γ' iff $\sigma\,\sigma'$ is a trace of Γ. In addition, it states that the lposet induced by $\sigma\,\sigma'$ is an extension of the lposet induced by σ.

4.24. Theorem. *Correctness of timed remainder*

For $\sigma \in T_T(\Gamma)$ and σ' a sequence of timed events:

1. $\sigma' \in T_T(\Gamma[\sigma]) \iff \sigma\,\sigma' \in T_T(\Gamma)$
2. $\sigma' \in T_T(\Gamma[\sigma]) \;\Rightarrow\; L_T(\sigma)$ is a prefix of $L_T(\sigma\,\sigma')$.

Proof.

1. Let $\Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ and $\Gamma[\sigma] = \Gamma' = \langle \mathcal{E}', \mathcal{T}', \mathcal{D}' \rangle$.

   '$\Rightarrow$' : Assume $\sigma \in T_T(\Gamma)$ and $\sigma' \in T_T(\Gamma')$. We prove that $\sigma'' = \sigma\,\sigma' \in T_T(\Gamma)$ by contradiction. So, suppose $\sigma'' \notin T_T(\Gamma)$. From the untimed case (cf. Theorem 2.30) we know that if $[\sigma] \in T(\mathcal{E})$

and $[\sigma'] \in T(\mathcal{E}')$ then $[\sigma''] \in T(\mathcal{E})$. Thus there can only be one reason for σ'' not being a timed event trace of Γ, viz. violation of constraint 2 of Definition 4.5: $\exists\, i : t_i < \mathsf{time}(\sigma''_i, e_i)$. This can only have the following causes:

(a) $t_i < \mathcal{D}(e_i)$. If $e_i \in \overline{[\sigma]}$ this is impossible, since $\sigma \in T_T(\Gamma)$, requiring $t_i \geqslant \mathcal{D}(e_i)$. Suppose $e_i \in \overline{[\sigma']}$. For Γ' it follows directly from Definition 4.22 that for all $e \in E' : \mathcal{D}(e) \leqslant \mathcal{D}'(e)$. As $\sigma' \in T_T(\Gamma')$ we have $t_i \geqslant \mathcal{D}'(e_i)$, and thus, $t_i \geqslant \mathcal{D}(e_i)$. Contradiction.

(b) $\exists\, X \subseteq E : X \stackrel{t}{\mapsto} e_i \ \wedge\ e_j \in X$ and $t_i < t_j{+}t$. The interesting case is when $X \cap \overline{[\sigma]} \neq \varnothing$ and $e_i \in \overline{[\sigma']}$. Since $X \cap \overline{[\sigma]} \neq \varnothing$ the bundle $X \stackrel{t}{\mapsto} e_i$ cannot be in Γ', so it has been removed according to Definition 2.28. But then $\mathcal{D}'(e_i)$ has been updated (cf. Definition 4.22) such that $\mathcal{D}'(e_i) \geqslant t_j + t$. As $\sigma' \in T_T(\Gamma')$ we have $t_i \geqslant \mathcal{D}'(e_i)$, and thus $t_i \geqslant t_j + t$. Contradiction.

(c) $\exists\, e_i, e_j : e_i \rightsquigarrow e_j \ \wedge\ t_j < t_i$. The interesting case is when $e_i \in \overline{[\sigma]}$ and $e_j \in \overline{[\sigma']}$. Suppose e_i happens at t_i. Then, according to Definition 4.22, $\mathcal{D}(e_j)$ has been updated such that $\mathcal{D}'(e_j) \geqslant t_i$. Since $\sigma' \in T_T(\Gamma')$ we have $t_j \geqslant \mathcal{D}'(e_j)$, and consequently, $t_j \geqslant t_i$. Contradiction.

'$\Leftarrow$' : Assume $\sigma \in T_T(\Gamma)$ and $\sigma\,\sigma' \in T_T(\Gamma)$. We prove that $\sigma' \in T_T(\Gamma')$ by contradiction. So, suppose $\sigma' \notin T_T(\Gamma')$. From the untimed case we know that $[\sigma'] \in T(\mathcal{E}')$, so if σ' is not a timed event trace of Γ' this can only be because $\exists\, e_i : t_i < \mathsf{time}(\sigma'_i, e_i)$. That is, either

(a) $t_i < \mathcal{D}'(e_i)$. From the fact that $\sigma\,\sigma' \in T_T(\Gamma)$ we know that $t_i \geqslant \mathcal{D}(e_i)$. $t_i < \mathcal{D}'(e_i)$ and $t_i \geqslant \mathcal{D}(e_i)$ means that the delay of e_i is updated by the execution of $\overline{\sigma}$. There are two possibilities for doing so:

   i. $\exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow e_i$. Then $\mathcal{D}'(e_i) = t_j$ and $t_j > \mathcal{D}(e_i)$. As $\sigma\,\sigma' \in T_T(\Gamma)$ we have $t_i \geqslant t_j$, and consequently, $t_i \geqslant \mathcal{D}'(e_i)$. Contradiction.

   ii. $\exists\, e_j : X \stackrel{t}{\mapsto} e_i \ \wedge\ X \cap \overline{[\sigma]} = \{\, e_j \,\}$. Then $\mathcal{D}'(e_i) = t_j + t$ and $t_j + t > \mathcal{D}(e_i)$. As $\sigma\,\sigma' \in T_T(\Gamma)$ we have $t_i \geqslant t_j + t$, and thus, $t_i \geqslant \mathcal{D}'(e_i)$. Contradiction.

(b) $X \stackrel{t}{\mapsto}{}' e_i$ with $e_j \in X$ and $t_i < t_j + t$. But then this bundle is either already present in Γ or is a newly created bundle. For the first case it immediately follows from the fact that $\sigma\,\sigma' \in T_T(\Gamma)$ that $t_i \geqslant t_j + t$. Contradiction. For the second case we have that (cf. Definition 2.28) $X = \varnothing$ and that there is an $e \in \overline{[\sigma]}$ such that $e_i \rightsquigarrow e$. Since $\overline{[\sigma]} \subseteq \overline{[\sigma\,\sigma']}$ this would mean that $\sigma\,\sigma' \notin T_T(\Gamma)$. Contradiction.

(c) $\exists\, e_i, e_j : e_i \rightsquigarrow' e_j \ \wedge\ t_j < t_i$. Since $\rightsquigarrow' \subseteq \rightsquigarrow$ this implies that $e_i \rightsquigarrow e_j$. As $\sigma\,\sigma' \in T_T(\Gamma)$ we have $t_i \leqslant t_j$. Contradiction.

2. From 1. it follows that $\sigma\,\sigma' \in T_T(\Gamma)$, so $L_T(\sigma\,\sigma')$ is defined. Clearly $\overline{\sigma} \subseteq \overline{\sigma\,\sigma'}$ and $<^*_{\sigma} \subseteq <^*_{\sigma\,\sigma'}$. Besides, since no event in σ' precedes an event in σ under $<_{\sigma\,\sigma'}$ it follows that

$$<^*_{\sigma\,\sigma'} \cap ((<^*_{\sigma'} \ \cup\ <^*_{\sigma}) \times <^*_{\sigma}) = <^*_{\sigma\,\sigma'} \cap (<^*_{\sigma} \times <^*_{\sigma}) = <^*_{\sigma} \quad .$$

This proves that $L_T(\sigma)$ is a prefix of $L_T(\sigma\,\sigma')$.

$\square$

4.2.6 Some transformation rules

Figure 4.4 presents some transformation rules for timed event structures that preserve lposets. We use the same notational conventions as in Section 2.3.4. The first rule gives a recipe for removing redundant bundles. Since $t'' \leqslant t+t'$ the relative time between X and e does not contribute to the delay of e, and hence can be safely removed. The fact that the bundle may be removed follows from the rule for the untimed case obtained by omitting all timing information in the depicted rule. In the second rule delay d of X indicates the minimal delay of some event in X. Since $X \overset{t}{\mapsto} e$ event e has at least delay $t+d$. In case $d' \leqslant t+d$ the event delay d' of e is superfluous and may be replaced by 0. The third rule allows for the removal of sub-bundles and is a straightforward generalization of a similar rule for the untimed case.

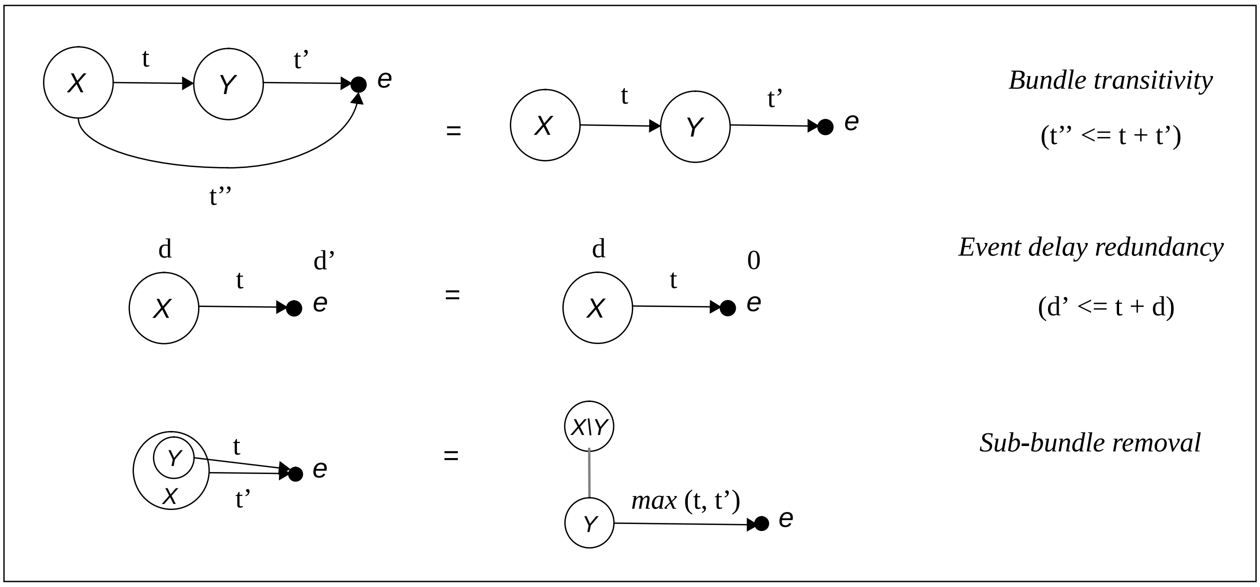


Figure 4.4: Some transformation rules for timed event structures.

The formal representation of the transformation rules is as follows:

4.25. Theorem. Timed event structure $\langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ is lposet equivalent with

1. $\langle (E, \rightsquigarrow, \mapsto \setminus \{\, (X,e) \,\}, l), \mathcal{D}, \mathcal{T} \setminus \{\, ((X,e),t'') \,\} \rangle$
   if $Y \overset{t'}{\mapsto} e \ \wedge \ X \overset{t}{\mapsto} Y \ \wedge \ X \overset{t''}{\mapsto} e \ \wedge \ t'' \leqslant t+t'$.
2. $\langle (E, \rightsquigarrow, \mapsto, l), (\mathcal{D} \setminus \{\, (e,d') \,\}) \cup \{\, (e,0) \,\}, \mathcal{T} \rangle$
   if $X \overset{t}{\mapsto} e \ \wedge \ \mathcal{D}(e) = d' \ \wedge \ d' \leqslant t+Min\{\, \mathcal{D}(e') \mid e' \in X \,\}$.
3. $\langle (E, \rightsquigarrow, \mapsto \setminus \{\, (X,e) \,\}, l), \mathcal{D}, (\mathcal{T} \setminus \{\, ((Y,e),t), ((X,e),t') \,\}) \cup \{\, ((Y,e),d) \,\} \rangle$
   if $Y \subseteq X \ \wedge \ X \overset{t'}{\mapsto} e \ \wedge \ Y \overset{t}{\mapsto} e \ \wedge \ d = max(t,t')$.

Proof. We only prove rules 2. and 3. as an example; the proof for rule 1. is similar. For each rule let Γ_l and Γ_r denote the left-hand and right-hand timed event structure, respectively.

2. The only difference between these two timed event structures is that Γ_l requires e to happen after time d'. The proof is by contradiction. Suppose that Γ_r has a timed event trace $\sigma\,(e, t')$ for which $t' < d'$ and $d' \leqslant t{+}d$ where $d = \mathit{Min}\{\, \mathcal{D}(e') \mid e' \in X\,\}$. Since X points to e, event e must be preceded by some event e_i in X. We have $t_i \geqslant d$. But then, since $X \overset{t}{\mapsto} e$ we have $t' \geqslant t{+}d$, and since $t{+}d \geqslant d'$, it follows $t' \geqslant d'$. Contradiction. So, the timed event structures have the same set of timed event traces, and by Theorem 4.19, also the same family of lposets.

3. Suppose Γ_l has timed trace $\sigma\,(e, t')\,\sigma'$. Event e can only occur if both bundles X and Y are satisfied. Since $Y \subseteq X$ and all events in X are in mutual conflict there is one event, e_i say, in $\overline{[\sigma]}$ which belongs to Y. As $\mathsf{time}(\sigma, e) = \mathit{Max}\{\, \ldots, t_i{+}t, t_i{+}t', \ldots\} = \mathit{Max}\{\, \ldots, t_i{+}\mathit{max}(t, t'), \ldots\}$ it follows that $\sigma\,(e, t')\,\sigma'$ is a trace of Γ_r. Obviously, for traces not involving e we also have that Γ_l and Γ_r are event trace equivalent. So, the timed event structures have the same set of timed traces, and by Theorem 4.19, also the same family of lposets.

□

4.3 A timed process algebra

This section introduces a simple timed process algebra and provides a causality-based semantics using timed event structures. Section 4.3.1 introduces the temporal process algebra, Section 4.3.2 provides the semantics and Section 4.3.3 provides some syntactical constraints that aim at a simplification of the timed event structures model.

4.3.1 Syntax

For $t \in \mathsf{Time}$ the syntax of PA_T of finite simple timed behaviours is defined as:

4.26. Definition. (*Simple timed process algebra* PA_T)

$$B ::= \mathbf{0} \;\Big|\; \surd \;\Big|\; (t)\ a\,;B \;\Big|\; B+B \;\Big|\; B\,||_G\,B \;\Big|\; B[H] \;\Big|\; B \setminus G \;\Big|\; B >> B \;\Big|\; B\,[> B. \qquad \square$$

PA_T is a timed extension of PA, the process algebra introduced in Chapter 1. Actions are considered to be atomic and occur instantaneously. The elementary timing construct of our language is a *delay function* that expresses the relative delay of an action. Behaviour $a\,;(t)\,b\,;\mathbf{0}$ behaves identically to $a\,;b\,;\mathbf{0}$, except that it is able to engage in b from t time units after the occurrence of a. For initial actions the time is related to the beginning of the system at hand. We abbreviate $(0)\,a$ by a. We also allow arithmetic expressions and consider syntactic equivalence to be modulo equal arithmetic expressions, identifying for example $(2{+}5)\ a\,;B$ and $(7)\ a\,;B$.

Behaviours may synchronize on a common action as soon as all participants are ready to engage in it, i.e., when all individual timing constraints on such action are met. This choice has been inspired by the constraint-oriented specification style of Vissers *et al.* [148], where global constraints (on the ordering of events) are expressed by conjunction (using parallel composition) of individual (or local) constraints. One may thus consider that the enabling of a common action is constrained by the various individual timing requirements. For example, in $a\,;\,(3)\,c\,;\,\mathbf{0}$ and $b\,;\,(7)\,c\,;\,\mathbf{0}$, action c is enabled in the composite behaviour

$$a\,;\,(3)\,c\,;\,\mathbf{0}\,||_c\, b\,;\,(7)\,c\,;\,\mathbf{0} \quad ,$$

if both a has occurred at least 3 time units before and b has occurred at least 7 time units before, that is, $t_c \geqslant t_a{+}3 \;\wedge\; t_c \geqslant t_b{+}7$ which is equivalent to $t_c \geqslant max(t_a{+}3, t_b{+}7)$. Using a similar reasoning, in behaviour

$$a\,;\,(t_1)\,b\,;\,\mathbf{0}\,||_{\{a,b\}}\, a\,;\,(t_2)\,b\,;\,\mathbf{0} \quad ,$$

b is enabled after $max(t_a{+}t_1, t_a{+}t_2) = t_a + max(t_1, t_2)$.

Intuitively it means that a system which is willing to participate in some action a from time t say, has to wait until the environment is ready for participation. The integrated behaviour of the system and the environment may then execute a from the moment on that both the system and the environment are willing to perform a.

4.3.2 Causality-based semantics

We now show how the timed event structures of Section 4.2 can be used as a vehicle to provide a true concurrency semantics to PA_T in a compositional way. We do so by defining a mapping $\mathcal{E}_T[\![\ ]\!] : \mathsf{PA}_T \longrightarrow \mathsf{EBES}_T$. In addition we use:

4.27. Definition. $\Phi_T : \mathsf{PA}_T \longrightarrow \mathsf{PA}$ is defined as follows:

$$\begin{array}{rcl}
\Phi_T(\mathbf{0}) & \triangleq & \mathbf{0} \\
\Phi_T(\surd) & \triangleq & \surd \\
\Phi_T((t)\ a\,;\,B) & \triangleq & a\,;\,\Phi_T(B) \\
\Phi_T(B_1 \ \mathsf{op}\ B_2) & \triangleq & \Phi_T(B_1)\ \mathsf{op}\ \Phi_T(B_2) \text{ for } \mathsf{op} \in \{\,+,\ ||_G\,, >\!\!>, [>\,\} \\
\Phi_T(\mathsf{op}\ B) & \triangleq & \mathsf{op}\ \Phi_T(B) \text{ for } \mathsf{op} \in \{\,\backslash, [\,]\,\}.
\end{array}$$

□

So, Φ_T associates to a timed behaviour B its corresponding untimed behaviour $\Phi_T(B)$ by simply omitting all time annotations in B.

The positive events of Γ are the events in Γ with a non-zero delay.

4.28. Definition. For $\Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ the set $pos(\Gamma)$ of positive events is defined by

$$pos(\Gamma) \triangleq \{\, e \in E \mid \mathcal{D}(e) \neq 0\,\} \quad .$$

□

In the rest of this section let $\mathcal{E}_T[\![B_i]\!] = \Gamma_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i \rangle$, for $i = 1, 2$, with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ and $E_1 \cap E_2 = \varnothing$. The functions *init* and *exit* which denote the set of initial and successful termination events, respectively, are defined in Chapter 2 for event structures and are used for timed event structures in the same way. Let $pin(\Gamma) \triangleq pos(\Gamma) \cup init(\Gamma)$ and E_U the universe of events. For convenience we use the denotational semantics $\mathcal{E}'[\![\ ]\!]$ for the untimed case which is defined in Chapter 2. This becomes explicit for timed action-prefix and enabling; for these constructs it is indicated which instantiation of $\mathcal{E}'[\![\ ]\!]$ is chosen.

4.29. DEFINITION. (*Timed semantics of* **0**, $\surd$, *and* $(t)\ a\,;$)

$$
\begin{array}{rcl}
\mathcal{E}_T[\![\mathbf{0}]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi_T(\mathbf{0})]\!], \varnothing, \varnothing \rangle \\
\mathcal{E}_T[\![\surd]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi_T(\surd)]\!], \{ (e_\delta, 0) \}, \varnothing \rangle \\
\mathcal{E}_T[\![(t)\ a\,;\, B_1]\!] & \triangleq & \langle (E, \rightsquigarrow_1, \mapsto, l_1 \cup \{ (e_a, a) \}), \mathcal{D}, \mathcal{T} \rangle \text{ where} \\
E & = & E_1 \cup \{ e_a \} \text{ for some } e_a \in E_U \setminus E_1 \\
\mapsto & = & \mapsto_1 \cup (\{ \{ e_a \} \} \times pin(\Gamma_1)) \\
\mathcal{D} & = & \{ (e_a, t) \} \cup (E_1 \times \{ 0 \}) \\
\mathcal{T} & = & \mathcal{T}_1 \cup \{ ((\{ e_a \}, e), \mathcal{D}_1(e)) \mid e \in pin(\Gamma_1) \}.
\end{array}
$$

□

The semantics of **0** and $\surd$ is self-explanatory. In $\mathcal{E}_T[\![(t)\ a\,;\, B_1]\!]$ a bundle is introduced from a new event e_a (labelled a) to all initial events in Γ_1 and, in addition, to all events in Γ_1 that have a non-zero delay. The delay of these events e becomes relative to e_a, so each bundle $\{ e_a \} \mapsto e$ is associated with a time delay $\mathcal{D}_1(e)$, and $\mathcal{D}(e)$ is made zero. Delay $\mathcal{D}(e_a)$ is set to t. In the untimed case it suffices to only introduce bundles from e to the initial events of Γ_1, cf. Definition 2.35. The additional bundles to the positive events of Γ_1 that are introduced in the timed case are used for the sole purpose of making delays relative to e_a. For events that have a zero delay this is not necessary; they can happen from any moment since the start of the system.

4.30. EXAMPLE. Figure 4.5 depicts (a) $\mathcal{E}_T[\![B]\!]$, and (b) $\mathcal{E}_T[\![(2)\ a\,;\, B]\!]$. The reader is invited to compare these figures with Figure 2.5. □

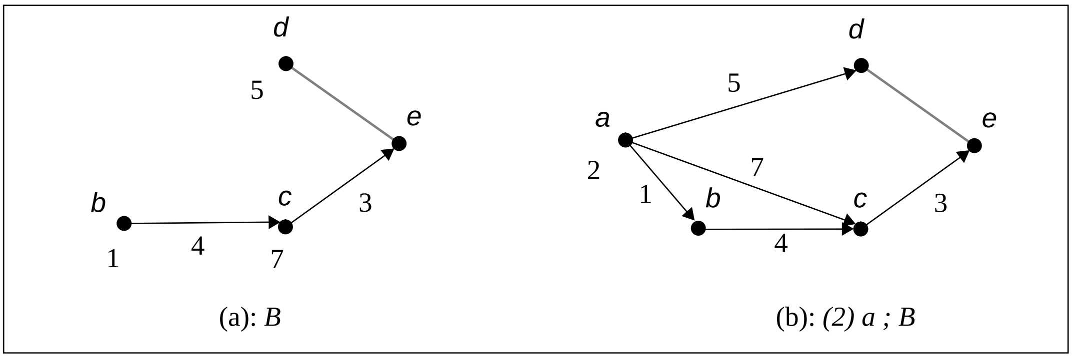


Figure 4.5: Example of semantics for timed action prefix.

4.31. DEFINITION. (*Timed semantics of* $\backslash$, $[\,]$, $+$, $>>$ *and* $[>$)

$$\begin{array}{rcl}
\mathcal{E}_T[\![\, B_1 \ \mathsf{op}\ B_2 \,]\!] & \triangleq & \langle \mathcal{E}'[\![\, \Phi_T(B_1 \ \mathsf{op}\ B_2) \,]\!], \mathcal{D}_1 \cup \mathcal{D}_2, \mathcal{T}_1 \cup \mathcal{T}_2 \rangle,\ \mathsf{op}\ \in \{ +, [> \} \\
\mathcal{E}_T[\![\, \mathsf{op}\ B_1 \,]\!] & \triangleq & \langle \mathcal{E}'[\![\, \Phi_T(\mathsf{op}\ B_1) \,]\!], \mathcal{D}_1, \mathcal{T}_1 \rangle \text{ for } \mathsf{op}\ \in \{ \backslash, [\,] \} \\
\mathcal{E}_T[\![\, B_1 >> B_2 \,]\!] & \triangleq & \langle (E_1 \cup E_2, \rightsquigarrow, \mapsto, l), \mathcal{D}, \mathcal{T} \rangle \text{ where} \\
\rightsquigarrow & = & \rightsquigarrow_1 \ \cup\ \rightsquigarrow_2 \ \cup \{ (e, e') \mid e, e' \in exit(\Gamma_1) \ \wedge\ e \neq e' \} \\
\mapsto & = & \mapsto_1 \ \cup\ \mapsto_2 \ \cup (\{ exit(\Gamma_1) \} \times pin(\Gamma_2)) \\
l & = & ((l_1 \cup l_2) \setminus (exit(\Gamma_1) \times \{ \delta \})) \cup (exit(\Gamma_1) \times \{ \tau \}) \\
\mathcal{D} & = & \mathcal{D}_1 \cup (E_2 \times \{ 0 \}) \\
\mathcal{T} & = & \mathcal{T}_1 \cup \mathcal{T}_2 \cup \{ ((exit(\Gamma_1), e), \mathcal{D}_2(e)) \mid e \in pin(\Gamma_2) \}.
\end{array}$$

□

For op equal to choice or disrupt $\mathcal{E}_T[\![\, B_1 \ \mathsf{op}\ B_2 \,]\!]$ is the untimed event structure of the corresponding expression in PA, $\mathcal{E}'[\![\, \Phi_T(B_1 \ \mathsf{op}\ B_2) \,]\!]$, where the timings of events and bundles in Γ_1 and Γ_2 are unaffected. Similarly, $\mathcal{E}_T[\![\ ]\!]$ is defined for relabelling and hiding.

$\mathcal{E}_T[\![\, B_1 >> B_2 \,]\!]$ is equal to $\Gamma_1 \cup \Gamma_2$ where bundles are introduced between the successful termination events of Γ_1 and the initial and positive events in Γ_2. The reason for introducing bundles to the positive events of Γ_2 is to make the event delays in Γ_2 relative to the termination of Γ_1. This is similar as for timed action-prefix. The timing of the introduced bundles and the positive events in Γ_2 are treated in a similar way as for timed action-prefix.

4.32. EXAMPLE. Let Figure 4.6(a) and (b) depict $\mathcal{E}_T[\![\, B_1 \,]\!]$ and $\mathcal{E}_T[\![\, B_2 \,]\!]$, respectively. $\mathcal{E}_T[\![\, B_1 >> B_2 \,]\!]$ and $\mathcal{E}_T[\![\, B_1 [> B_2 \,]\!]$ are depicted in Figure 4.6(c) and (d), respectively. The reader is invited to compare this figure with Figure 2.6. □

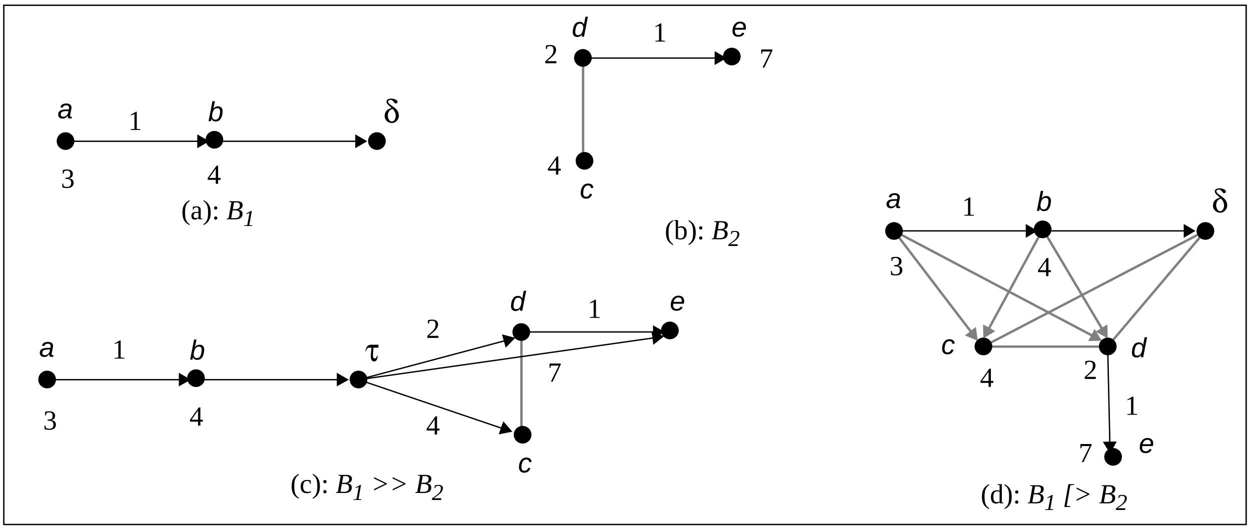


Figure 4.6: Example of semantics for enable and disrupt.

Finally, we explain the timed components of the semantics of the parallel composition operator. Recall that events are pairs of events of Γ_1 and Γ_2, or with one component equal to $*$. The delay of an event is the maximum of the delays of its components that are different from $*$. The time associated with a bundle is equal to the maximum of the times associated with the bundles we get by projecting on the i-th components (i=1, 2) of the events in the bundle, if this projection yields a bundle in Γ_i.

As a subsidiary notion we define projection of bundles as follows:

4.33. Definition. Let $E = (E_1 \cup \{*\}) \times (E_2 \cup \{*\}), (e_1, e_2) \in E$ and $X \subseteq E$. Let

- $\mathsf{pr}_i((e_1, e_2)) \triangleq e_i$, if $e_i \neq *$, for i=1, 2
- $\mathsf{pr}_i(X) \triangleq \{\mathsf{pr}_i(e) \mid e \in X \cap dom(\mathsf{pr}_i)\}$, for i=1, 2.

□

4.34. Definition. (*Timed semantics of* $||_G$)

$$\begin{array}{rcl} \mathcal{E}_T[\![B_1 \,||_G\, B_2]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi_T(B_1 \,||_G\, B_2)]\!], \mathcal{D}, \mathcal{T} \rangle \text{ where} \\ \mathcal{D}((e_1, e_2)) & = & max(\mathcal{D}_1(e_1), \mathcal{D}_2(e_2)) \text{ with } \mathcal{D}_i(*) = 0. \\ \mathcal{T}((X, (e_1, e_2))) & = & max(\mathcal{T}_1((pr_1(X), e_1)), \mathcal{T}_2((pr_2(X), e_2))) \\ & & \text{with } \mathcal{T}_i((\varnothing, e_i)) = 0, \text{ for } i{=}1,2. \end{array}$$

□

4.35. Example. Consider the following timed behaviours

$$\begin{array}{rcl} B_1 & = & (1)\ a\,;\ (5)\ b\,;\ \mathbf{0}\,||_b\,(4)\ c\,;\ (7)\ b\,;\ \mathbf{0} \\ B_2 & = & (4)\ a\,;\ (2)\ b\,;\ \mathbf{0}\,||_b\,((4)\ b\,;\ \mathbf{0} + (3)\ d\,;\ \mathbf{0}) \quad . \end{array}$$

Figure 4.7 shows how $\mathcal{E}_T[\![B_1 \,||_{\{a,b\}}\, B_2]\!]$ is constructed from $\mathcal{E}_T[\![B_1]\!]$ and $\mathcal{E}_T[\![B_2]\!]$. For example, $\mathcal{D}(e_a) = max(1, 4)$, $\mathcal{T}((\{e_a\}, e_b)) = max(5, 2)$, and $\mathcal{T}((\{e_c\}, e_b)) = max(7, 0)$.

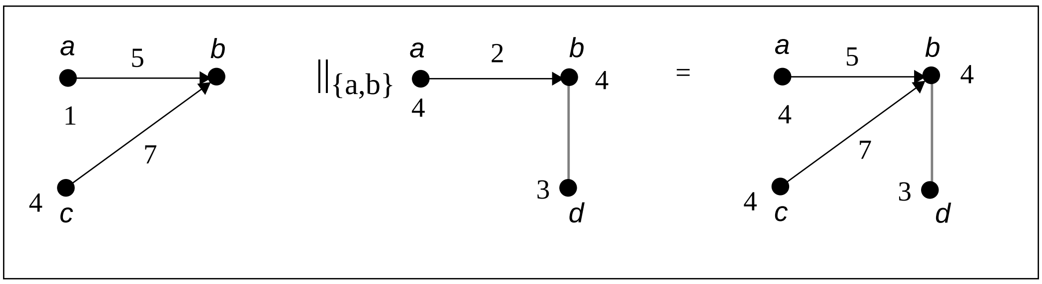


Figure 4.7: Example of semantics for parallel composition (I).

Figure 4.8 shows the timed event structures corresponding to the following behaviours:

$$\begin{array}{ll} (a) & ((2)\ a\,;\ (3)\ d\,;\ \mathbf{0} + (1)\ b\,;\ (2)\ e\,;\ \mathbf{0})\,|||\,(27)\ c\,;\ \mathbf{0} \\ (b) & ((2)\ a\,;\ (7)\ c\,;\ \mathbf{0} + (4)\ a\,;\ (11)\ d\,;\ \mathbf{0})\,||_a\,((5)\ a\,;\ (2)\ b\,;\ \mathbf{0}) \\ (c) & (2)\ a\,;\ (1)\ b\,;\ \mathbf{0}\,||_{\{a,b\}}\,(7)\ b\,;\ \mathbf{0} \quad . \end{array}$$

□

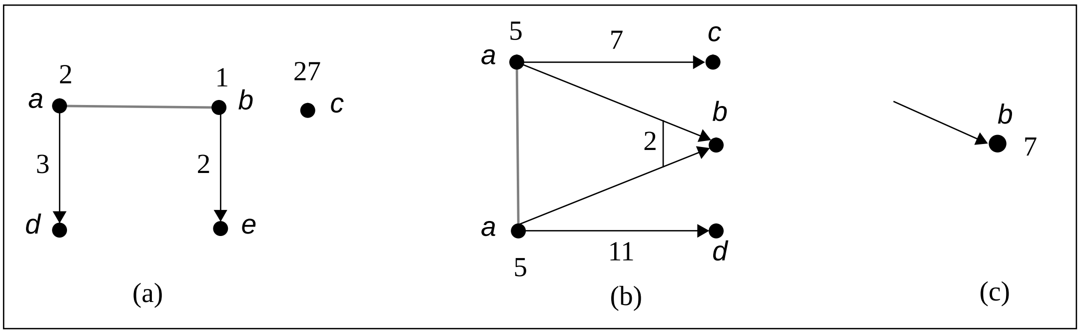


Figure 4.8: Example of semantics for parallel composition (II).

The definition of $\mathcal{E}_T[\![P]\!]$ where $P := B$ can be defined by an extension of the untimed case and is fully treated in Chapter 10 of this dissertation.

The timed extension of behaviours is "backward compatible" with the untimed case, in the following sense. For an expression $B \in \mathsf{PA}_T$ the lposets that are obtained by removing the times from $L_T(\Gamma)$ where Γ is the denotational semantics of B, i.e., $\Gamma = \mathcal{E}_T[\![B]\!]$, are equal to the lposets obtained from the event structure corresponding to $\Phi_T(B)$, the untimed counterpart of B. This means that causal dependencies are unaffected by the timed components in PA_T.

4.36. THEOREM. *Compatibility theorem*

$$\forall\, B \in \mathsf{PA}_T : L(\mathcal{E}_T[\![B]\!]) = L(\mathcal{E}[\![\Phi_T(B)]\!]).$$

PROOF. We derive:

$$\begin{array}{ll} & L(\mathcal{E}_T[\![B]\!]) \\ = & \{ \text{ Theorem 4.21 } \} \\ & L(\varphi(\mathcal{E}_T[\![B]\!])) \\ = & \{ \mathcal{E}_T[\![B]\!] = \langle \mathcal{E}'[\![\Phi_T(B)]\!], \mathcal{D}, \mathcal{T} \rangle \} \\ & L(\mathcal{E}'[\![\Phi_T(B)]\!]) \\ = & \{ \text{ Theorem 2.44 } \} \\ & L(\mathcal{E}[\![\Phi_T(B)]\!]) \quad . \end{array}$$

□

4.37. THEOREM. $\forall\, B \in \mathsf{PA}_T : \mathcal{E}_T[\![B]\!] \in \mathsf{EBES}_T$.

PROOF. Simply by the fact that $\mathcal{E}_T[\![B]\!] = \langle \mathcal{E}'[\![\Phi_T(B)]\!], \mathcal{D}, \mathcal{T} \rangle$ and the fact that $\mathcal{E}'[\![\Phi_T(B)]\!] \in \mathsf{EBES}$. It is easy to check from the definition of $\mathcal{E}_T[\![\ ]\!]$ that $\mathcal{T}$ and $\mathcal{D}$ associate time values to all bundles and events, respectively, in $\mathcal{E}_T[\![B]\!]$. □

$\mathcal{E}_T[\![B]\!]$ can successfully terminate as soon as all events causally preceding a successful termination event have happened.

4.38. LEMMA. For $B \in \mathsf{PA}_T$ let $\mathcal{E}_T[\![B]\!] = \Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$. Then

1. $\forall\, e \in \mathit{exit}(\Gamma) : \mathcal{D}(e) = 0$
2. $\forall\, X \subseteq E : X \overset{t}{\mapsto} e \ \wedge \ e \in \mathit{exit}(\Gamma) \ \Rightarrow \ t = 0.$

PROOF. Straightforward by induction on the structure of B. Routine and omitted. □

4.3.3 Syntactic conditions for simplification

In this section we investigate under which syntactic conditions the timed event structure model can be simplified. More specifically we aim at a constraint on behaviour expressions such that event delays become superfluous and thus can be omitted from the model.

As an auxiliary notion we define (syntactically) the set of initial actions of B which B cannot perform immediately.

4.39. DEFINITION. (*Non-immediate initial actions*)

$\mathsf{nii} : \mathsf{PA}_T \longrightarrow \mathcal{P}(\mathsf{Act}^{\tau,\delta})$ is defined as follows:

$$\begin{array}{rcl}
\mathsf{nii}(\mathbf{0}) & \triangleq & \varnothing \\
\mathsf{nii}(\surd) & \triangleq & \varnothing \\
\mathsf{nii}((t)\ a\,;\,B) & \triangleq & \begin{cases} \{\,a\,\} & \text{if } t > 0 \\ \varnothing & \text{if } t = 0 \end{cases} \\
\mathsf{nii}(B_1 + B_2) & \triangleq & \mathsf{nii}(B_1) \cup \mathsf{nii}(B_2) \\
\mathsf{nii}(B \setminus G) & \triangleq & (\mathsf{nii}(B) \setminus G) \cup \{\,\tau \mid \mathsf{nii}(B) \cap G \neq \varnothing\,\} \\
\mathsf{nii}(B[H]) & \triangleq & \{\,H(a) \mid a \in \mathsf{nii}(B)\,\} \\
\mathsf{nii}(B_1 >> B_2) & \triangleq & (\mathsf{nii}(B_1) \setminus \{\,\delta\,\}) \cup \{\,\tau \mid \delta \in \mathsf{nii}(B_1)\,\} \\
\mathsf{nii}(B_1\,[> B_2) & \triangleq & \mathsf{nii}(B_1) \cup \mathsf{nii}(B_2) \\
\mathsf{nii}(B_1 \,||_G\, B_2) & \triangleq & (\mathsf{nii}(B_1) \cup \mathsf{nii}(B_2)) \setminus G^\delta \cup (\mathsf{nii}(B_1) \cap \mathsf{nii}(B_2) \cap G^\delta).
\end{array}$$

□

Note that successful termination events can always be executed immediately, so $\delta \notin \mathsf{nii}(B)$, for all $B \in \mathsf{PA}_T$.

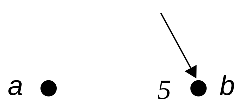
The following lemma shows that $\mathsf{nii}(B)$ indeed characterizes the set of initial actions of B that cannot be performed immediately.

4.40. LEMMA. For $B \in \mathsf{PA}_T$ with $\Gamma = \mathcal{E}_T[\![\,B\,]\!] = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$:

$$\mathsf{nii}(B) = \{\,l(e) \mid e \in \mathit{init}(\Gamma)\ \wedge\ \mathcal{D}(e) \neq 0\,\}.$$

PROOF. By induction on the structure of B; the proof is quite straightforward but somewhat elaborative. □

We now concentrate on a syntactic constraint under which event delays become superfluous, for instance, by enforcing that all event delays are 0. At first sight it seems that all events in $\mathcal{E}_T[\![\,B\,]\!]$ have delay 0 iff all initial actions of the corresponding behaviour, B, have delay 0 (i.e., $\mathsf{nii}(B) = \varnothing$). Due to the fact that synchronization can give rise to empty bundles pointing to events (see also Chapter 2) this conjecture is, however, not true. Consider



which corresponds to $a \,|||\, (c\,;\,(2)\ b) \,||_{\{b,c\}}\, (5)\ b$. Obviously, this structure violates the aforementioned conjecture—initial actions a and c have delay 0 but event e_b in the resulting timed event structure has a non-zero delay. The problem is that synchronization is required on an initial action (i.e., c) of one of the components in $||_G$ which does not succeed.

To avoid such cases we require that all parallel compositions $B_1 \,||_G\, B_2$ occurring as subexpression in B satisfy $\mathsf{nii}(B_1) \cap G^\delta = \mathsf{nii}(B_2) \cap G^\delta$. This guarantees that B_1 and B_2 are both able to participate on the same initial actions in G^δ. Let PA_T^* denote the set of expressions in PA_T that satisfy this syntactic constraint. Then we have that for $B \in \mathsf{PA}_T^*$ all events in $\mathcal{E}_T[\![\, B \,]\!]$ have delay 0 iff all initial actions of B have delay 0. This implies that our timed event structures model can be simplified, only having bundle delays and omitting the event delays, once this (syntactical) condition is met.

4.41. LEMMA. $\forall\, B \in \mathsf{PA}_T^* : \mathsf{nii}(B) = \varnothing \iff pos(\mathcal{E}_T[\![\, B \,]\!]) = \varnothing.$

PROOF. '$\Leftarrow$': Straightforward from Lemma 4.40 and omitted.
'$\Rightarrow$' : By induction on the structure of B.
Base: For $B = \mathbf{0}$ and $B = \surd$ the theorem trivially holds as $\mathsf{nii}(\mathbf{0}) = \varnothing$ and $\mathsf{nii}(\surd) = \varnothing$, and all events in $\mathcal{E}_T[\![\, B \,]\!]$ have delay 0. For $B = (t)\ a\,;\,B_1$ we have $\mathsf{nii}(B) = \{\,a\,\}$ if $t = 0$. But then $\mathcal{D}(e_a) = t = 0$, and for all other events $\mathcal{D}(e) = 0$ (cf. definition of $\mathcal{E}_T[\![\ ]\!]$).
Induction Step: Assume the lemma holds for B_1 and B_2. Let $\Gamma_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i \rangle = \mathcal{E}_T[\![\, B_i \,]\!]$, for i=1, 2 with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$. We provide proofs for abstraction, enabling and parallel composition. The proofs for the other constructs are quite similar and omitted here.

1. $B = B_1 \setminus G$. We prove that

$$
\begin{array}{ll}
& \mathsf{nii}(B_1 \setminus G) = \varnothing \\
\Leftrightarrow & \{\ \text{Definition 4.39}\ \} \\
& \mathsf{nii}(B_1) \setminus G = \varnothing \ \wedge\ \{\, \tau \mid \mathsf{nii}(B_1) \cap G \neq \varnothing \,\} = \varnothing \\
\Leftrightarrow & \{\ \} \\
& \mathsf{nii}(B_1) \setminus G = \varnothing \ \wedge\ \mathsf{nii}(B_1) \cap G = \varnothing \\
\Leftrightarrow & \{\ \} \\
& \mathsf{nii}(B_1) = \varnothing \\
\Rightarrow & \{\ \text{induction hypothesis}\ \} \\
& pos(\mathcal{E}_T[\![\, B_1 \,]\!]) = \varnothing \\
\Leftrightarrow & \{\ \text{definition of } \mathcal{E}_T[\![\ ]\!]\ \} \\
& pos(\mathcal{E}_T[\![\, B_1 \setminus G \,]\!]) = \varnothing \quad .
\end{array}
$$

2. $B = B_1 >> B_2$. For this case we derive:

$$
\begin{array}{ll}
& \mathsf{nii}(B_1 >> B_2) = \varnothing \\
\Leftrightarrow & \{\ \text{Definition 4.39}\ \} \\
& \mathsf{nii}(B_1) \setminus \{\,\delta\,\} = \varnothing \ \wedge\ \{\, \tau \mid \delta \in \mathsf{nii}(B_1) \,\} = \varnothing \\
\Leftrightarrow & \{\ \} \\
& \mathsf{nii}(B_1) = \varnothing \\
\Rightarrow & \{\ \text{induction hypothesis}\ \}
\end{array}
$$

$$\begin{aligned}
& pos(\mathcal{E}_T[\![\, B_1 \,]\!]) = \varnothing \\
\Leftrightarrow \quad & \{ \text{ definition of } \mathcal{E}_T[\![\;]\!] \ \} \\
& pos(\mathcal{E}_T[\![\, B_1 >> B_2 \,]\!]) = \varnothing \quad .
\end{aligned}$$

3. $B = B_1 \,||_G\, B_2$. For this case we infer:

$$\begin{aligned}
& \mathsf{nii}(B_1 \,||_G\, B_2) = \varnothing \\
\Leftrightarrow \quad & \{ \text{ Definition 4.39 } \} \\
& \mathsf{nii}(B_1) \setminus G^\delta \cup \mathsf{nii}(B_2) \setminus G^\delta \cup (\mathsf{nii}(B_1) \cap \mathsf{nii}(B_2) \cap G^\delta) = \varnothing \\
\Leftrightarrow \quad & \{ \ A \setminus C \cup B \setminus C \cup (A \cap B \cap C) = A \setminus C \cup B \setminus C \cup (A \cap B) \ \} \\
& \mathsf{nii}(B_1) \setminus G^\delta \cup \mathsf{nii}(B_2) \setminus G^\delta \cup (\mathsf{nii}(B_1) \cap \mathsf{nii}(B_2)) = \varnothing \\
\Leftrightarrow \quad & \{ \ B \in \mathsf{PA}_T^*;\ A \setminus C \cup B \setminus C \cup (A \cap B) = A \cup B \text{ if } A \cap C = B \cap C \ \} \\
& \mathsf{nii}(B_1) \cup \mathsf{nii}(B_2) = \varnothing \\
\Rightarrow \quad & \{ \text{ induction hypothesis } \} \\
& pos(\mathcal{E}_T[\![\, B_1 \,]\!]) = \varnothing \ \wedge \ pos(\mathcal{E}_T[\![\, B_2 \,]\!]) = \varnothing \\
\Rightarrow \quad & \{ \text{ definition } \mathcal{E}_T[\![\;]\!] \ \} \\
& pos(\mathcal{E}_T[\![\, B_1 \,||_G\, B_2 \,]\!]) = \varnothing \quad .
\end{aligned}$$

$\square$

4.4 Conclusions

In this chapter we have presented a simple timed extension of extended bundle event structures that allows the specification of minimal time constraints. The theory of extended bundle event structures is carried over to the timed setting in a rather smooth way—notions like timed event trace and timed remainder are straightforward conservative extensions of their untimed counterparts.

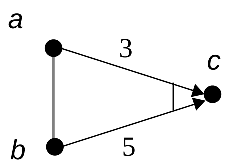
One of the features of the model is the absence of actions that represent the passage of time, which in one way or another make their appearance in most interleaving models (see also Chapter 5). Here, time is dealt with in a way comparable to ordinary physical models, viz. by means of parameterization (e.g., for recording the delays). Another important feature of the timed model is that it is a conservative extension of the untimed case; the causal dependencies present in the untimed model are unaffected by the inclusion of minimal time constraints. This stems from the fact that events do not become impossible by imposing minimal time constraints. In Chapters 6 and 7 we will encounter timed extensions which violate backward compatibility.

The timed model in this chapter is kept rather simple—expressiveness was not our first main goal. The incorporation of urgency in the simple timed model of this chapter is dealt with in Chapter 6. In Chapter 7 we will investigate how the theory of this chapter can be generalized by allowing intervals (or even sets) of time instants to be associated with events and bundles; in this chapter we also compare our approach with existing timed extensions of partial-order models.

From several perspectives it would be interesting to elaborate the timed model in even other directions, some of which are mentioned below:

- Associate time with the asymmetric conflict relation. The intuitive meaning of $e_a \stackrel{t}{\rightsquigarrow} e_b$ is that (i) if e_b occurs it disables the occurrence of e_a (as in the untimed case), and (ii) if e_a and e_b both occur in a single system run then e_a causally precedes e_b (as in the untimed case) and the minimal time between the enabling of e_b and the occurrence of e_a is t. When in addition $e_c \stackrel{t'}{\rightsquigarrow} e_b$ and all three events e_a, e_b and e_c occur in a single run, then e_b is enabled at $max(t_a{+}t, t_c{+}t')$. Note that $\rightsquigarrow = \stackrel{0}{\rightsquigarrow}$. The extension of the model with this construct is fairly straightforward.

- In the current model time is associated with bundles. E.g., when $\{e_a, e_b\} \stackrel{t}{\mapsto} e_c$ the minimal relative time between e_c and either of its causal predecessors is equal to t. An alternative would be to allow for the association of different time values to the different 'branches' of the bundle. For instance,

  intuitively specifies that (i) if e_c and e_a occur in a run then the minimal timing between the occurrences of these events equals 3 and (ii) if e_b and e_c occur then this time is 5. We believe that also this construct can be added to our model in a reasonably straightforward way.

- The previous construct can also be used as a basis to add time to one of the primitives in our model of Chapter 3, disjunctive causality. Even adding time to the interleaving relation of that model could make sense. The interpretation of $e_a \;{}^{t}{\rightleftharpoons}_{t'}\; e_b$ is that e_a and e_b are interleaved, e_b being caused by e_a after a minimal delay of t' time units, or e_a is being caused by e_b after a minimal delay of t time units. This subject is left for further study.

5 Timed operational semantics

This chapter presents two timed event transition systems for the timed process algebra PA_T. Opposed to the standard case transitions are equipped with event and action (and time) labels. The timed event transition systems are defined by structured operational semantics. One transition model is based on timed-action transitions and the other is based on the separation between time- and (untimed) action-transitions. The compatibility of these timed transition models with the causality-based semantics of PA_T as provided in Chapter 4 is investigated. The timed event traces of the timed-action transition model and the causality-based semantical model are shown to coincide. For the model distinguishing between time- and action-transitions this holds when restricting to time-consistent traces.

5.1 Introduction

In Chapter 4 we have presented a causality-based semantics for a temporal variant of the process algebra PA. The basic timing ingredient in PA_T is a delay function that specifies the minimal relative delay of an action with respect to its causal predecessors (if any). This chapter presents an event-based operational semantics for this formalism in two ways and shows that these operational semantical models are compatible with the causality-based semantics of PA_T.

If we are mainly interested in a causality-based semantics why do we have to define an operational semantics as well? This understandable question can be answered adequately as follows. First of all, a rather 'standard' means to provide a semantics to process algebras, let alone timed variants thereof, is to present an operational semantics. By providing an operational view on our timed event structure semantics we facilitate a comparison with existing approaches. Various timed extensions of process algebras have been (and still are being) proposed in the literature based on timed variants of labelled transition systems. Since there is no canonical way to include time into transition systems different approaches appear. A (timed) event-based operational semantics for PA_T provides a basis to determine our position in this broad field.

Secondly, like for interleaving semantics of timed formalisms there are various ways in which a partial-order semantics can be defined for such formalisms. A natural demand is that the partial-order semantics is compatible with less discriminating semantics such as pomset, step and interleaving semantics. This has been well-recognized in the literature. Langerak, for instance, shows that his event structure semantics of LOTOS is compatible with the standard interleaving semantics of LOTOS [89], Boudol & Castellani [23, 26] consider the compatibility between a flow event structure and interleaving semantics of CCS, and Baier & Majster-Cederbaum [10] prove the consistency between a prime event structure and interleaving semantics of theoretical CSP, extending the results of a previous attempt by Loogen & Goltz [95].

These studies are all performed in an untimed setting. A problem, compared to the untimed case, is that there is no consensus on how to include time into a transition system and, as a consequence, different styles have been developed for providing an operational semantics for timed process algebras. This chapter concentrates on, what we consider to be, the two major schools in timed interleaving models—models that explicitly distinguish between time-advancing transitions and the occurrence of 'normal' actions, and models that do not and combine these two notions into a single transition relation.

The difference between *timed-action* and *time-* and *action* transition systems can best be understood by means of a simple example. In the timed-action model (Figure 5.1(a)) transitions

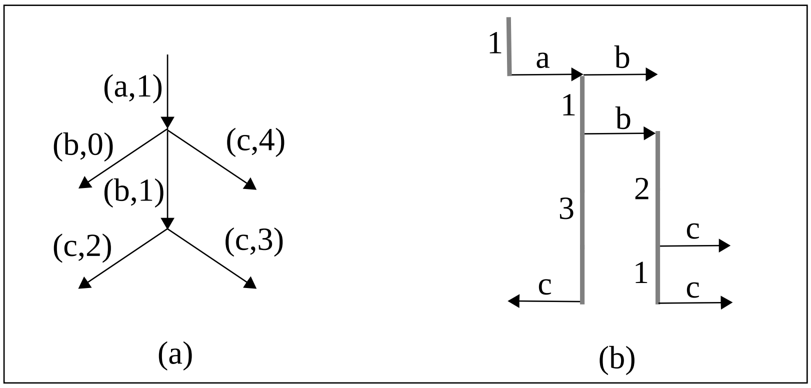


Figure 5.1: Timed-action transitions versus time- and action transitions.

are labelled with timed actions and the passage of time is not explicitly modelled. In time- and action-transition systems (Figure 5.1(b)) the passage of time is modelled explicitly (depicted vertically) and action transitions (depicted horizontally) are untimed. Action transitions are orthogonal to time transitions and the projection of action transitions on the time axis has zero length, indicating that actions consume no time.

The approach followed in this chapter is adopted from [89, Chapter 7]; the same scheme is used by Rensink [127] to obtain an operational semantics for a process algebraic formalism including a refinement operator. Since our timed variant of extended bundle event structures is in fact just a parameterization of this model we might expect that we can quite closely follow this approach. The approach—inspired by [23, 26]—embodies defining a *timed event transition system*, which is a transition system in which we keep track of action occurrences (i.e., events) rather than the actions themselves (as usual in structured operational semantics), and showing that this transition system generates the same set of timed event traces as the causality-based semantics.

As argued above we concentrate on two types of timed interleaving models. This results in *two* timed event-based operational semantics for PA_T.

In the first part of this chapter we consider a timed model for PA_T based on timed-action transitions. This model turns out to be a straightforward (and minimal) extension of the untimed event transition system of Chapter 2—by just omitting the time labels in each inference rule the untimed event transition system for PA is obtained. The resulting event-based operational semantics is fully compatible to the causality-based semantics of Chapter 4 in the sense that it generates the same set of timed traces, and, since timed event traces can be used to deduce lposets, it generates the same set of lposets.

In Section 5.4 we distinguish between time-transitions (denoted $\rightsquigarrow$) and action-transitions (denoted $\longrightarrow$). This gives rise to a transition system which is an orthogonal extension of the untimed event transition system for PA (of Chapter 2) in the sense that the rules for $\longrightarrow$ are identical to the untimed event-based inference rules. Thus, time is indeed considered as an orthogonal dimension of the untimed model. The model forces derivations to be time-consistent, and therefore is only partially compatible to the causality-based semantics of Chapter 4—it generates the same set of time-consistent traces.

In more detail this chapter is organized as follows. Section 5.2 defines an operational semantics for PA_T. Section 5.3 considers the consistency between the operational and causality-based semantics of PA_T at the level of timed event transition systems. The alternative approach with separate time- and action-transitions is presented in Section 5.4. The consistency between the alternative model and the denotational semantics is studied in Section 5.5. Model properties like time determinism, action persistency, and time additivity—properties of timed transition systems that are commonly considered to be of importance, see Nicollin & Sifakis [112]—are considered in Section 5.6. Finally, Section 5.7 discusses some related work and Section 5.8 draws conclusions.

In this chapter we confine ourselves (as in the previous chapters) to finite behaviours; event-based operational semantics for $P := B$ where B might contain occurrences of P is dealt with in Chapter 10.

5.2 Event-based operational semantics for PA_T

In this section we present an operational semantics, for PA_T, the simple timed process algebra of Chapter 4. This semantics is defined by means of inference rules (in the style of Plotkin [120]) that determine a timed event transition relation. We follow the procedure of Section 2.5.

Event identities are generated by annotating each action occurrence in term B with a unique event occurrence identifier, denoted by a Greek letter. Recall that for parallel composition new event names can be created. If e is an event name of B and e' an event name in B', then possible new event names in $B \,||_G\, B'$ are $(e, *)$ and $(*, e')$ for unsynchronized events and (e, e') for synchronized events.

The operational semantics defines a set of transition relations $\xrightarrow{(e,a,t)}\!\!\!\!\rightarrow$. $B \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B'$ denotes that behaviour B can perform event $e \in Ev$, labelled with action $a \in \mathsf{Act}^{\tau,\delta}$, at time $t \in \mathsf{Time}$, and subsequently evolve into B'. The transition relation $\longrightarrow\!\!\!\!\rightarrow$ is the smallest relation closed under all inference rules in Table 5.1.

$$\frac{}{\surd_\xi \xrightarrow{(\xi,\delta,t)}\!\!\!\!\rightarrow \mathbf{0}}$$

$$\frac{}{(t)\, a_\xi \,;\, B \xrightarrow{(\xi,a,t')}\!\!\!\!\rightarrow {}^{t'}[\, B\,]} \quad (t' \geqslant t) \qquad \frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{{}^{t'}[\, B\,] \xrightarrow{(\xi,a,t+t')}\!\!\!\!\rightarrow {}^{t'}[\, B'\,]}$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1'}{B_1 + B_2 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1'} \qquad \frac{B_2 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_2'}{B_1 + B_2 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_2'}$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1'}{B_1 >> B_2 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1' >> B_2} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{(\xi,\delta,t)}\!\!\!\!\rightarrow B_1'}{B_1 >> B_2 \xrightarrow{(\xi,\tau,t)}\!\!\!\!\rightarrow {}^{t}[\, B_2\,]}$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1'}{B_1 \,[> B_2 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1' \,[> {}^{t}\{\, B_2\,\}} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{(\xi,\delta,t)}\!\!\!\!\rightarrow B_1'}{B_1 \,[> B_2 \xrightarrow{(\xi,\delta,t)}\!\!\!\!\rightarrow B_1'}$$

$$\frac{B_2 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_2'}{B_1 \,[> B_2 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_2'} \qquad \frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{{}^{t'}\{\, B\,\} \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow {}^{t'}\{\, B'\,\}} \quad (t \geqslant t')$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,*),a,t)}\!\!\!\!\rightarrow B_1' \,||_G\, B_2} \quad (a \notin G^\delta) \qquad \frac{B_2 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((*,\xi),a,t)}\!\!\!\!\rightarrow B_1 \,||_G\, B_2'} \quad (a \notin G^\delta)$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1' \ \wedge \ B_2 \xrightarrow{(\psi,a,t)}\!\!\!\!\rightarrow B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,\psi),a,t)}\!\!\!\!\rightarrow B_1' \,||_G\, B_2'} \quad (a \in G^\delta)$$

$$\frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{B \setminus G \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B' \setminus G} \quad (a \notin G) \qquad \frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{B \setminus G \xrightarrow{(\xi,\tau,t)}\!\!\!\!\rightarrow B' \setminus G} \quad (a \in G)$$

$$\frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{B[H] \xrightarrow{(\xi,H(a),t)}\!\!\!\!\rightarrow B'[H]}$$

Table 5.1: Event-based operational semantics for PA_T.

As $\mathbf{0}$ cannot perform any transition there is no rule for this construct. $\surd$ can perform the successful termination action δ at any time t. $(t)\ a_\xi\ ;\ B$ can perform event ξ at time t', $t' \geqslant t$, and evolves into ${}^{t'}[\,B\,]$. ${}^{t'}[\,B\,]$ can be considered as behaviour B shifted t' time units in advance. That is, if B can perform event ξ, say, at time t, then ${}^{t'}[\,B\,]$ can perform ξ at time $t{+}t'$. Note that ${}^{t'}[\,B\,]$ is only an auxiliary construct; it has no counterpart at the language level.

The rules for choice are a straightforward extension of the untimed case—if either B_1 or B_2 can do an event e labelled a at time t, then $B_1 + B_2$ can do so either. The same applies for the rules for parallel composition in which no synchronization takes place, hiding, and relabelling. Synchronization can only take place when both participants can perform an equally labelled event whose label is in the synchronization set G (or equals δ) at time t.

The rules for $>>$ are also a straightforward extension of the rules for the untimed case except that in case B_1 performs a successful termination action δ at time t, then $B_1 >> B_2$ evolves into ${}^{t}[\,B_2\,]$ rather than B_2. This represents that t time units have been passed before B_2 can start with its execution. This is similar to the timed action-prefix case.

For $B_1\,[>\,B_2$ the rules are justified as follows. If B_1 performs an event at time t and evolves into B_1' then $B_1\,[>\,B_2$ can do the same while evolving into $B_1'\,[>\,{}^{t}\{\,B_2\,\}$. ${}^{t}\{\,B_2\,\}$ behaves like B_2 except that it is unable to perform events before t. This ensures that B_2 cannot disrupt $B_1'\,[>\,B_2$ by performing an event at time t', say, while B_1 has performed an event at time $t > t'$. The other inference rules for disrupt are straightforward extensions of the rules for the untimed case.

The inference rule for ${}^{t'}\{\,B\,\}$ is that if B can perform an event at time t, then ${}^{t'}\{\,B\,\}$ can do so if $t \geqslant t'$. Note that ${}^{t'}\{\,B\,\}$ is—like ${}^{t'}[\,B\,]$—an auxiliary operator that cannot be specified by the user.

The inference rules are a conservative (and minimal) extension of the SOS-rules that determine the (untimed) event transition system for PA (cf. Table 2.1)—when we omit the time labels in the transitions and turn ${}^{t}[\,B\,]$ into B we obtain identical rules. Note that the inference rules for ${}^{t}[\,B\,]$ and ${}^{t}\{\,B\,\}$ then reduce to a tautology.

5.1. EXAMPLE. Let $B = (3)\ a_\xi\ ;\ (((2)\ b_\chi\ ;\ \mathbf{0} + (7)\ c_\psi\ ;\ (3)\ d_\eta\ ;\ \mathbf{0})\,||_d\,(12)\ d_\rho\ ;\ \mathbf{0})$. Then we derive using the inference rules of Table 5.1:

$$
\begin{array}{ll}
 & (3)\ a_\xi\ ;\ (((2)\ b_\chi\ ;\ \mathbf{0} + (7)\ c_\psi\ ;\ (3)\ d_\eta\ ;\ \mathbf{0})\,||_d\,(12)\ d_\rho\ ;\ \mathbf{0}) \\
\xrightarrow{(\xi,a,6)}\!\!\!\!\rightarrow & \{\ \text{(timed-action prefix)}\ \} \\
 & {}^{6}[\,((2)\ b_\chi\ ;\ \mathbf{0} + (7)\ c_\psi\ ;\ (3)\ d_\eta\ ;\ \mathbf{0})\,||_d\,(12)\ d_\rho\ ;\ \mathbf{0}\,] \\
\xrightarrow{(\psi,c,13)}\!\!\!\!\rightarrow & \{\ \text{(timed-action prefix), (choice-right), (par-left), (time-shift)}\ \} \\
 & {}^{6}[\,{}^{7}[\,(3)\ d_\eta\ ;\ \mathbf{0}\,]\,||_d\,(12)\ d_\rho\ ;\ \mathbf{0}\,] \\
\xrightarrow{((\eta,\rho),d,22)}\!\!\!\!\rightarrow & \{\ \text{(synchronization), (time-shift)}\ \} \\
 & {}^{6}[\,{}^{7}[\,{}^{9}[\,\mathbf{0}\,]\,]\,||_d\,{}^{16}[\,\mathbf{0}\,]\,]\quad .
\end{array}
$$

It should be noted that time labels in successive transitions do not have to increase, in fact, they can even decrease. Take, for instance, $B = (3)\,a_\xi\ ;\ \mathbf{0}\,|||\,(7)\ b_\psi\ ;\ \mathbf{0}$. A possible derivation of B is $B \xrightarrow{(\psi,b,9)}\!\!\!\!\rightarrow B' \xrightarrow{(\xi,a,4)}\!\!\!\!\rightarrow B''$ where $B' = (3)\,a_\xi\,|||\,{}^{9}[\,\mathbf{0}\,]$ and $B'' = {}^{4}[\,\mathbf{0}\,]\,|||\,{}^{9}[\,\mathbf{0}\,]$; see also

Corollary 5.19. □

5.2. EXAMPLE. As a second example consider

$$B := \Big(((2)\ a_\xi\ ;\ \surd_\psi\ |||\ (14)\ b_\chi\ ;\ \surd_\eta) >> (1)\ c_\rho\ ;\ \mathbf{0}\Big)\ [>\ ((1)\ d_\mu\ ;\ \mathbf{0}\ |||\ (3)\ f_\nu\ ;\ \mathbf{0})\quad .$$

Using the inference rules of Table 5.1 we derive

$$\begin{array}{l}
\quad \Big(((2)\ a_\xi\ ;\ \surd_\psi\ |||\ (14)\ b_\chi\ ;\ \surd_\eta) >> (1)\ c_\rho\ ;\ \mathbf{0}\Big)\ [>\ ((1)\ d_\mu\ ;\ \mathbf{0}\ |||\ (3)\ f_\nu\ ;\ \mathbf{0}) \\
\xrightarrow{((*,\chi),b,17)}\!\!\!\twoheadrightarrow \{\ \text{(timed action-prefix), (par-right), (enabling-left), (disrupt-left)}\ \} \\
\quad \Big(((2)\ a_\xi\ ;\ \surd_\psi\ |||\ {}^{17}[\ \surd_\eta\ ]) >> (1)\ c_\rho\ ;\ \mathbf{0}\Big)\ [>\ {}^{17}\{\ (1)\ d_\mu\ ;\ \mathbf{0}\ |||\ (3)\ f_\nu\ ;\ \mathbf{0}\ \} \\
\xrightarrow{((\xi,*),a,5)}\!\!\!\twoheadrightarrow \{\ \text{(timed action-prefix), (par-left), (enabling-left), (disrupt-left)}\ \} \\
\quad \Big(({}^{5}[\ \surd_\psi\ ]\ |||\ {}^{17}[\ \surd_\eta\ ]) >> (1)\ c_\rho\ ;\ \mathbf{0}\Big)\ [>\ {}^{5}\{\ {}^{17}\{\ (1)\ d_\mu\ ;\ \mathbf{0}\ |||\ (3)\ f_\nu\ ;\ \mathbf{0}\ \}\ \} \\
\xrightarrow{(\mu,d,33)}\!\!\!\twoheadrightarrow \{\ \text{(timed action-prefix), (disrupt-right), (rule for } {}^{t}\{\ B\ \})\ \} \\
\quad {}^{5}\{\ {}^{17}\{\ {}^{33}[\ \mathbf{0}\ ]\ |||\ (3)\ f_\nu\ ;\ \mathbf{0}\ \}\ \} \\
\xrightarrow{(\nu,f,18)}\!\!\!\twoheadrightarrow \{\ \text{(timed action-prefix), (rule for } {}^{t}\{\ B\ \}\text{), (par-right)}\ \} \\
\quad {}^{5}\{\ {}^{17}\{\ {}^{33}[\ \mathbf{0}\ ]\ |||\ {}^{18}[\ \mathbf{0}\ ]\ \}\ \}\quad .
\end{array}$$

□

Remark that nested ${}^{t}[\ ]$ and ${}^{t}\{\ \}$ operators can be simplified as follows: ${}^{t}[\ {}^{t'}[\ B\ ]\ ] = {}^{t+t'}[\ B\ ]$ and ${}^{t}\{\ {}^{t'}\{\ B\ \}\ \} = {}^{max(t,t')}\{\ B\ \}$.

In the remainder of this section we formally define the transition system defined by $\longrightarrow\!\!\!\!\twoheadrightarrow$ and show the correspondence of this transition system with the standard transition system for PA defined in Chapter 1. The intuition is that if we take the transition system for B induced by $\longrightarrow\!\!\!\!\twoheadrightarrow$ and abstract from the timing aspects and event identities then we obtain the standard transition system for $\Phi_T(B)$, the untimed counterpart of B.

The set of derivatives of expression B consists of all expressions B' that can be reached from B by performing zero or more $\longrightarrow\!\!\!\!\twoheadrightarrow$ transitions.

5.3. DEFINITION. (*Behaviour derivatives*)

For $B \in \mathsf{PA}_T$ the set of derivatives of B, $Der(B)$, is the smallest set satisfying:

- $B \in Der(B)$
- $B' \in Der(B)\ \wedge\ B' \xrightarrow{(e,a,t)}\!\!\!\twoheadrightarrow B''\ \Rightarrow\ B'' \in Der(B)$.

□

5.4. DEFINITION. For $B \in \mathsf{PA}_T$ the set of events in B, denoted $E(B)$, is defined by

$$E(B)\ \triangleq\ \{\ e \in Ev \mid \exists\, a \in \mathsf{Act}^{\tau,\delta}, B', B'' \in Der(B), t \in \mathsf{Time} : B' \xrightarrow{(e,a,t)}\!\!\!\twoheadrightarrow B''\ \}.$$

□

Let $l_B : E(B) \longrightarrow \mathsf{Act}^{\tau,\delta}$ associate to each event in B its action label. That is, $l_B(e) = a$ iff $\exists\, B', B'' \in Der(B) : B' \xrightarrow{(e,a,t)}\!\!\!\twoheadrightarrow B''$. This permits us to write $B \xrightarrow{(e,t)}\!\!\!\twoheadrightarrow B'$ instead of $B \xrightarrow{(e,l_B(e),t)}\!\!\!\twoheadrightarrow B''$.

5.5. DEFINITION. (*Timed event transition system*)

For $B \in \mathsf{PA}_T$ the timed event transition system $\mathsf{TS}_T(B) \triangleq \langle S, L, T, s_0 \rangle$ with

- $S = Der(B)$
- $L = \{ (e,t) \mid \exists\, a \in \mathsf{Act}^{\tau,\delta}, B', B'' \in Der(B) : B' \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B'' \}$
- $T = \{ \xrightarrow{(e,t)}\!\!\!\!\rightarrow \mid (e,t) \in L \}$ where $\xrightarrow{(e,t)}\!\!\!\!\rightarrow = \{ (B_1, B_2) \in S \times S \mid B_1 \xrightarrow{(e,t)}\!\!\!\!\rightarrow B_2 \}$
- $s_0 = B$.

$\square$

Transition relation $\longrightarrow\!\!\!\!\rightarrow$ is said to be *deterministic* iff

$$\forall\, B : (\exists\, B', B'' : B \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B' \;\wedge\; B \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B'' \;\Rightarrow\; B' = B'') \quad .$$

A transition relation that does not satisfy this property is called *nondeterministic*.

Since event identifiers are unique and (together with the time at which they occur) uniquely determine the successor state of a state in $\mathsf{TS}_T(B)$ we have that the transition system does not contain nondeterminism.

5.6. LEMMA. $\forall\, B \in \mathsf{PA}_T : \mathsf{TS}_T(B)$ is deterministic.

PROOF. Straightforward by induction on the structure of B. $\square$

We extend the function $\Phi_T : \mathsf{PA}_T \longrightarrow \mathsf{PA}$, which associates to a timed behaviour its corresponding untimed behaviour by simply omitting all time annotations, to include ${}^t[\, B\,]$ and ${}^t\{\, B\,\}$ as well. Let PA_T^+ denote the extension of PA_T with ${}^t[\, B\,]$ and ${}^t\{\, B\,\}$ and let $\Phi_T({}^t[\, B\,]) \triangleq \Phi_T(B)$ and $\Phi_T({}^t\{\, B\,\}) \triangleq \Phi_T(B)$ for any B. We then have (recall from Chapter 1 that $\xrightarrow{a}$ is the transition relation from the standard interleaving semantics of PA):

5.7. LEMMA. $\forall\, B, B' \in \mathsf{PA}_T^+ : (\exists\, e, t : B \xrightarrow{(e,t)}\!\!\!\!\rightarrow B' \;\wedge\; l_B(e) = a)$ iff $\Phi_T(B) \xrightarrow{a} \Phi_T(B')$.

PROOF. By induction on the structure of B.
Base : The base cases that we consider are $\mathbf{0}$, $\surd_\xi$ and timed action-prefix.

1. $B = \mathbf{0}$. Trivial as $\mathbf{0}$ cannot perform any $\longrightarrow\!\!\!\!\rightarrow$ transitions and $\Phi_T(\mathbf{0}) = \mathbf{0}$ cannot perform any $\longrightarrow$ transitions.

2. $B = \surd_\xi$. Trivial as $\surd_\xi$ can only perform δ at time t, evolving into $\mathbf{0}$. $\Phi_T(\surd_\xi) = \surd_\xi$ can only perform δ and evolves into $\mathbf{0} = \Phi_T(\mathbf{0})$.

3. $B = (t)\ a_\xi\, ; B_1$. Then we have $B \xrightarrow{(\xi,a,t')}\!\!\!\!\rightarrow {}^{t'}[\, B_1\,]$ for $t' \geqslant t$. $\Phi_T(B) = a_\xi\, ; \Phi_T(B_1)$. For this construct the only possible $\longrightarrow$ transition is $\Phi_T(B) \xrightarrow{a} \Phi_T(B_1)$. Since $\Phi_T({}^{t'}[\, B_1\,]) = \Phi_T(B_1)$ this proves the case.

Induction Step: Assume the lemma holds for B_1 and B_2. We only provide the proof for time-shift and parallel composition. The proofs for the other operators are rather similar and omitted here.

1. $B = {}^t[\,B_1\,]$. For this case we derive

$$
\begin{array}{ll}
& \exists\, e, t' : {}^t[\,B_1\,] \xrightarrow{(e,t+t')}\!\!\!\!\rightarrow {}^t[\,B_2\,] \;\wedge\; l_B(e) = a \\
\Leftrightarrow & \{ \text{ SOS-rule for } {}^t[\;] \} \\
& \exists\, e, t' : B_1 \xrightarrow{(e,t')}\!\!\!\!\rightarrow B_2 \;\wedge\; l_{B_1}(e) = a \\
\Leftrightarrow & \{ \text{ induction hypothesis } \} \\
& \Phi_T(B_1) \xrightarrow{a} \Phi_T(B_2) \\
\Leftrightarrow & \{ \text{ definition of } \Phi_T \} \\
& \Phi_T({}^t[\,B_1\,]) \xrightarrow{a} \Phi_T({}^t[\,B_2\,]) \quad .
\end{array}
$$

2. $B = B_1 \,||_G\, B_2$. For this case we derive

$$
\begin{array}{ll}
& \exists\, e, t : B_1 \,||_G\, B_2 \xrightarrow{(e,t)}\!\!\!\!\rightarrow B' \;\wedge\; l_B(e) = a \\
\Leftrightarrow & \{ \text{ SOS-rule } (\twoheadrightarrow) \text{ for } ||_G \} \\
& \quad (\exists\, e, t : B_1 \xrightarrow{(e,t)}\!\!\!\!\rightarrow B_1' \;\wedge\; l_{B_1}(e) = a \;\wedge\; a \notin G^\delta) \\
& \vee\; (\exists\, e, t : B_2 \xrightarrow{(e,t)}\!\!\!\!\rightarrow B_2' \;\wedge\; l_{B_2}(e) = a \;\wedge\; a \notin G^\delta) \\
& \vee\; (\exists\, e, e', t : B_1 \xrightarrow{(e,t)}\!\!\!\!\rightarrow B_1' \;\wedge\; B_2 \xrightarrow{(e',t)}\!\!\!\!\rightarrow B_2' \;\wedge\; l_{B_1}(e){=}l_{B_2}(e'){=}a \;\wedge\; a \in G^\delta) \\
\Leftrightarrow & \{ \text{ induction hypothesis } \} \\
& \quad (\Phi_T(B_1) \xrightarrow{a} \Phi_T(B_1') \;\wedge\; a \notin G^\delta) \;\vee\; (\Phi_T(B_2) \xrightarrow{a} \Phi_T(B_2') \;\wedge\; a \notin G^\delta) \\
& \vee\; (\Phi_T(B_1) \xrightarrow{a} \Phi_T(B_1') \;\wedge\; \Phi_T(B_2) \xrightarrow{a} \Phi_T(B_2') \;\wedge\; a \in G^\delta) \\
\Leftrightarrow & \{ \text{ SOS-rule } (\longrightarrow) \text{ for } ||_G \} \\
& \Phi_T(B_1) \,||_G\, \Phi_T(B_2) \xrightarrow{a} B'' \\
\Leftrightarrow & \{ \text{ definition of } \Phi_T \} \\
& \Phi_T(B_1 \,||_G\, B_2) \xrightarrow{a} B'' \quad .
\end{array}
$$

It is now easy to check that $B'' = \Phi_T(B')$.

□

For a set S of behaviours $\{ B_1, \ldots, B_n \}$ let $\Phi_T(S) \triangleq \{ \Phi_T(B_1), \ldots, \Phi_T(B_n) \}$.

5.8. COROLLARY. $\forall\, B \in \mathsf{PA}_T : \Phi_T(\mathit{Der}(B)) = \mathit{Der}(\Phi_T(B))$.

PROOF. Straightforward from Lemma 5.7. □

5.9. DEFINITION. Let $\mathsf{TS}_T(B) = \langle S, L, T, s_0 \rangle$. The corresponding untimed transition system, denoted $\Phi(\mathsf{TS}_T(B))$, is defined as $\Phi(\mathsf{TS}_T(B)) \triangleq \langle S', L', T', s_0' \rangle$ where

- $S' = \Phi_T(S)$
- $L' = \{ l_B(e) \mid (e, t) \in L \}$
- $T' = \{ \xrightarrow{a} \mid a \in L' \}$ where
  $$\xrightarrow{a} \;=\; \{ (\Phi_T(B_1), \Phi_T(B_2)) \mid \exists\, e, t : B_1 \xrightarrow{(e,t)}\!\!\!\!\rightarrow B_2 \;\wedge\; l_B(e) = a \}$$
- $s_0' = \Phi_T(s_0)$.

□

The correspondence between the timed event transition system of PA_T and the standard interleaving system of PA is presented in the following theorem. It says that when we construct for timed behaviour B the automaton $\mathsf{TS}_T(B)$ and subsequently omit times from this transition system while focusing on action labels rather than on event labels (i.e., construct $\Phi(\mathsf{TS}_T(B))$), we obtain the same result as we get by constructing the standard transition system TS for the corresponding untimed behaviour $\Phi_T(B)$. That is,

5.10. THEOREM. $\forall\, B \in \mathsf{PA}_T : \Phi(\mathsf{TS}_T(B)) = \mathsf{TS}(\Phi_T(B))$.

PROOF. Let $\Phi(\mathsf{TS}_T(B)) = \langle S', L', T', s_0' \rangle$. We then derive

1. For the set of states S' we have by definition of $\mathsf{TS}_T(B)$ that $S' = \Phi_T(S)$, and since $S = Der(B)$, we obtain $S' = \Phi_T(Der(B))$. From Corollary 5.8 it immediately follows $S' = Der(\Phi_T(B))$.
2. For the label set L' we derive

$$\begin{array}{ll} & \{\, l_B(e) \mid (e,t) \in L \,\} \\ = & \{ \text{ Definition 5.5 } \} \\ & \{\, l_B(e) \mid \exists\, a \in \mathsf{Act}^{\tau,\delta}, e, t \in \mathsf{Time}, B', B'' \in Der(B) : B' \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B'' \,\} \\ = & \{ \text{ Lemma 5.7 } \} \\ & \{\, a \mid \exists\, \Phi_T(B'), \Phi_T(B'') \in Der(\Phi_T(B)) : \Phi_T(B') \xrightarrow{a} \Phi_T(B'') \,\} \quad . \end{array}$$

3. For T' we have

$$\begin{array}{ll} & \{\, (\Phi_T(B_1), \Phi_T(B_2)) \mid \exists\, e, t : B_1 \xrightarrow{(e,t)}\!\!\!\!\rightarrow B_2 \ \wedge \ l_B(e) = a \,\} \\ = & \{ \text{ Lemma 5.7 } \} \\ & \{\, (\Phi_T(B_1), \Phi_T(B_2)) \mid \exists\, a : \Phi_T(B_1) \xrightarrow{a} \Phi_T(B_2) \,\} \quad . \end{array}$$

4. For the initial state we have $s_0' = \Phi_T(s_0) = \Phi_T(B)$.

It is now easy to check that $\Phi(\mathsf{TS}_T(B)) = \mathsf{TS}(\Phi_T(B))$ for any B. □

This shows that the timed event transition system (induced by $\longrightarrow\!\!\!\!\rightarrow$) for $B \in \mathsf{PA}_T$ is a straightforward and conservative extension of the transition system (induced by $\longrightarrow$) for $\Phi_T(B) \in \mathsf{PA}$.

5.3 Correspondence with causality-based semantics

This section proves the consistency between the denotational semantics $\mathcal{E}_T[\![\ ]\!]$ of PA_T in terms of timed event structures (see Chapter 4) and its operational semantics induced by the inference rules for $\longrightarrow\!\!\!\!\rightarrow$. The consistency proof is carried out in two steps. First, we characterize the timed event traces that are generated by the operational semantics of B in a denotational way. This yields a denotational trace semantics for B, denoted $\mathcal{T}_T[\![\, B \,]\!]$. Secondly, it is proven that this trace semantics coincides with the timed event traces obtained from the causality-based semantics of B, $\mathcal{E}_T[\![\, B \,]\!]$.

We start by extending $\longrightarrow\!\!\!\!\rightarrow$ towards a trace derivation relation $\xrightarrow{\sigma}\!\!\!\!\rightarrow$ in the usual way:

5.11. DEFINITION. For $B \in \mathsf{PA}_T$, and $\sigma = (e_1, a_1, t_1) \dots (e_n, a_n, t_n)$ for $n \geqslant 0$, let

$$B \xrightarrow{\sigma}\!\!\!\!\rightarrow B' \triangleq (\exists\, B_i : B = B_0 \xrightarrow{(e_1,a_1,t_1)}\!\!\!\!\rightarrow B_1 \xrightarrow{(e_2,a_2,t_2)}\!\!\!\!\rightarrow \dots \xrightarrow{(e_n,a_n,t_n)}\!\!\!\!\rightarrow B_n = B').$$

□

The following notion is needed to characterize the timed event traces for parallel composition.

5.12. DEFINITION. Let S_1 and S_2 be sets of triples of events, actions and a time, and let G be a set of action labels ($G \subseteq \mathsf{Act}$). The set $S_1 \bowtie_G S_2$ is defined by

$$\begin{aligned} S_1 \bowtie_G S_2 \triangleq \{\, ((e,e'),a,t) \mid\; & (a \in G^\delta \;\wedge\; (e,a,t) \in S_1 \;\wedge\; (e',a,t) \in S_2) \;\vee \\ & (a \notin G^\delta \;\wedge\; (e,a,t) \in S_1 \;\wedge\; e' = *) \;\vee \\ & (a \notin G^\delta \;\wedge\; e = * \;\wedge\; (e',a,t) \in S_2) \,\} \enspace . \end{aligned}$$

□

So, $((e,e'),a,t)$ is a member of $S_1 \bowtie_G S_2$ if (i) a is a synchronization event (i.e., $a \in G^\delta$), $(e,a,t) \in S_1$ and $(e',a,t) \in S_2$ or (ii) a is a non-synchronization event (i.e., $a \notin G^\delta$), $(e,a,t) \in S_1$ and $e' = *$ (or similar for $(e',a,t) \in S_2$ and $e = *$). Notice that for case (i) triples of S_1 and S_2 are required to have identical timings.

$(S_1 \bowtie_G S_2)^*$ consists of all finite sequences constructed from elements of the set $S_1 \bowtie_G S_2$.

5.13. DEFINITION. For $\sigma \in (S_1 \bowtie_G S_2)^*$ projections $\pi_1(\sigma)$ and $\pi_2(\sigma)$ are defined by:

- $\pi_i(\varepsilon) \triangleq \varepsilon$, for $i = 1, 2$
- $\pi_1(((e,e'),a,t)\ \sigma') \triangleq \begin{cases} \pi_1(\sigma') & \text{if } e = * \\ (e,a,t)\ \pi_1(\sigma') & \text{otherwise} \end{cases}$
- $\pi_2(((e,e'),a,t)\ \sigma') \triangleq \begin{cases} \pi_2(\sigma') & \text{if } e' = * \\ (e',a,t)\ \pi_2(\sigma') & \text{otherwise .} \end{cases}$

□

In order to characterize the set of timed event traces generated by the SOS-rules for $\longrightarrow\!\!\!\!\rightarrow$ we need the following auxiliary operations on traces.

5.14. DEFINITION. The following operations on timed event trace σ are defined:

1. For set of actions G, $\sigma \setminus G$ ('σ with G hidden') is defined by
   (a) $\varepsilon \setminus G \triangleq \varepsilon$
   (b) $((e,a,t)\ \sigma') \setminus G \triangleq \begin{cases} (e,\tau,t)\ (\sigma' \setminus G) & \text{if } a \in G \\ (e,a,t)\ (\sigma' \setminus G) & \text{if } a \notin G \end{cases}$
2. For relabelling function H, $\sigma[H]$ ('σ relabelled with H') is defined by
   (a) $\varepsilon[H] \triangleq \varepsilon$
   (b) $((e,a,t)\ \sigma')[H] \triangleq (e,H(a),t)\ (\sigma'[H])$

3. For $t \in \mathsf{Time}$, ${}^{t}[\,\sigma\,]$ ('σ delayed by t') is defined by
   (a) ${}^{t}[\,\varepsilon\,] \triangleq \varepsilon$
   (b) ${}^{t}[\,(e,a,t')\ \sigma'\,] \triangleq (e,a,t'{+}t)\ {}^{t}[\,\sigma'\,]$
4. $\mathsf{mx}(\sigma)$ denotes the maximal time instant occurring in σ and is defined by
   (a) $\mathsf{mx}(\varepsilon) \triangleq 0$
   (b) $\mathsf{mx}((e,a,t)\ \sigma') \triangleq max(t, \mathsf{mx}(\sigma'))$.

$\square$

Let $\overline{V}$ for V a set of timed event traces denote the set of timed labelled events occurring in a timed trace in V.

5.15. DEFINITION. For V a set of timed event traces let $\overline{V} \triangleq \{\, s \mid \exists\, \sigma \in V : s \in \overline{\sigma}\,\}$. $\square$

The set of timed event traces of B is defined in a denotational way as follows.

5.16. DEFINITION. For $B \in \mathsf{PA}_T$ the set of timed traces of B, $\mathcal{T}_T[\![\,B\,]\!]$, is defined by:

$$
\begin{array}{rcl}
\mathcal{T}_T[\![\,\mathbf{0}\,]\!] & \triangleq & \{\,\varepsilon\,\} \\
\mathcal{T}_T[\![\,\surd_\xi\,]\!] & \triangleq & \{\,\varepsilon\,\} \cup \{\,(\xi,\delta,t) \mid t \in \mathsf{Time}\,\} \\
\mathcal{T}_T[\![\,(t)\ a_\xi\,;\,B\,]\!] & \triangleq & \{\,(\xi,a,t')\,{}^{t'}[\,\sigma\,] \mid t' \geqslant t\ \wedge\ \sigma \in \mathcal{T}_T[\![\,B\,]\!]\,\} \cup \{\,\varepsilon\,\} \\
\mathcal{T}_T[\![\,B_1 + B_2\,]\!] & \triangleq & \mathcal{T}_T[\![\,B_1\,]\!] \cup \mathcal{T}_T[\![\,B_2\,]\!] \\
\mathcal{T}_T[\![\,B_1 >> B_2\,]\!] & \triangleq & \quad \{\,\sigma_1\,(e,\tau,t)\,{}^{t}[\,\sigma_2\,] \mid \sigma_1\,(e,\delta,t) \in \mathcal{T}_T[\![\,B_1\,]\!]\ \wedge\ \sigma_2 \in \mathcal{T}_T[\![\,B_2\,]\!]\,\} \\
 & & \cup\ \{\,\sigma \in \mathcal{T}_T[\![\,B_1\,]\!] \mid \sigma \neq \sigma'\,(e,\delta,t)\,\} \\
\mathcal{T}_T[\![\,B_1\,[>\,B_2\,]\!] & \triangleq & \{\,\sigma_1\,\sigma_2 \mid \sigma_1 \in \mathcal{T}_T[\![\,B_1\,]\!]\ \wedge\ \sigma_1 \neq \sigma_1'(e,\delta,t)\ \wedge\ \sigma_2 \in \mathcal{T}_T[\![\,B_2\,]\!] \\
 & & \qquad \wedge\ (\forall\, e_i \in \overline{\sigma_2} : t_i \geqslant \mathsf{mx}(\sigma_1))\,\} \\
 & & \cup\ \{\,\sigma \in \mathcal{T}_T[\![\,B_1\,]\!] \mid \sigma = \sigma'\,(e,\delta,t)\,\} \\
\mathcal{T}_T[\![\,B[H]\,]\!] & \triangleq & \{\,\sigma \mid \exists\,\sigma' \in \mathcal{T}_T[\![\,B\,]\!] : \sigma = \sigma'[H]\,\} \\
\mathcal{T}_T[\![\,B \setminus G\,]\!] & \triangleq & \{\,\sigma \mid \exists\,\sigma' \in \mathcal{T}_T[\![\,B\,]\!] : \sigma = \sigma' \setminus G\,\} \\
\mathcal{T}_T[\![\,B_1\,||_G\,B_2\,]\!] & \triangleq & \{\,\sigma \in (\overline{\mathcal{T}_T[\![\,B_1\,]\!]} \bowtie_G \overline{\mathcal{T}_T[\![\,B_2\,]\!]})^* \mid \pi_i(\sigma) \in \mathcal{T}_T[\![\,B_i\,]\!] \text{ for } i{=}1,2\,\}.
\end{array}
$$

$\square$

Definition 5.16 provides a denotational timed event trace semantics for PA_T. The following lemma shows that this denotational timed event trace semantics $\mathcal{T}_T[\![\,B\,]\!]$ coincides with the timed event traces generated by $\longrightarrow\!\!\!\!\!\rightarrow$.

5.17. LEMMA. $\forall\, B \in \mathsf{PA}_T : \mathcal{T}_T[\![\,B\,]\!] = \{\,\sigma \mid \exists\, B' : B \xrightarrow{\sigma}\!\!\!\!\rightarrow B'\,\}$.

PROOF. By induction on the structure of B.
Base: For $B = \mathbf{0}$ we have that $\{\,\sigma \mid \exists\, B' : \mathbf{0} \xrightarrow{\sigma}\!\!\!\!\rightarrow B'\,\}$ equals $\{\,\varepsilon\,\}$. By Definition 5.16 this equals $\mathcal{T}_T[\![\,\mathbf{0}\,]\!]$. From the SOS-rules it follows directly that for $B = \surd_\xi$ the only timed event traces are ε and (ξ,δ,t) for any $t \in \mathsf{Time}$. By Definition 5.16 this equals $\mathcal{T}_T[\![\,\surd_\xi\,]\!]$.
Induction Step: Assume the lemma holds for B_1 and B_2. For convenience let $T(B)$ denote $\{\,\sigma \mid \exists\, B' : B \xrightarrow{\sigma}\!\!\!\!\rightarrow B'\,\}$. We consider the proofs for timed action-prefix and disrupt. The proofs for the other constructs are similar and omitted.

1. $B = (t)\ a_\xi\,;\, B_1$. By induction on the length of σ it is rather straightforward to prove, using the SOS-rules of Table 5.1, that for nonempty σ behaviour $(t)\ a_\xi\,;\, B_1 \xrightarrow{\sigma}\!\!\!\!\twoheadrightarrow$ iff $\sigma = (\xi, a, t')\,\sigma'$ with $t' \geqslant t$ such that $(t)\ a_\xi\,;\, B_1 \xrightarrow{(\xi,a,t')}\!\!\!\!\twoheadrightarrow {}^{t'}[\,B_1\,]$ and ${}^{t'}[\,B_1\,] \xrightarrow{\sigma'}\!\!\!\!\twoheadrightarrow$. Thus, we have:

$$\begin{array}{ll}
 & \{\,\sigma \mid \exists\, B' : (t)\ a_\xi\,;\, B_1 \xrightarrow{\sigma}\!\!\!\!\twoheadrightarrow B'\,\} \\
= & \{\text{ see above }\} \\
 & \{\,(\xi,a,t')\,\sigma' \mid t' \geqslant t\ \wedge\ \sigma' \in T({}^{t'}[\,B_1\,])\,\} \cup \{\,\varepsilon\,\} \\
= & \{\ \sigma' \in T({}^{t'}[\,B_1\,]) \Leftrightarrow (\sigma \in T(B_1)\ \wedge\ \sigma' = {}^{t'}[\,\sigma\,])\ \} \\
 & \{\,(\xi,a,t')\,{}^{t'}[\,\sigma\,] \mid t' \geqslant t\ \wedge\ \sigma \in T(B_1)\,\} \cup \{\,\varepsilon\,\} \\
= & \{\text{ induction hypothesis }\} \\
 & \{\,(\xi,a,t')\,{}^{t'}[\,\sigma\,] \mid t' \geqslant t\ \wedge\ \sigma \in \mathcal{T}_T[\![\,B_1\,]\!]\,\} \cup \{\,\varepsilon\,\} \\
= & \{\text{ Definition 5.16 }\} \\
 & \mathcal{T}_T[\![\,(t)\,a_\xi\,;\,B_1\,]\!]\quad .
\end{array}$$

2. $B = B_1 \,[>\, B_2$. By induction on the length of σ, using the SOS-rules of Table 5.1, it is not hard to prove (but tedious) that $B_1\,[>\,B_2 \xrightarrow{\sigma}\!\!\!\!\twoheadrightarrow$ iff either

   (i) $\sigma = \sigma_1\,(e,\delta,t)$, and $B_1 \xrightarrow{\sigma}\!\!\!\!\twoheadrightarrow B_1'$ or

   (ii) $\sigma = \sigma_1\,\sigma_2$, $B_1 \xrightarrow{\sigma_1}\!\!\!\!\twoheadrightarrow B_1'$, $\sigma_1 \neq \sigma_1'\,(e,\delta,t)$, and ${}^{t_n}\{\,\ldots{}^{t_1}\{\,B_2\,\}\,\} \xrightarrow{\sigma_2}\!\!\!\!\twoheadrightarrow B_2'$ for $\sigma_1 = (e_1,t_1)\ldots(e_n,t_n)$.

   So, we can derive:

$$\begin{array}{ll}
 & \{\,\sigma \mid \exists\, B' : B_1\,[>\,B_2 \xrightarrow{\sigma}\!\!\!\!\twoheadrightarrow B'\,\} \\
= & \{\text{ (i) and (ii) }\} \\
 & \quad\{\,\sigma \in T(B_1) \mid \sigma \neq \sigma'\,(e,\delta,t)\,\} \\
 & \cup\,\{\,\sigma_1\,\sigma_2 \mid \sigma_1 \in T(B_1)\ \wedge\ \sigma_1 \neq \sigma_1'\,(e,\delta,t)\ \wedge\ \sigma_2 \in T({}^{t_n}\{\,\ldots{}^{t_1}\{\,B_2\,\}\,\})\,\} \\
= & \{\ {}^{t}\{\,{}^{t'}\{\,B\,\}\,\} = {}^{\max(t,t')}\{\,B\,\};\ \text{definition } \mathsf{mx}\ \} \\
 & \quad\{\,\sigma \in T(B_1) \mid \sigma \neq \sigma'\,(e,\delta,t)\,\} \\
 & \cup\,\{\,\sigma_1\,\sigma_2 \mid \sigma_1 \in T(B_1)\ \wedge\ \sigma_1 \neq \sigma_1'\,(e,\delta,t)\ \wedge\ \sigma_2 \in T({}^{\mathsf{mx}(\sigma_1)}\{\,B_2\,\})\,\} \\
= & \{\ \sigma \in T({}^{t}\{\,B\,\}) \Leftrightarrow (\sigma' \in T(B)\ \wedge\ (\forall\, e_i \in \overline{\sigma}' : t_i \geqslant t)\ \} \\
 & \quad\{\,\sigma \in T(B_1) \mid \sigma \neq \sigma'\,(e,\delta,t)\,\} \\
 & \cup\,\{\,\sigma_1\,\sigma_2 \mid \sigma_1 \in T(B_1)\ \wedge\ \sigma_1 \neq \sigma_1'\,(e,\delta,t)\ \wedge\ \sigma_2 \in T(B_2)\ \wedge\ (\forall\, e_i \in \overline{\sigma_2} : t_i \geqslant \mathsf{mx}(\sigma_1))\,\} \\
= & \{\text{ induction hypothesis }\} \\
 & \{\,\sigma \in \mathcal{T}_T[\![\,B_1\,]\!] \mid \sigma \neq \sigma'\,(e,\delta,t)\,\} \cup \\
 & \{\,\sigma_1\,\sigma_2 \mid \sigma_1 \in \mathcal{T}_T[\![\,B_1\,]\!]\ \wedge\ \sigma_1 \neq \sigma_1'\,(e,\delta,t)\ \wedge\ \sigma_2 \in \mathcal{T}_T[\![\,B_2\,]\!]\ \wedge\ (\forall\, e_i \in \overline{\sigma_2} : t_i \geqslant \mathsf{mx}(\sigma_1))\,\} \\
= & \{\text{ Definition 5.16 }\} \\
 & \mathcal{T}_T[\![\,B_1\,[>\,B_2\,]\!]\quad .
\end{array}$$

□

In order to relate the operationally characterized timed event traces and the traces obtained from the causality-based semantics $\mathcal{E}_T[\![\ ]\!]$ we slightly adapt the definition of $\mathcal{E}_T[\![\ ]\!]$ for $\surd$ and $(t)\ a\,;\,B$. In the current definition of $\mathcal{E}_T[\![\ ]\!]$ a unique but arbitrary event is introduced for these constructs modelling the appearance of δ and a, respectively. Here we assume that all occurrences of $\surd$ and action-prefix are uniquely identified, and we take this unique identification as the unique event name in the definition of $\mathcal{E}_T[\![\;]\!]$.

The following theorem says that the set of timed event traces of behaviour B of PA_T is identical to the set of timed event traces of the corresponding timed event structure $\mathcal{E}_T[\![\,B\,]\!]$.

5.18. THEOREM. $\forall\, B \in \mathsf{PA}_T : T_T(\mathcal{E}_T[\![\,B\,]\!]) = \mathcal{T}_T[\![\,B\,]\!]$.

PROOF. By induction on the structure of B.
Base: For $B = \mathbf{0}$ we simply have $T_T(\mathcal{E}_T[\![\,\mathbf{0}\,]\!]) = \{\,\varepsilon\,\} = \mathcal{T}_T[\![\,\mathbf{0}\,]\!]$.
For $B = \surd_\xi$ we have $T_T(\mathcal{E}_T[\![\,\surd_\xi\,]\!]) = \{\,\varepsilon\,\} \cup \{\,(\xi,\delta,t) \mid t \in \mathsf{Time}\,\} = \mathcal{T}_T[\![\,\surd_\xi\,]\!]$.
Induction Step: Assume the theorem holds for B_1 and B_2. We only provide proofs for timed action-prefix, choice, enabling and disrupt. The proofs for the other operators are conducted in a similar way as for the untimed case [89, Theorem 7.5.3], and are omitted here.
Let $\Gamma_i = \mathcal{E}_T[\![\,B_i\,]\!] = \langle(E_i, \rightsquigarrow_i, \mapsto_i, l_i), \mathcal{D}_i, \mathcal{T}_i\rangle$ for i=1,2, and $\Gamma = \mathcal{E}_T[\![\,B\,]\!]$.

1. $B = (t)\ a_\xi\,;B_1$. For Γ bundles $\{\,\{\,(\xi,a)\,\}\,\} \times \mathit{pin}(E_1)$ have been added to $\langle(\{\,\xi\,\},\varnothing,\varnothing, \{\,(\xi,a)\,\}), \{\,(\xi,t)\,\},\varnothing\rangle$. The non-empty timed event traces of Γ are therefore those interleavings of (ξ,a,t') and ${}^{t'}[\,\sigma\,]$, with $\sigma \in T_T(\Gamma_1)$, that satisfy the following constraints: (i) the first element of ${}^{t'}[\,\sigma\,]$ is preceded by (ξ,a,t'), and (ii) $t' \geqslant \mathcal{D}(\xi) = t$. Thus we derive:
$$\begin{array}{cl} & T_T(\mathcal{E}_T[\![\,(t)\ a_\xi\,;B_1\,]\!]) \\ = & \{\text{ see above }\} \\ & \{\,(\xi,a,t')\,{}^{t'}[\,\sigma\,] \mid t' \geqslant t\ \wedge\ \sigma \in T_T(\Gamma_1)\,\} \cup \{\,\varepsilon\,\} \\ = & \{\text{ induction hypothesis }\} \\ & \{\,(\xi,a,t')\,{}^{t'}[\,\sigma\,] \mid t' \geqslant t\ \wedge\ \sigma \in \mathcal{T}_T[\![\,B_1\,]\!]\,\} \cup \{\,\varepsilon\,\} \\ = & \{\text{ Definition 5.16 }\} \\ & \mathcal{T}_T[\![\,(t)\ a_\xi\,;B_1\,]\!]\quad . \end{array}$$

2. $B = B_1 + B_2$. In Γ mutual conflicts are introduced between $\mathit{init}(\Gamma_1)$ and $\mathit{init}(\Gamma_2)$. So, the timed event traces of Γ are those interleavings of $\sigma_1 \in T_T(\Gamma_1)$ and $\sigma_2 \in T_T(\Gamma_2)$ such that the first timed event in σ_1 precedes the first timed event in σ_2, and vice versa. That is, the trace is either σ_1 or σ_2. So, $T_T(\mathcal{E}_T[\![\,B_1 + B_2\,]\!]) = T_T(\Gamma_1) \cup T_T(\Gamma_2)$, which, by induction hypothesis, equals $\mathcal{T}_T[\![\,B_1\,]\!] \cup \mathcal{T}_T[\![\,B_2\,]\!]$. By Definition 5.16 this equals $\mathcal{T}_T[\![\,B_1 + B_2\,]\!]$.

3. $B = B_1 >> B_2$. In Γ a bundle from $\mathit{exit}(\Gamma_1)$ to $\mathit{pin}(\Gamma_2)$ is introduced. This means that traces of Γ are either (i) traces of Γ_1 that do not contain a δ, or (ii) concatenations of $\sigma_1\,(e,\tau,t)$ and ${}^{t}[\,\sigma_2\,]$ with $\sigma_1\,(e,\delta,t)$ a trace of Γ_1, and σ_2 a trace of Γ_2. That is,
$$\begin{array}{cl} & T_T(\mathcal{E}_T[\![\,B_1 >> B_2\,]\!]) \\ = & \{\text{ see discussion above }\} \\ & \{\,\sigma \in T_T(\Gamma_1) \mid \sigma \neq \sigma'\,(e,\delta,t)\,\} \cup \{\,\sigma_1\,(e,\tau,t)\,{}^{t}[\,\sigma_2\,] \mid \sigma_1\,(e,\delta,t) \in T_T(\Gamma_1) \wedge \sigma_2 \in T_T(\Gamma_2)\,\} \\ = & \{\text{ induction hypothesis }\} \\ & \quad \{\,\sigma \in \mathcal{T}_T[\![\,B_1\,]\!] \mid \sigma \neq \sigma'\,(e,\delta,t)\,\} \\ & \cup \{\,\sigma_1\,(e,\tau,t)\,{}^{t}[\,\sigma_2\,] \mid \sigma_1\,(e,\delta,t) \in \mathcal{T}_T[\![\,B_1\,]\!]\ \wedge\ \sigma_2 \in \mathcal{T}_T[\![\,B_2\,]\!]\,\} \\ = & \{\text{ Definition 5.16 }\} \\ & \mathcal{T}_T[\![\,B_1 >> B_2\,]\!]\quad . \end{array}$$

4. $B = B_1 [> B_2$. From the untimed case we know that traces of Γ are either (i) traces of Γ_1 that end with a δ, or (ii) concatenations of traces $\sigma_1 \in T_T(\Gamma_1)$ and $\sigma_2 \in T_T(\Gamma_2)$ where σ_1 does not contain a δ. In Γ asymmetric conflicts are introduced between E_1 and $init(\Gamma_2)$. This means—according to Definition 4.5—that the timing of events in σ_2 should exceed the timing of all events of Γ_1 that have occurred in σ_1. So, we derive:

$$
\begin{aligned}
& T_T(\mathcal{E}_T[\![B_1 [> B_2]\!]) \\
= \quad & \{ \text{ see discussion above } \} \\
& \{ \sigma \in T_T(\Gamma_1) \mid \sigma = \sigma\,(e,\delta,t) \} \cup \\
& \{ \sigma_1\,\sigma_2 \mid \sigma_1 \in T_T(\Gamma_1) \ \wedge\ \sigma_2 \in T_T(\Gamma_2) \ \wedge\ \sigma_1 \neq \sigma_1'\,(e,\delta,t) \ \wedge\ (\forall\, e_i \in \overline{\sigma_2} : t_i \geqslant \mathsf{mx}(\sigma_1)) \} \\
= \quad & \{ \text{ induction hypothesis } \} \\
& \{ \sigma \in \mathcal{T}_T[\![B_1]\!] \mid \sigma = \sigma\,(e,\delta,t) \} \cup \\
& \{ \sigma_1\sigma_2 \mid \sigma_1 \in \mathcal{T}_T[\![B_1]\!] \ \wedge\ \sigma_2 \in \mathcal{T}_T[\![B_2]\!] \ \wedge\ \sigma_1 \neq \sigma_1'\,(e,\delta,t) \ \wedge\ (\forall\, e_i \in \overline{\sigma_2} : t_i \geqslant \mathsf{mx}(\sigma_1)) \} \\
= \quad & \{ \text{ Definition 5.16 } \} \\
& \mathcal{T}_T[\![B_1 [> B_2]\!] \quad .
\end{aligned}
$$

□

5.19. Corollary. $\forall\, B, B_1, B_2 \in \mathsf{PA}_T, t, t' \in \mathsf{Time}$:

$$\left(B \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B_1 \xrightarrow{(e',a',t')}\!\!\!\!\rightarrow B_2 \ \wedge\ t' < t\right) \ \Rightarrow\ (\exists\, B' : B \xrightarrow{(e',a',t')}\!\!\!\!\rightarrow B' \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B_2) \quad .$$

Proof. Directly from Theorems 5.18 and 4.9. □

We now rephrase Theorem 5.18 in the sense of timed event trace equivalence between transition systems. Let $\mathsf{TS}_T(B)$ be the transition system obtained by applying the inference rules of Table 5.1 to B. For $\mathcal{E}_T[\![B]\!]$ a transition system is constructed in the following way.

5.20. Definition. For $\Gamma \in \mathsf{EBES}_T$, the set of derivatives, $der(\Gamma)$, is defined as the smallest set satisfying:

- $\Gamma \in der(\Gamma)$
- $(\Gamma' \in der(\Gamma) \ \wedge\ \Gamma'' = \Gamma'[(e,t)]) \ \Rightarrow\ \Gamma'' \in der(\Gamma)$.

□

States of the transition system for $\mathcal{E}_T[\![B]\!]$ are derivatives of $\mathcal{E}_T[\![B]\!]$ with $\mathcal{E}_T[\![B]\!]$ being the initial state. There is a transition from Γ to Γ' if $\Gamma' = \Gamma[\sigma]$ for trace σ with $\mid \sigma \mid = 1$.

5.21. Definition. For $\Gamma \in \mathsf{EBES}_T$ let $\mathsf{ETS}_T(\Gamma) \triangleq \langle S, L, T, s_0 \rangle$ with

- $S = der(\Gamma)$
- $L = \{ (e,t) \mid \exists\, \Gamma', \Gamma'' \in der(\Gamma) : \Gamma'' = \Gamma'[(e,t)] \}$
- $T = \{ (\Gamma', (e,t), \Gamma'') \mid \Gamma', \Gamma'' \in der(\Gamma) \ \wedge\ \Gamma'' = \Gamma'[(e,t)] \}$
- $s_0 = \Gamma$.

□

Theorem 5.18 implies that the timed event transition systems $\mathsf{TS}_T(B)$ and $\mathsf{ETS}_T(\mathcal{E}_T[\![B]\!])$ are (timed) event trace equivalent. It is easy to check that for each transition $B \xrightarrow{(e,t)}\!\!\!\!\rightarrow B'$ there is a unique way in which this transition is derived from the SOS-rules for $\longrightarrow\!\!\!\!\rightarrow$. Since—in addition—both (timed) event transition systems are deterministic it follows that $\mathsf{TS}_T(B)$ and $\mathsf{ETS}_T(\mathcal{E}_T[\![B]\!])$ are strong (timed) bisimulation equivalent.[1]

5.22. Theorem. $\forall\, B \in \mathsf{PA}_T : \mathsf{TS}_T(B) \sim \mathsf{ETS}_T(\mathcal{E}_T[\![B]\!])$.

Proof. Similar to the proof of Theorem 2.46. □

5.4 An alternative approach for PA_T

This section presents an alternative event-based operational semantics for PA_T which is inspired by the separation of the passage of time (relation $\rightsquigarrow$) and the occurrence of actions (relation $\longrightarrow$) as introduced by Moller & Tofts [105] and Wang [149] and adopted by, amongst others, Bolognesi *et al.* [18] and (partly) Schneider [133].

The transition relations $\rightsquigarrow$ and $\longrightarrow$ transform a pair $\langle B, t\rangle$, where $B \in \mathsf{PA}_T$ and $t \in \mathsf{Time}$. $\langle B, t\rangle$ should be interpreted as behaviour B at time t. Usually one starts with $\langle B, 0\rangle$. $\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$ denotes that B at time t can pass time to B', which is often equal to B, at time t' ($t' \geqslant t$). Thus, time is advanced with an amount of $t'-t$ time units. $\langle B, t\rangle \xrightarrow{(e,a)} \langle B', t\rangle$ means that B at time t performs event e, labelled with action a, and turns into B' (at t).

The relations $\rightsquigarrow$ and $\longrightarrow$ are the smallest relations closed under all inference rules defined below.

Inaction
This behaviour cannot perform any action, i.e., it can perform no $\longrightarrow$ transitions. $\mathbf{0}$ permits any amount of time to pass, remaining $\mathbf{0}$.

$$\boxed{\frac{}{\langle \mathbf{0}, t\rangle \rightsquigarrow \langle \mathbf{0}, t'\rangle} \quad (t' \geqslant t)}$$

Successful termination
$\surd$ can only perform a δ action, and permits any amount of time to pass, remaining $\surd$.

$$\boxed{\frac{}{\langle \surd_\xi, t\rangle \rightsquigarrow \langle \surd_\xi, t'\rangle} \quad (t' \geqslant t) \qquad \frac{}{\langle \surd_\xi, t\rangle \xrightarrow{(\xi,\delta)} \langle \mathbf{0}, t\rangle}}$$

[1] For the sake of brevity, we refrain from formally defining the notion of strong *timed* bisimulation equivalence; its definition is similar to Definition 1.4 labelling transitions also with time labels.

Timed action-prefix
The behaviour $(t)\ a_\xi\,;\,B$ will wait for t time units to become $(0)\ a_\xi\,;\,B$ after which it either permits any amount of time to pass, remaining the same behaviour, or it may perform event (ξ, a) and behave subsequently like B. (Recall that $x \ominus y$ denotes $max(x-y, 0)$ for $x, y \in \mathsf{Time}$.) The fact that $(0)\ a\,;\,B$ may delay is reasonable; if the environment is not possible to engage in a then it should be allowed to delay until the environment is able to engage.

$$\frac{}{\langle (t')\ a_\xi\ ;\ B, t\rangle \rightsquigarrow \langle (t' \ominus (t''-t))\ a_\xi\ ;\ B, t''\rangle} \quad (t'' \geqslant t)$$

$$\frac{}{\langle (0)\ a_\xi\ ;\ B, t\rangle \xrightarrow{(\xi,a)} \langle B, t\rangle}$$

Choice
If the component behaviours B_1 and B_2 permit the passage of time with some amount then so does their choice $B_1 + B_2$. Note that the passage of time does not decide the choice between B_1 and B_2.[2] If B_1 (or B_2) performs event (ξ, a) and evolves into B_1' (B_2') then $B_1 + B_2$ can do the same. Thus,

$$\frac{\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle\ \wedge\ \langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle}{\langle B_1 + B_2, t\rangle \rightsquigarrow \langle B_1' + B_2', t'\rangle}$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle}{\langle B_1 + B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle} \qquad \frac{\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}{\langle B_1 + B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}$$

Enabling
If the first component B_1 permits the passage of time with some amount, then so does the enabling of it with B_2. The action transitions are similar to the untimed case.

$$\frac{\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle}{\langle B_1 >> B_2, t\rangle \rightsquigarrow \langle B_1' >> B_2, t'\rangle}$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle}{\langle B_1 >> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1' >> B_2, t\rangle} \quad (a \neq \delta) \qquad \frac{\langle B_1, t\rangle \xrightarrow{(\xi,\delta)} \langle B_1', t\rangle}{\langle B_1 >> B_2, t\rangle \xrightarrow{(\xi,\tau)} \langle B_2, t\rangle}$$

[2] In the 'standard' jargon of Nicollin & Sifakis [112] our choice construct is classified as a *strong* choice; a weak choice allows the passage of time to decide the choice.

Disrupt
If the component behaviours B_1 and B_2 permit the passage of time with some amount then so does $B_1 \,[> B_2$. The action transitions are similar to the untimed case.

$$\frac{\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \;\wedge\; \langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle}{\langle B_1 \,[> B_2, t\rangle \rightsquigarrow \langle B_1' \,[> B_2', t'\rangle} \qquad \frac{\langle B_1, t\rangle \xrightarrow{(\xi,\delta)} \langle B_1', t\rangle}{\langle B_1 \,[> B_2, t\rangle \xrightarrow{(\xi,\tau)} \langle B_1', t\rangle}$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle}{\langle B_1 \,[> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1' \,[> B_2, t\rangle} \;(a \neq \delta) \qquad \frac{\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}{\langle B_1 \,[> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}$$

Parallel composition
Like for choice, $B_1 \,||_G\, B_2$ allows the passage of time with some amount if both component behaviours permit this. Components of a parallel composition may perform actions not in the synchronization set G^δ independent of each other, while if both B_1 and B_2 can participate in a synchronization action $a \in G^\delta$ then so can their parallel composition.

$$\frac{\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \;\wedge\; \langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle}{\langle B_1 \,||_G\, B_2, t\rangle \rightsquigarrow \langle B_1' \,||_G\, B_2', t'\rangle}$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle}{\langle B_1 \,||_G\, B_2, t\rangle \xrightarrow{((\xi,*),a)} \langle B_1' \,||_G\, B_2, t\rangle} \qquad (a \notin G^\delta)$$

$$\frac{\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}{\langle B_1 \,||_G\, B_2, t\rangle \xrightarrow{((*,\xi),a)} \langle B_1 \,||_G\, B_2', t\rangle} \qquad (a \notin G^\delta)$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle \;\wedge\; \langle B_2, t\rangle \xrightarrow{(\psi,a)} \langle B_2', t\rangle}{\langle B_1 \,||_G\, B_2, t\rangle \xrightarrow{((\xi,\psi),a)} \langle B_1' \,||_G\, B_2', t\rangle} \qquad (a \in G^\delta)$$

Hiding
If B allows the passage of time with a certain amount, then so does $B \setminus G$. Any action that B can perform, can also be performed by $B \setminus G$ whereby actions in set G are turned into silent actions τ.

$$\frac{\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle}{\langle B \setminus G, t\rangle \rightsquigarrow \langle B' \setminus G, t'\rangle}$$

$$\frac{\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle}{\langle B \setminus G, t\rangle \xrightarrow{(\xi,a)} \langle B' \setminus G, t\rangle} \;(a \notin G) \qquad \frac{\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle}{\langle B \setminus G, t\rangle \xrightarrow{(\xi,\tau)} \langle B' \setminus G, t\rangle} \;(a \in G)$$

Relabelling
Like for hiding, if B allows the passage of time with a certain amount, then so does $B[H]$. If B can perform action a and evolve into B', then $B[H]$ can perform $H(a)$ and evolve into $B'[H]$.

$$\frac{\langle B,t\rangle \rightsquigarrow \langle B',t'\rangle}{\langle B[H],t\rangle \rightsquigarrow \langle B'[H],t'\rangle} \qquad \frac{\langle B,t\rangle \xrightarrow{(\xi,a)} \langle B',t\rangle}{\langle B[H],t\rangle \xrightarrow{(\xi,H(a))} \langle B'[H],t\rangle}$$

From the event transition system defined by $\longrightarrow$ we can easily obtain the standard interleaving semantics for PA (as defined in Chapter 1) by omitting time components of tuples $\langle\ldots\rangle$ and the event identifiers from transitions and expressions. When retaining the event identifiers and only omitting the time components we obtain the event-based operational semantics of PA (see Table 2.1). In this sense the presented transition system(s) can be considered to be an *orthogonal* extension of the untimed one.

5.23. Example. Consider $B = (3)\ a_\xi\,;\,(((2)\ b_\chi\,;\,\mathbf{0} + (7)\ c_\psi\,;\,(3)\ d_\eta\,;\,\mathbf{0})\,||_d\,(12)\ d_\rho\,;\,\mathbf{0})$. Then we derive using the inference rules for $\rightsquigarrow$ and $\longrightarrow$ (compare with Example 5.1):

$$\begin{array}{ll}
 & \langle (3)\ a_\xi\,;\,(((2)\ b_\chi\,;\,\mathbf{0} + (7)\ c_\psi\,;\,(3)\ d_\eta\,;\,\mathbf{0})\,||_d\,(12)\ d_\rho\,;\,\mathbf{0}),0\rangle \\
\rightsquigarrow & \{\ \text{(timed action-prefix)}\ \} \\
 & \langle (0)\ a_\xi\,;\,(((2)\ b_\chi\,;\,\mathbf{0} + (7)\ c_\psi\,;\,(3)\ d_\eta\,;\,\mathbf{0})\,||_d\,(12)\ d_\rho\,;\,\mathbf{0}),6\rangle \\
\xrightarrow{(\xi,a)} & \{\ \text{(timed action-prefix)}\ \} \\
 & \langle ((2)\ b_\chi\,;\,\mathbf{0} + (7)\ c_\psi\,;\,(3)\ d_\eta\,;\,\mathbf{0})\,||_d\,(12)\ d_\rho\,;\,\mathbf{0},6\rangle \\
\rightsquigarrow & \{\ \text{(parallel composition), (choice), (timed action-prefix)}\ \} \\
 & \langle ((0)\ b_\chi\,;\,\mathbf{0} + (0)\ c_\psi\,;\,(3)\ d_\eta\,;\,\mathbf{0})\,||_d\,(5)\ d_\rho\,;\,\mathbf{0},13\rangle \\
\xrightarrow{(\psi,c)} & \{\ \text{(par-left), (choice-right), (timed action-prefix)}\ \} \\
 & \langle (3)\ d_\eta\,;\,\mathbf{0}\,||_d\,(5)\ d_\rho\,;\,\mathbf{0},13\rangle \\
\rightsquigarrow & \{\ \text{(parallel composition), (timed action-prefix)}\ \} \\
 & \langle (0)\ d_\eta\,;\,\mathbf{0}\,||_d\,(0)\ d_\rho\,;\,\mathbf{0},22\rangle \\
\xrightarrow{((\eta,\rho),d)} & \{\ \text{(synchronization), (timed action-prefix)}\ \} \\
 & \langle \mathbf{0}\,||_d\,\mathbf{0},22\rangle \quad .
\end{array}$$

Opposed to the transition system based on timed-action transitions, time labels in successive transitions do increase, and as a result all derivations are time-consistent. E.g., for $B = (3)\ a_\xi\,;\,\mathbf{0}\,|||\,(7)\ b_\psi\,;\,\mathbf{0}$ we have $\langle B,0\rangle \xrightarrow{(\psi,b,9)}_* \langle B',9\rangle \overset{(\xi,a,4)}{\not\longrightarrow}_*$, where $\longrightarrow_*$ is defined below. □

$\longrightarrow_*$ is defined as the combination of $\rightsquigarrow$ and $\longrightarrow$.

5.24. Definition. $\langle B,t\rangle \xrightarrow{(e,a,t')}_* \langle B',t'\rangle$ iff $\Big(\exists\, B'' : \langle B,t\rangle \rightsquigarrow \langle B'',t'\rangle \xrightarrow{(e,a)} \langle B',t'\rangle\Big)$. □

Using the relation $\longrightarrow_*$ the notion of timed event trace and a trace derivation relation $\xrightarrow{\sigma}_*$ for timed event trace σ can be defined in the usual way.

5.5 Alternative timed event transition semantics

This section proves the consistency between the denotational semantics $\mathcal{E}_T[\![\ ]\!]$ of PA_T in terms of timed event structures and its operational semantics induced by the inference rules for $\longrightarrow$ and $\rightsquigarrow$. We start by giving an operational characterization of timed event traces of B under $\longrightarrow_*$, and relate this to the denotational characterization of timed traces, $\mathcal{T}_T[\![\, B \,]\!]$ (see Definition 5.16).

We have the following result for timed event traces generated by $\longrightarrow_*$. For parallel composition we use the projections $\pi_1(\sigma)$ and $\pi_2(\sigma)$ for $\sigma \in (S_1 \ \bowtie_G \ S_2)^+$ (rather than *), the set containing all *time-consistent* sequences constructed from elements of the set $S_1 \ \bowtie_G \ S_2$.

5.25. LEMMA. For trace σ, behaviours B, B_1 and B_2, and time t, t'' we have:

1. $\langle \mathbf{0}, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff $\sigma = \varepsilon$, $B' = \mathbf{0}$ and $t' \geqslant t$.
2. $\langle \surd_\xi, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff either
   (i) $\sigma = \varepsilon$, $B' = \surd_\xi$ and $t' \geqslant t$, or
   (ii) $\sigma = (\xi, \delta, t')$, $B' = \mathbf{0}$ and $t' \geqslant t$.
3. $\langle (t)\ a_\xi\ ;\ B, t''\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff either
   (i) $\sigma = \varepsilon$, $B' = (t \ominus (t' - t''))\ a_\xi\ ;\ B$ and $t' \geqslant t''$, or
   (ii) $\sigma = (\xi, a, t_a)\ \sigma'$ with $t_a \geqslant t'' + t$ such that $\langle (t)\ a_\xi\ ;\ B, t''\rangle \xrightarrow{(\xi,a,t_a)}_* \langle B, t_a\rangle$ and $\langle B, t_a\rangle \xrightarrow{\sigma'}_* \langle B', t'\rangle$.
4. $\langle B_1 + B_2, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff either
   (i) $\sigma = \varepsilon \ \wedge\ \langle B_1, t\rangle \xrightarrow{\varepsilon}_* \langle B_1', t'\rangle \ \wedge\ \langle B_2, t\rangle \xrightarrow{\varepsilon}_* \langle B_2', t'\rangle \ \wedge\ B' = B_1' + B_2'$, or
   (ii) $\langle B_1, t\rangle \xrightarrow{\sigma'}_* \langle B_1', t'\rangle \ \wedge\ B' = B_1' \ \wedge\ \sigma = \sigma' \ \wedge\ \sigma \neq \varepsilon$, or
   (iii) $\langle B_2, t\rangle \xrightarrow{\sigma'}_* \langle B_2', t'\rangle \ \wedge\ B' = B_2' \ \wedge\ \sigma = \sigma' \ \wedge\ \sigma \neq \varepsilon$.
5. $\langle B_1 >> B_2, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff either
   (i) $\sigma \neq \sigma_1(e, \delta, t')$, $\langle B_1, t\rangle \xrightarrow{\sigma}_* \langle B_1', t'\rangle$, and $B' = B_1' >> B_2$, or
   (ii) $\sigma = \sigma_1(e, \tau, t'')\sigma_2$, $\langle B_1, t\rangle \xrightarrow{\sigma_1(e,\delta,t'')}_* \langle B_1', t''\rangle$, $\langle B_2, t''\rangle \xrightarrow{\sigma_2}_* \langle B_2', t'\rangle$ and $B' = B_2'$.
6. $\langle B_1 [> B_2, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff either
   (i) $\sigma = \sigma_1$, $\langle B_1, t\rangle \xrightarrow{\sigma_1}_* \langle B_1', t'\rangle$, $\sigma_1 \neq \sigma_1'\,(e, \delta, t')$ and $B' = B_1' [> B_2$, or
   (ii) $\sigma = \sigma_1\,(e, \delta, t')$, $\langle B_1, t\rangle \xrightarrow{\sigma_1\,(e,\delta,t')}_* \langle B_1', t'\rangle$ and $B' = B_1'$, or
   (iii) $\sigma = \sigma_1\,\sigma_2$, $\langle B_1, t\rangle \xrightarrow{\sigma_1}_* \langle B_1', t''\rangle$, $\sigma_1 \neq \sigma_1'(e, \delta, t'')$, and $\langle B_2, t\rangle \xrightarrow{\varepsilon}_* \langle B_2'', t''\rangle$, $\langle B_2'', t''\rangle \xrightarrow{\sigma_2}_* \langle B_2', t'\rangle$, $\sigma_2 \neq \varepsilon$ and $B' = B_2'$.
7. $\langle B_1 \,||_G\, B_2, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff
   $\langle B_1, t\rangle \xrightarrow{\pi_1(\sigma)}_* \langle B_1', t'\rangle$ and $\langle B_2, t\rangle \xrightarrow{\pi_2(\sigma)}_* \langle B_2', t'\rangle$ and $B' = B_1' \,||_G\, B_2'$.
8. $\langle B[H], t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff $\langle B, t\rangle \xrightarrow{\sigma'}_* \langle B'', t'\rangle$ and $B' = B''[H]$ and $\sigma = \sigma'[H]$.
9. $\langle B \setminus G, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff $\langle B, t\rangle \xrightarrow{\sigma'}_* \langle B'', t'\rangle$ and $B' = B'' \setminus G$ and $\sigma = \sigma' \setminus G$.

PROOF. For all syntactical constructs the proof is by induction on the length of σ using the transition rules for $\longrightarrow$ and $\rightsquigarrow$. These proofs are rather laborious, but quite straightforward. Here, we only provide the proof for action-prefix. Consider $(t)\ a_\xi\,;B$ we distinguish between two cases, $\sigma = \varepsilon$ and $\sigma \neq \varepsilon$.

1. For $\sigma = \varepsilon$ we derive

$$\begin{array}{ll} & \langle (t)\ a_\xi\,; B, t''\rangle \xrightarrow{\varepsilon}_* \langle B', t'\rangle \\ \Leftrightarrow & \{\ \text{Definition 5.24}\ \} \\ & \langle (t)\ a_\xi\,; B, t''\rangle \rightsquigarrow \langle B', t'\rangle \\ \Leftrightarrow & \{\ \text{SOS-rules for} \rightsquigarrow\ \} \\ & t' \geqslant t'' \ \wedge\ B' = (t \ominus (t' - t''))\ a_\xi\,; B \quad . \end{array}$$

2. For $\sigma \neq \varepsilon$ it follows immediately from the SOS-rules for $\rightsquigarrow$ and $\longrightarrow$ that $\sigma = (\xi, a, t_a)\ \sigma'$.

$$\begin{array}{ll} & \langle (t)\ a_\xi\,; B, t''\rangle \xrightarrow{(\xi,a,t_a)\ \sigma'}_* \langle B', t'\rangle \\ \Leftrightarrow & \{\ \text{Definition 5.24 and} \xrightarrow{\sigma}_*\ \} \\ & \langle (t)\ a_\xi\,; B, t''\rangle \rightsquigarrow \langle B'', t_a\rangle \xrightarrow{(\xi,a)} \langle B''', t_a\rangle \xrightarrow{\sigma'}_* \langle B', t'\rangle \\ \Leftrightarrow & \{\ \text{see proof just above for } \sigma = \varepsilon\ \} \\ & \langle (t)\ a_\xi\,; B, t''\rangle \rightsquigarrow \langle (t \ominus (t_a - t''))\ a_\xi\,; B, t_a\rangle \xrightarrow{(\xi,a)} \langle B''', t_a\rangle \xrightarrow{\sigma'}_* \langle B', t'\rangle \\ \Leftrightarrow & \{\ \text{SOS-rule for} \longrightarrow \text{implies } t_a \geqslant t{+}t'' \text{ and } B''' = B\ \} \\ & \langle (t)\ a_\xi\,; B, t''\rangle \rightsquigarrow \langle (0)\ a_\xi\,; B, t_a\rangle \xrightarrow{(\xi,a)} \langle B, t_a\rangle \xrightarrow{\sigma'}_* \langle B', t'\rangle \ \wedge\ t_a \geqslant t + t'' \\ \Leftrightarrow & \{\ \text{Definition 5.24}\ \} \\ & \langle (t)\ a_\xi\,; B, t''\rangle \xrightarrow{(\xi,a,t_a)}_* \langle B, t_a\rangle \xrightarrow{\sigma'}_* \langle B', t'\rangle \ \wedge\ t_a \geqslant t + t'' \quad . \end{array}$$

□

5.26. DEFINITION. The set of timed event traces of B at t under $\longrightarrow_*$ is defined as:

$$\mathcal{T}_T^*[\![\, B \,]\!]\, t \triangleq \{\, \sigma \mid \exists\, B', t' : \langle B, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle \,\} \quad .$$

□

The following lemma shows that the set of timed traces of B under $\longrightarrow_*$, $\mathcal{T}_T^*[\![\, B \,]\!]$, corresponds to the time-consistent timed traces obtained from the transition system based on timed-actions, $\mathcal{T}_T[\![\, B \,]\!]$.

5.27. LEMMA. $\forall\, B \in \mathsf{PA}_T, t \in \mathsf{Time} : \mathcal{T}_T^*[\![\, B \,]\!]\, t = \{\, {}^t[\,\sigma\,] \mid \sigma \in \mathcal{T}_T[\![\, B \,]\!] \ \wedge\ tc(\sigma)\,\}$.

PROOF. By induction on the structure of B.
Base: For $B = \mathbf{0}$ we have $\mathcal{T}_T^*[\![\, \mathbf{0} \,]\!]\, t = \{\, \varepsilon \,\}$. From Definition 5.16 we infer that $\mathcal{T}_T[\![\, \mathbf{0} \,]\!] = \{\, \varepsilon \,\}$. Since ${}^t[\,\varepsilon\,] = \varepsilon$ and $tc(\varepsilon)$, the lemma holds for this case. For $B = \surd_\xi$ we have $\mathcal{T}_T^*[\![\, \mathbf{0} \,]\!]\, t = \{\, \varepsilon \,\} \cup \{\, (\xi, \delta, t') \mid t' \geqslant t \,\}$, that is, $\{\, \varepsilon \,\} \cup \{\, {}^t[\,(\xi, \delta, t'')\,] \mid t'' \geqslant 0 \,\}$. From Definition 5.16 the lemma follows directly.
Induction Step: Assume the lemma holds for B_1 and B_2. We provide the proofs for timed action-prefix and enabling. The proofs for the other operators are similar and are omitted here.

1. Timed action-prefix. For this case we derive:

$$
\begin{array}{ll}
 & \mathcal{T}_T^*[\![\,(t'')\ a_\xi\,;\,B\,]\!]\,t \\
= & \{\ \text{Definition 5.26}\ \} \\
 & \{\,\sigma \mid \exists\, B', t' : \langle (t'')\ a_\xi\,;\,B, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle\,\} \\
= & \{\ \text{Lemma 5.25; Definition 5.26}\ \} \\
 & \{\,\varepsilon\,\} \cup \{\,(\xi, a, t_a)\,\sigma' \mid t_a \geqslant t{+}t'' \ \wedge\ \sigma' \in \mathcal{T}_T^*[\![\,B\,]\!]\,t_a\,\} \\
= & \{\ \text{induction hypothesis}\ \} \\
 & \{\,\varepsilon\,\} \cup \{\,(\xi, a, t_a)\,{}^{t_a}[\,\sigma''\,] \mid t_a \geqslant t{+}t'' \ \wedge\ \sigma'' \in \mathcal{T}_T[\![\,B\,]\!] \ \wedge\ tc(\sigma'')\,\} \\
= & \{\ \text{calculus; } tc(\sigma) \Leftrightarrow tc({}^t[\,\sigma\,])\ \} \\
 & \{\,\varepsilon\,\} \cup \{\,(\xi, a, t'_a{+}t)\,{}^{t'_a+t}[\,\sigma''\,] \mid t'_a \geqslant t'' \ \wedge\ \sigma'' \in \mathcal{T}_T[\![\,B\,]\!] \ \wedge\ tc({}^{t'_a}[\,\sigma''\,])\,\} \\
= & \{\ \text{definition of } {}^t[\,\sigma\,]\ \} \\
 & \{\,\varepsilon\,\} \cup \{\,{}^t[\,(\xi, a, t'_a)\,{}^{t'_a}[\,\sigma''\,]\,] \mid t'_a \geqslant t'' \ \wedge\ \sigma'' \in \mathcal{T}_T[\![\,B\,]\!] \ \wedge\ tc((\xi, a, t'_a)\,{}^{t'_a}[\,\sigma''\,])\,\} \\
= & \{\ \text{Definition 5.16; } {}^t[\,\varepsilon\,] = \varepsilon\text{; } tc(\varepsilon)\ \} \\
 & \{\,{}^t[\,\sigma\,] \mid \sigma \in \mathcal{T}_T[\![\,(t'')\ a_\xi\,;\,B\,]\!] \ \wedge\ tc(\sigma)\,\} \quad .
\end{array}
$$

2. Enabling. For this case we derive:

$$
\begin{array}{ll}
 & \mathcal{T}_T^*[\![\,B_1 >> B_2\,]\!]\,t \\
= & \{\ \text{Definition 5.26}\ \} \\
 & \{\,\sigma \mid \exists\, B', t' : \langle B_1 >> B_2, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle\,\} \\
= & \{\ \text{Lemma 5.25; Definition 5.26}\ \} \\
 & \quad \{\,\sigma \mid \sigma \neq \sigma_1\,(e, \delta, t') \ \wedge\ \sigma \in \mathcal{T}_T^*[\![\,B_1\,]\!]\,t\,\} \\
 & \cup\, \{\,\sigma_1\,(e, \tau, t'')\,\sigma_2 \mid \sigma_1\,(e, \delta, t'') \in \mathcal{T}_T^*[\![\,B_1\,]\!]\,t \ \wedge\ \sigma_2 \in \mathcal{T}_T^*[\![\,B_2\,]\!]\,t''\,\} \\
= & \{\ \text{induction hypothesis}\ \} \\
 & \quad \{\,{}^t[\,\sigma\,] \mid {}^t[\,\sigma\,] \neq \sigma_1\,(e, \delta, t') \ \wedge\ \sigma \in \mathcal{T}_T[\![\,B_1\,]\!] \ \wedge\ tc(\sigma)\,\} \\
 & \cup\, \{\,{}^t[\,\sigma'_1\,(e, \tau, t''{-}t)\,]\,{}^{t''}[\,\sigma'_2\,] \mid \sigma'_1\,(e, \delta, t''{-}t) \in \mathcal{T}_T[\![\,B_1\,]\!] \ \wedge\ \sigma'_2 \in \mathcal{T}_T[\![\,B_2\,]\!] \\
 & \qquad\qquad\qquad\qquad\qquad \wedge\ tc(\sigma'_1\,(e, \tau, t''{-}t)) \ \wedge\ tc(\sigma'_2)\,\} \\
= & \{\ \text{calculus}\ \} \\
 & \quad \{\,{}^t[\,\sigma\,] \mid {}^t[\,\sigma\,] \neq \sigma_1\,(e, \delta, t') \ \wedge\ \sigma \in \mathcal{T}_T[\![\,B_1\,]\!] \ \wedge\ tc(\sigma)\,\} \\
 & \cup\, \{\,{}^t[\,\sigma'_1\,(e, \tau, t''{-}t)\,{}^{t''-t}[\,\sigma'_2\,]\,] \mid \sigma'_1\,(e, \delta, t''{-}t) \in \mathcal{T}_T[\![\,B_1\,]\!] \ \wedge\ \sigma'_2 \in \mathcal{T}_T[\![\,B_2\,]\!] \\
 & \qquad\qquad\qquad\qquad\qquad \wedge\ tc(\sigma'_1\,(e, \tau, t''{-}t)\,{}^{t''-t}[\,\sigma'_2\,])\,\} \\
= & \{\ \text{Definition 5.16}\ \} \\
 & \{\,{}^t[\,\sigma\,] \mid \sigma \in \mathcal{T}_T[\![\,B_1 >> B_2\,]\!] \ \wedge\ tc(\sigma)\,\} \quad .
\end{array}
$$

□

Since we know from Theorem 5.18 that $\mathcal{T}_T[\![\,B\,]\!]$ equals the set of timed traces generated from the event structure corresponding to B, $\mathcal{E}_T[\![\,B\,]\!]$, we immediately have

5.28. COROLLARY. $\forall\, B \in \mathsf{PA}_T : \mathcal{T}_T^*[\![\,B\,]\!]\,t = \{\,{}^t[\,\sigma\,] \mid \sigma \in T_T(\mathcal{E}_T[\![\,B\,]\!]) \ \wedge\ tc(\sigma)\,\}$.

PROOF. Straightforward from the previous lemma and Theorem 5.18. □

5.6 Model properties

In this section we prove some properties of our timed transition system based on time- and action-transitions. More precisely, we will prove time determinism, time additivity and persistency (this terminology is adopted from Nicollin & Sifakis [112]).

The first property conforms to the intuition that a process can always evolve into itself by not advancing time.

5.29. THEOREM. For all $B \in \mathsf{PA}_T, t \in \mathsf{Time} : \langle B, t\rangle \rightsquigarrow \langle B, t\rangle$.

PROOF. Straightforward by induction on the structure of B. □

It is easy to verify that the transition system defined by $\longrightarrow$ is deterministic. The transition system defined by $\rightsquigarrow$ is *time deterministic*. This means that the passage of time does not introduce any nondeterminism into the execution of a behaviour.

5.30. THEOREM. *Time determinism*

$$\forall\, B, B', B'' \in \mathsf{PA}_T, t, t' \in \mathsf{Time} : (\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle \;\wedge\; \langle B, t\rangle \rightsquigarrow \langle B'', t'\rangle) \;\Rightarrow\; B' = B''.$$

PROOF. By induction on the structure of B with $\mathbf{0}$, $\surd$, and action-prefix as base cases.
Base : For $B = \mathbf{0}$ and $B = \surd$ the theorem trivially follows from the fact that there is only one SOS-rule for $\rightsquigarrow$ for these cases. For $B = (t'')\ a\,;\, B_1$ we have

$$\begin{array}{ll}
 & \langle (t'')\ a\,;\, B_1, t\rangle \rightsquigarrow \langle B', t'\rangle \;\wedge\; \langle (t'')\ a\,;\, B_1, t\rangle \rightsquigarrow \langle B'', t'\rangle \\
\Rightarrow & \{\text{ Lemma 5.25 }\} \\
 & B' = (t'' \ominus (t'-t))\ a\,;\, B_1 \;\wedge\; B'' = (t'' \ominus (t'-t))\ a\,;\, B_1 \\
\Rightarrow & \{\text{ calculus }\} \\
 & B' = B'' \quad .
\end{array}$$

Induction Step : Assume the theorem holds for B_1 and B_2. We only provide the proof for choice. The proofs for the other constructs are similar and omitted here. For $B = B_1 + B_2$ we derive:

$$\begin{array}{ll}
 & \langle B_1 + B_2, t\rangle \rightsquigarrow \langle B', t'\rangle \;\wedge\; \langle B_1 + B_2, t\rangle \rightsquigarrow \langle B'', t'\rangle \\
\Leftrightarrow & \{\text{ SOS-rules for } \rightsquigarrow \} \\
 & \langle B_1 + B_2, t\rangle \rightsquigarrow \langle B_1' + B_2', t'\rangle \wedge B' = B_1' + B_2' \;\wedge\; \langle B_1 + B_2, t\rangle \rightsquigarrow \langle B_1'' + B_2'', t'\rangle \wedge B'' = B_1'' + B_2'' \\
\Leftrightarrow & \{\text{ Lemma 5.25 }\} \\
 & \quad\;\; \langle B_1, t\rangle \rightsquigarrow \langle B_1', t\rangle \;\wedge\; \langle B_2, t\rangle \rightsquigarrow \langle B_2', t\rangle \;\wedge\; B' = B_1' + B_2' \\
 & \;\wedge\; \langle B_1, t\rangle \rightsquigarrow \langle B_1'', t\rangle \;\wedge\; \langle B_2, t\rangle \rightsquigarrow \langle B_2'', t\rangle \;\wedge\; B'' = B_1'' + B_2'' \\
\Rightarrow & \{\text{ induction hypothesis }\} \\
 & B_1' = B_1'' \;\wedge\; B_2' = B_2'' \;\wedge\; B' = B_1' + B_2' \;\wedge\; B'' = B_1'' + B_2'' \\
\Rightarrow & \{\text{ calculus }\} \\
 & B' = B'' \quad .
\end{array}$$

□

The next property (sometimes referred to as *action persistency*) conforms to the intuition that the progress of time should not suppress the ability to perform an action. That is,

5.31. THEOREM. *Action persistency*

$$\forall\, B, B' \in \mathsf{PA}_T, t, t' \in \mathsf{Time} : \Big(\langle B, t\rangle \xrightarrow{(e,a)} \ \wedge\ \langle B, t\rangle \rightsquigarrow \langle B', t'\rangle\Big) \ \Rightarrow\ \langle B', t'\rangle \xrightarrow{(e,a)} .$$

PROOF. By induction on the structure of B.
Base: For $B = \mathbf{0}$ the theorem trivially holds as $\mathbf{0}$ cannot perform any $\longrightarrow$ transitions. For $B = \surd_\xi$ the theorem holds as $\langle \surd_\xi, t\rangle$ for any t can perform δ and $\langle \surd_\xi, t\rangle \rightsquigarrow \langle \surd_\xi, t'\rangle$, for any $t' \geqslant t$. Now consider $B = (t'')\ b_\xi\ ; B_1$. For this case we derive

$$\begin{array}{ll}
& \langle (t'')\ b_\xi\ ; B_1, t\rangle \xrightarrow{(e,a)} \ \wedge\ \langle (t'')\ b_\xi\ ; B_1, t\rangle \rightsquigarrow \langle B', t'\rangle \\
\Leftrightarrow & \{\ \text{SOS-rules for action-prefix}\ \} \\
& \langle (0)\ b_\xi\ ; B_1, t\rangle \xrightarrow{(\xi,b)} \ \wedge\ \langle (0)\ b_\xi\ ; B_1, t\rangle \rightsquigarrow \langle (0 \ominus (t'{-}t))\ b_\xi\ ; B_1, t'\rangle \\
\Rightarrow & \{\ \} \\
& \langle (0)\ b_\xi\ ; B_1, t\rangle \rightsquigarrow \langle (0)\ b_\xi\ ; B_1, t'\rangle \\
\Leftrightarrow & \{\ \text{SOS-rule}\ (\longrightarrow)\ \text{for action-prefix}\ \} \\
& \langle (0)\ b_\xi\ ; B_1, t'\rangle \xrightarrow{(\xi,b)} \quad .
\end{array}$$

Induction Step: Assume the theorem holds for B_1 and B_2. We consider the proof for parallel composition; the proofs for the other constructs are similar and omitted.

$$\begin{array}{ll}
& \langle B_1 \,||_G\, B_2, t\rangle \xrightarrow{(e,a)} \ \wedge\ \langle B_1 \,||_G\, B_2, t\rangle \rightsquigarrow \langle B', t'\rangle \\
\Leftrightarrow & \{\ \text{distinguish between } a \in G^\delta \text{ and } a \notin G^\delta;\ \text{SOS-rule } (\rightsquigarrow) \text{ for } ||_G\ \} \\
& \quad (\langle B_1, t\rangle \xrightarrow{(e_1,a)} \ \wedge\ \langle B_2, t\rangle \xrightarrow{(e_2,a)} \ \wedge\ e = (e_1, e_2)\ \wedge\ a \in G^\delta) \\
& \vee\ (\langle B_1, t\rangle \xrightarrow{(e_1,a)} \ \wedge\ e = (e_1, *)\ \wedge\ a \notin G^\delta)\ \vee\ (\langle B_2, t\rangle \xrightarrow{(e_2,a)} \ \wedge\ e = (*, e_2)\ \wedge\ a \notin G^\delta) \\
& \wedge\ \langle B_1 \,||_G\, B_2, t\rangle \rightsquigarrow \langle B_1' \,||_G\, B_2', t\rangle \\
\Rightarrow & \{\ \text{SOS-rule } (\rightsquigarrow) \text{ for } ||_G\ ;\ \text{induction hypothesis}\ \} \\
& \quad (\langle B_1', t'\rangle \xrightarrow{(e_1,a)} \ \wedge\ \langle B_2', t'\rangle \xrightarrow{(e_2,a)} \ \wedge\ e = (e_1, e_2)\ \wedge\ a \in G^\delta) \\
& \vee\ (\langle B_1', t'\rangle \xrightarrow{(e_1,a)} \ \wedge\ e = (e_1, *)\ \wedge\ a \notin G^\delta)\ \vee\ (\langle B_2', t'\rangle \xrightarrow{(e_2,a)} \ \wedge\ e = (*, e_2)\ \wedge\ a \notin G^\delta) \\
\Leftrightarrow & \{\ \text{SOS-rule}\ (\longrightarrow)\ \text{for}\ ||_G\ \} \\
& \langle B_1' \,||_G\, B_2', t'\rangle \xrightarrow{(e,a)} \quad .
\end{array}$$

□

Finally, a process at time t is able to evolve to a certain time t' iff it can evolve to any time instant in between t and t'. This property which is often referred to as *time additivity* (or *time continuity*) is formally stated as[3]

5.32. THEOREM. *Time additivity*

$\forall\, B, B' \in \mathsf{PA}_T, t, t', t'' \in \mathsf{Time}$:

$$\langle B, t\rangle \rightsquigarrow \langle B', t{+}(t'{+}t'')\rangle \Longleftrightarrow (\exists\, B'' : \langle B, t\rangle \rightsquigarrow \langle B'', t{+}t'\rangle \rightsquigarrow \langle B', t{+}(t'{+}t'')\rangle).$$

PROOF. Straightforward by induction on the structure of B. □

[3] Lynch & Vaandrager [98] adopt for their timed I/O-automata a stronger notion that says that there must be a *trajectory* of consistent states through the interval $[t, t']$. Since our timed transition system satisfies the image-finiteness condition (i.e., for any B and t there are at most finitely many B' such that $\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$) it follows from Jeffrey *et al.* [79] that our model also satisfies this stronger trajectory condition.

5.7 Related work

To our knowledge this chapter constitutes the first attempt to relate a causality-based semantics and an (event-based) operational model in a *timed* setting. For the untimed case several related approaches have been published to relate a causality-based semantics to an operational one [10, 26, 95]. These investigations differ from our work in particular in the causality-based model, the language at hand, and the type of consistency relation between the two types of semantics. The relation between the approach we followed and the work of Boudol & Castellani [26, 23] (for finite CCS and flow event structures) is discussed at length in Langerak [89].

Baier & Majster-Cederbaum [10] prove the consistency between an operational semantics for theoretical CSP (TCSP) and a compositional true concurrency semantics based on labelled prime event structures. They show that the 'interleaved view' of the event structure semantics—obtained by considering remainders of prime event structures after the execution of a single event—is (weak) bisimilar to the operational semantics of TCSP. An identical technique was used by Loogen & Goltz [95] but they studied TCSP without recursion. We will consider consistency for recursion in Chapter 10.

Degano *et al.* [42] proposed an approach to prove the consistency of an operational noninterleaving semantics of CCS (with guarded recursion) and a denotational semantics based on labelled prime event structures. From the operational semantics an occurrence net is derived which is shown—using the well-known connection between this class of nets and event structures by Nielsen *et al.* [114]—to be equal to the event structure obtained in the denotational way.

On relating operational and denotational models in a timed setting we mention the work of Schneider [133] and Murphy [107]. [133] provides an operational semantics of timed CSP, a mature timed extension of CSP, and studies the relation of this semantics with an (interleaved) denotational model for timed CSP based on timed failures. [107] introduces a timed process algebra where actions are assumed to have a fixed duration. Murphy provides a true concurrent operational semantics, based on timed asynchronous transition systems, and sketches the relation with timed Petri nets.

5.8 Conclusions

In this chapter we have introduced two event-based operational semantics for PA_T which keep track of timed action occurrences (that is, timed events).

The first timed operational semantics is based on timed-actions (relation $\longrightarrow\!\!\!\!\rightarrowtail$) and is a straightforward generalization of the untimed event transition system for PA, see Chapter 2. Consequently, a natural and minimal extension of the standard operational semantics for PA (as introduced in Chapter 1) is obtained. (Notwithstanding Bolognesi *et al.* [19] who conclude that it 'would seem particularly difficult' to obtain a natural, or what they call conservative, extension of an untimed process algebra like LOTOS without a clear separation between time and action transition rules.) One of the features of the timed-action model is the absence of actions/transitions that represent solely the passage of time. Here time is dealt with in a way comparable to physical models, viz. by means of parameterization.

The model based on timed-actions allows for the generation of ill-timed traces like in Aceto & Murphy [1, 2]. Recently, Gorrieri *et al.* [56] proposed a timed process algebra with the TCSP parallel operator that also includes ill-timed traces. In the proposals of Aceto & Murphy and Gorrieri *et al.* sub-processes have their independent local clock, and since local clocks are only synchronized at interaction, ill-timedness appears. We believe that the operational semantics presented in this chapter is simpler by avoiding local clocks.

Ill-timedness is a phenomenon that is sometimes explicitly avoided by others (like in real-time ACP of Baeten & Bergstra [7] and TIC of Quemada *et al.* [123]), since the precedence of timed events in the trace does not reflect the order in time. To our opinion ill-timed traces are not that obscure—we have shown earlier that for each ill-timed trace there exists a corresponding time-consistent trace with the same timed events—and we think that the avoidance of them leads to a more complicated operational semantics. We remark that the operational semantics of Table 5.1 can easily be adapted such that only time-consistent traces are generated, by replacing the rule for independent parallelism by

$$\frac{B_1 \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B_1'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,*),a,t)}\!\!\!\!\rightarrow B_1' \,||_G\, {}^{t}\{\, B_2 \,\}} \quad (a \notin G^{\delta})$$

and similar for the symmetric case.

The second transition system is inspired by the separation of the passage of time (relation $\rightsquigarrow$) and the occurrence of actions (relation $\longrightarrow$) as introduced by Moller & Tofts [105] and Wang [149] and adopted by several others [18, 133]. It turns out that the transition system for $\longrightarrow$ is identical to the untimed transition model presented in Chapter 2. That is, time is added in a completely orthogonal way. This model allows for the generation of well-timed traces only and will be used in Chapter 6 where the notion of urgency is discussed.

The compatibility of both event-based operational semantical models with respect to the causality-based semantics for PA_T provided in Chapter 4 has been investigated. The timed-action model generates the same set of timed traces for behaviour B as the causality-based semantics, whereas for the transition model induced by $\rightsquigarrow$ and $\longrightarrow$ this holds when only considering time-consistent traces. This result provides the basis for proving that the timed-action model and the 'interleaving view' of the causality-based semantics are strong bisimulation equivalent. Since the second transition model forces derivations to be time-consistent, a similar result for this model does not hold. The main features of the interleaving models are their simplicity and compatibility with the standard interleaving semantics of PA, and the untimed event transition system of Langerak (see Chapter 2). We consider these aspects to provide evidence for the adequacy of our timed event structures model.

6 The urgency module

This chapter introduces the concept of urgent events—roughly speaking, events that are forced to occur once they are enabled—in timed event structures. Typically an urgent event 'guards' the occurrence time of an alternative event in the sense that this other event is prevented from happening after a particular time instant. Timeout mechanisms are well-known urgent phenomena. It is investigated how the theory of Chapter 4 carries over to this new model, referred to as urgent event structures. The timed process algebra PA_T is extended with an urgency operator that forces (local or synchronized) actions to happen in an urgent fashion. Urgent event structures are used as a vehicle to provide a denotational causality-based semantics for this formalism. In the spirit of Chapter 5 a consistent event-based operational semantics based on a separation of the passage of time and the occurrence of actions is presented.

6.1 Introduction

In realistic designs one often encounters events that once enabled—i.e., their causal predecessors have occurred and their timing constraints are respected—are forced to occur, provided they are not disabled by other events. Typically such events are timeout mechanisms that guard the occurrence time of other events (like receiving an acknowledgement message) in the sense that they prevent these events from happening after a certain time instant. We call such events *urgent*. Urgent events are graphically denoted as open dots, nonurgent events as closed dots (as before).

To provide a better understanding of our intuition consider, for example, a timer process that is started once a message m is transmitted (represented by event *send*), and assume that it is reasonable to expect an acknowledgment from 3 time units (event *receive*) on since m was transmitted. When after 5 time units, say, the expected acknowledgement message is not yet received it is assumed that some error occurred; at that time the timer will expire (event *timeout*) and a retransmission of m is initiated (not modelled explicitly here). Figure 6.1 depicts an event structure that models this situation. The interpretation is as follows. Event *send* may happen from the start of the system; no timing constraint is imposed on its occurrence. Once event *send* has appeared either *receive* or *timeout* can happen. Event *receive* can happen between 3 and 5 time units after *send*; if not, event *timeout* happens at exactly 5 time units after *send*. At 5 time units after *send* a nondeterministic choice appears between events *timeout* and *receive*. Such timeout mechanisms are sometimes referred to as 'weak'

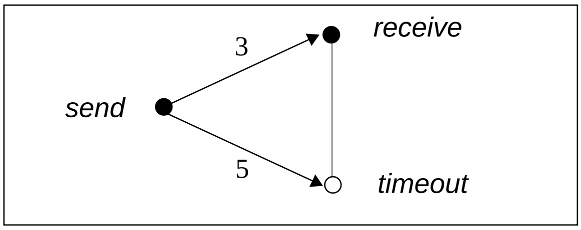


Figure 6.1: Timer example using urgent events.

timeouts, as opposed to 'strong' timeouts where in the timer example at time 5 event *receive* would already become impossible [112].

In this chapter we equip timed event structures, as introduced in Chapter 4, with the notion of urgent events. Section 6.2 introduces the notion of urgent event structures, and investigates how the theory of Chapter 4 carries over to the urgent setting. In Section 6.3 the temporal process algebra PA_T is enriched with an urgency operator, denoted $\mathcal{U}_U()$, that forces (local or synchronized) actions (in U) to happen urgently. A denotational causality-based semantics is provided for the resulting timed process algebra, called PA_U, and is related to a consistent event-based operational semantics. Due to the presence of urgent actions the consistency proof is more involved than for the nonurgent case as treated in Chapter 6. Therefore, this consistency proof is divided into two parts. In Section 6.4 we prove that the way in which urgency is dealt with in the operational semantics of PA_U corresponds to our intuition. Subsequently, in Section 6.5 the actual consistency proof is carried out (in three steps). Section 6.6 relates PA_U to some proposals in the literature that incorporate urgency in a timed process algebra. Section 6.7 summarizes the technical results.

6.2 Urgent event structures

An urgent event structure is a timed event structure in which a distinction is made between nonurgent and urgent events. Urgency is modelled by a predicate $\mathcal{U}$ on events—$\mathcal{U}(e)$ is true if and only if e is urgent.

6.1. DEFINITION. (*Urgent event structure*)

An *urgent* event structure is a tuple $\langle \Gamma, \mathcal{U} \rangle$ with Γ a timed event structure and $\mathcal{U} : E \longrightarrow \mathsf{Bool}$, the *urgency* predicate. $\square$

We use Ψ, possibly subscripted and/or primed, to denote an urgent event structure and EBES_U to denote the universe of urgent event structures. In this chapter we consider urgent event structures with a finite number of events; infinite structures are considered in Chapter 10.

6.2.1 Timed event traces

For convenience we recall the definition of the auxiliary function time:

6.2. DEFINITION. For σ a sequence of timed events $(e_1, t_1) \dots (e_n, t_n)$ with $e_i \in E$, $t_i \in \mathsf{Time}$, for $0 < i \leqslant n$, and $e \in \mathsf{en}([\sigma])$, let

$$\begin{aligned} \mathsf{time}(\sigma, e) &\triangleq Max(\{\, \mathcal{D}(e) \,\} \cup H_1 \cup H_2) \text{ where} \\ &H_1 = \{\, t + t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \{\, e_j \,\} \,\} \text{ and} \\ &H_2 = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow e \,\} \quad . \end{aligned}$$

□

As a next step we generalize the notion of timed event trace (cf. Definition 4.5) towards the urgent case.

6.3. DEFINITION. (*Timed event trace (revisited)*)

A *timed event trace* of urgent event structure $\Psi = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ is a sequence σ of timed events $(e_1, t_1) \dots (e_n, t_n)$ with $e_i \in E$, $t_i \in \mathsf{Time}$, for $0 < i \leqslant n$, satisfying

1. $e_1 \dots e_n \in T(\mathcal{E})$
2. $\forall\, i : (\neg\, \mathcal{U}(e_i) \;\Rightarrow\; t_i \geqslant \mathsf{time}(\sigma_i, e_i)) \;\wedge\; (\mathcal{U}(e_i) \;\Rightarrow\; t_i = \mathsf{time}(\sigma_i, e_i))$
3. $\forall\, i, e : e \in \mathsf{en}([\sigma_i]) \;\wedge\; \mathcal{U}(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}(\sigma_i, e)$
4. $\forall\, i, j : i < j \;\Rightarrow\; t_i \leqslant t_j \quad .$

$C \subseteq E \times \mathsf{Time}$ is a timed configuration iff there is a timed event trace σ such that $C = \overline{\sigma}$.

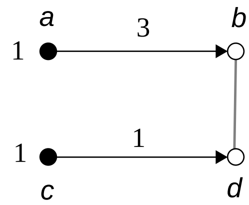
□

$T_U(\Psi)$ denotes the set of timed event traces of Ψ and $C_U(\Psi)$ its set of timed configurations.

According to the first constraint we should obtain an (untimed) event trace of the corresponding event structure $\mathcal{E}$ when we omit the times from a timed event trace. The second constraint requires correct times to be associated to events in σ—ordinary events can happen at any moment from the time they are enabled and urgent events can happen only as soon as they are enabled, they cannot be further delayed.

These two constraints do, however, not take into account the fact that urgent events may prevent other events to occur after a certain time. For instance, according to the first two constraints, Figure 6.1 would have event trace $(\mathit{send}, 3)\,(\mathit{receive}, 9)$ whereas if event *receive* has not happened before time instant 8, the *timeout* should have occurred. Thus $(\mathit{send}, 3)\,(\mathit{receive}, 9)$ should not be considered a legal timed event trace. The third constraint takes this matter into account. It says that σ_i may be extended with (e_i, t_i) iff there is no urgent event enabled after σ_i that could occur at any time earlier than t_i.

The fourth constraint requires timed event traces to be time-consistent. The reason for this is that urgency is an intrinsically *global* property: the fact that some event e is urgent influences for events, which seem at first sight completely independent of e, the ability to appear at a certain time instant. So, in order to decide whether an event may happen it is necessary to know in the entire system which events have happened already (in time). For instance, according to the first three constraints the urgent event structure



would have timed event trace $(e_a, 1)\,(e_b, 4)\,(e_c, 2)$, whereas if e_c happens at time 2 urgent event e_d is forced at time 3 and should disable the occurrence of e_b.

6.4. EXAMPLE. For the following sequences of timed events the conditions are given under

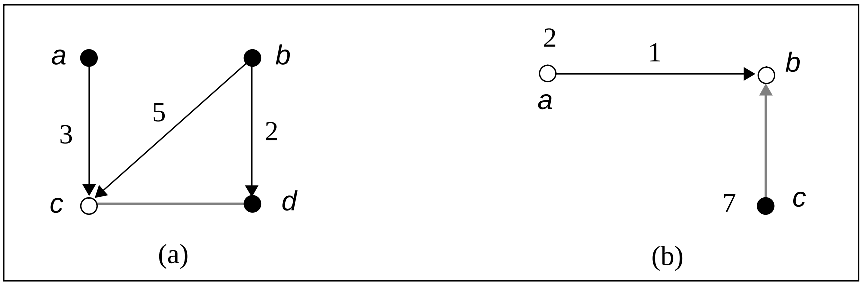


Figure 6.2: Some example urgent event structures.

which they are timed event traces of Figure 6.2(a):

$$(e_a, t_a)\,(e_b, t_b)\,(e_d, t_d) \quad \text{if} \quad t_a \leqslant t_b \ \wedge\ t_b{+}2 \leqslant t_d \leqslant max(t_a{+}3, t_b{+}5), \text{ and}$$
$$(e_a, t_a)\,(e_b, t_b)\,(e_c, t_c) \quad \text{if} \quad t_a \leqslant t_b \ \wedge\ t_c = max(t_a{+}3, t_b{+}5).$$

The only maximal timed event trace of Figure 6.2(b) is $(e_a, 2)\,(e_b, 3)$. In this urgent event structure event e_c can never happen since after the occurrence of e_a (which will be forced at time 2) e_b will occur (at time 3), so excluding e_c. Thus, e_a excludes e_c though they seem to be completely independent! It appears that the asymmetric conflict between e_c and e_b 'propagates back' to an asymmetric conflict between e_a and e_c. □

Now consider Section 4.2.3. In that section we proved that timed event traces having the same timed events constitute a lattice with a least element. It can easily be verified that in presence of urgent events $\langle [\sigma]_\sim, \preccurlyeq \rangle$ is still a poset with a least element. That is, we can still construct chains of event traces (more precisely, equivalence classes of traces) under $\preccurlyeq$ with a fast event trace as least element. The lattice construction in Section 4.2.3 does, however, no longer apply, since it cannot be guaranteed that the *lub* and *glb* are again timed event traces of the event structure at hand. Consider, for example, the urgent event structures of Figure 6.3. (a) has traces $(e_a, 0)\,(e_b, 1)\,(e_c, 2)$ and $(e_b, 0)\,(e_a, 1)\,(e_c, 2)$, but the *lub* of these traces $(e_a, 0)\,(e_b, 0)\,(e_c, 2)$ is not a legal trace. Similarly, (b) has traces $(e_a, 1)\,(e_b, 1)\,(e_c, 3)$ and $(e_a, 0)\,(e_c, 1)\,(e_b, 3)$, but the *glb* of these traces $(e_a, 0)\,(e_b, 3)\,(e_c, 3)$ is not a legal trace.

6.2.2 Families of lposets

This section characterizes the lposets of an urgent event structure. For timed event structures we used an operational scheme by generating lposets from timed event traces. This procedure

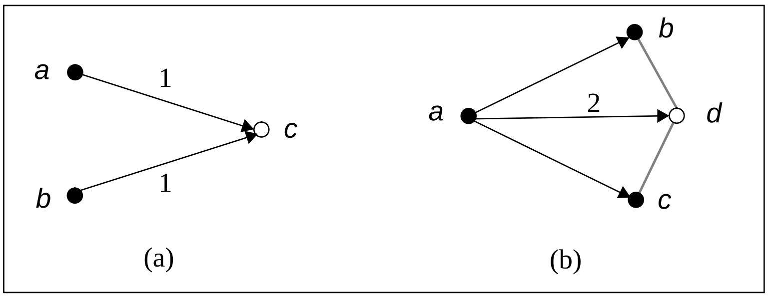


Figure 6.3: Structures for which (a) *lub* and (b) *glb* are not traces.

does not work for urgent event structures. E.g., the urgent event structures

have identical timed event traces, and consequently, would have identical lposets if we would deduce lposets out of traces. We, therefore, take another route and associate to a timed configuration an lposet in the same intensional way as in Chapter 2 for the untimed case.

6.5. DEFINITION. For $C \in C_U(\Psi)$ let $\prec_C \subseteq C \times C$ be the smallest relation satisfying, for all $(e_i, t_i), (e_j, t_j) \in C$:

1. $(\exists\, X \subseteq E : e_i \in X \;\wedge\; X \overset{t}{\mapsto} e_j) \;\Rightarrow\; (e_i, t_i) \prec_C (e_j, t_j)$
2. $e_i \rightsquigarrow e_j \;\Rightarrow\; (e_i, t_i) \prec_C (e_j, t_j)$.

□

Let $\prec_C^*$ be the reflexive and transitive closure of $\prec_C$ and let the labelling of (e, t) equal $l(e)$.

6.6. LEMMA. $\forall\, \sigma \in T_U(\Psi) : \prec_{\overline{\sigma}}^* \subseteq <_\sigma^*$.

PROOF. Suppose $\sigma \in T_U(\Psi)$ and let $C = \overline{\sigma}$. Let $(e_i, t_i), (e_j, t_j) \in C$ such that $(e_i, t_i) \prec_C (e_j, t_j)$. According to Definition 6.5 this can only be because either

1. $\exists\, X \subseteq E : e_i \in X \;\wedge\; X \overset{t}{\mapsto} e_j$. Then by definition of event trace we have $X \cap \overline{[\sigma_j]} \neq \varnothing$. Suppose $X \cap \overline{[\sigma_j]} = \{\, e_k \,\}$. If $e_k \neq e_i$ then it follows from the stability constraint that $e_k \rightsquigarrow e_i$ and $e_i \rightsquigarrow e_k$. Since $[\sigma]$ is an event trace then $e_k <_{[\sigma]} e_i \;\wedge\; e_i <_{[\sigma]} e_k$, which is a contradiction. So, $e_i = e_k$ and $(e_i, t_i) <_\sigma (e_j, t_j)$.
2. $e_i \rightsquigarrow e_j$. Then, by the definition of event trace, $(e_i, t_i) <_\sigma (e_j, t_j)$.

This proves $\prec_{\overline{\sigma}} \subseteq <_\sigma$ and implies that $\prec_{\overline{\sigma}}^* \subseteq <_\sigma^*$. □

Given this lemma it is now easy to verify that $\prec_C^*$ is a partial order on C.

6.7. COROLLARY. $\langle C, \prec_C^* \rangle$ is a poset.

PROOF. Similar to the proof of Corollary 2.21. □

The family of lposets of Ψ, denoted $L_U(\Psi)$, is defined as the set of all lposets corresponding to its timed configurations.

6.8. DEFINITION. (*Lposets of an urgent event structure*)

For $\Psi \in \mathsf{EBES}_U : L_U(\Psi) \triangleq \{ \langle C, \prec^*_C, l \restriction C \rangle \mid C \in C_U(\Psi) \}$. $\square$

6.9. THEOREM. $\forall\, \Psi, \Psi' \in \mathsf{EBES}_U : L_U(\Psi) = L_U(\Psi') \;\Rightarrow\; T_U(\Psi) = T_U(\Psi')$.

PROOF. Straightforward and omitted. $\square$

A few remarks concerning the relationship between the lposets of Ψ and the lposets of its untimed equivalent $\mathcal{E}$ are in order. The lposets of a timed event structure are equal to those of $\mathcal{E}$, see Theorem 4.21. For urgent event structures this does not hold, since some events may not occur at all because an urgent event prevents them to happen. Since this phenomenon is absent in $\mathcal{E}$ there does not need to be a timed configuration C for each configuration in $C(\mathcal{E})$.

6.2.3 Urgent remainder

The notion of timed remainder (cf. Definition 4.22) can easily be extended by incorporating urgent events—an event in the remainder of Ψ is urgent iff it is an urgent event in Ψ.

6.10. DEFINITION. (*Urgent remainder*)

The *urgent remainder* of urgent event structure $\Psi = \langle \Gamma, \mathcal{U} \rangle$ after timed event σ is $\Psi[\sigma] = \langle \Gamma', \mathcal{U}' \rangle$ where $\Gamma' = \Gamma[\sigma] = \langle (E', \rightsquigarrow', \mapsto', l'), \mathcal{D}', \mathcal{T}' \rangle$, and $\mathcal{U}' = \mathcal{U} \restriction E'$. $\square$

In order to prove the correctness of the urgent remainder it is convenient to introduce the following lemmata. Consider Ψ and let σ be a timed event trace of Ψ. Assume σ' is a timed event trace of Ψ after σ, $\Psi[\sigma]$. Then event e is enabled in $\Psi[\sigma]$ after the execution of σ' iff it is enabled in Ψ after the execution of $\sigma\,\sigma'$. This is stated in Lemma 6.11. In addition, the time at which e can occur in $\Psi[\sigma]$ after σ' equals the time at which it can occur in Ψ after $\sigma\,\sigma'$. This is stated in Lemma 6.12.

6.11. LEMMA. For $\sigma \in T_U(\Psi)$ and $\sigma' \in T_U(\Psi[\sigma])$ we have:

$$\forall\, 0 < i \leqslant \mid \sigma' \mid : \mathsf{en}_{\Psi[\sigma]}([\sigma'_i]) = \mathsf{en}_{\Psi}([\sigma\,\sigma'_i]) \quad .$$

PROOF. Similar to the proof of Lemma 6.12 and omitted. $\square$

6.12. LEMMA. For $\sigma \in T_U(\Psi)$ and $\sigma' \in T_U(\Psi[\sigma])$ we have:

$$\forall\, 0 < i \leqslant \mid \sigma' \mid, e \in \mathsf{en}_{\Psi[\sigma]}([\sigma'_i]) : \mathsf{time}_{\Psi[\sigma]}(\sigma'_i, e) = \mathsf{time}_{\Psi}(\sigma\,\sigma'_i, e) \quad .$$

PROOF. Assume $\sigma \in T_U(\Psi)$ and $\sigma' \in T_U(\Psi[\sigma])$. Let $\Psi = \langle (E, \rightsquigarrow, \mapsto, l), \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ and $\Psi[\sigma] = \Psi' = \langle (E', \rightsquigarrow', \mapsto', l'), \mathcal{D}', \mathcal{T}', \mathcal{U}' \rangle$. Let $0 < i \leqslant \mid \sigma' \mid$ and $e \in \mathsf{en}_{\Psi'}([\sigma'_i])$. ($\mathsf{time} = \mathsf{time}_\Psi$ and $\mathsf{time}' = \mathsf{time}_{\Psi'}$.)

$\quad \mathsf{time}'(\sigma'_i, e)$

$= \quad \{$ definition of time $\}$

$\quad Max(\{ \mathcal{D}'(e) \} \cup H'_1 \cup H'_2)$ where

$$
\begin{array}{rl}
& \quad H_1' = \{\, t{+}t_j \mid \exists\, X \subseteq E' : X \overset{t}{\mapsto}{}' e \;\wedge\; X \cap \overline{[\sigma_i']} = \{\, e_j \,\}\,\} \text{ and} \\
& \quad H_2' = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma_i']} : e_j \rightsquigarrow' e \,\} \\
= & \{ \text{ Definition 4.22 } \} \\
& Max(\{\, Max(\{\, \mathcal{D}(e) \,\} \cup H_1 \cup H_2) \,\} \cup H_1' \cup H_2') \text{ where} \\
& \quad H_1' = \{\, t{+}t_j \mid \exists\, X \subseteq E' : X \overset{t}{\mapsto}{}' e \;\wedge\; X \cap \overline{[\sigma_i']} = \{\, e_j \,\}\,\} \text{ and} \\
& \quad H_1 = \{\, t{+}t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \{\, e_j \,\}\,\} \text{ and} \\
& \quad H_2' = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma_i']} : e_j \rightsquigarrow' e \,\} \text{ and} \\
& \quad H_2 = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow e \,\} \\
= & \{ \text{ calculus } \} \\
& Max(\{\, \mathcal{D}(e) \,\} \cup H_1' \cup H_1 \cup H_2' \cup H_2) \text{ where} \ldots \text{ as above } \ldots \\
= & \{\ H_i \cup H_i' = H_i'' \ (i{=}1,2) \text{ (see below) } \} \\
& Max(\{\, \mathcal{D}(e) \,\} \cup H_1'' \cup H_2'') \text{ where} \\
& \quad H_1'' = \{\, t{+}t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma\,\sigma_i']} = \{\, e_j \,\}\,\} \text{ and} \\
& \quad H_2'' = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma\,\sigma_i']} : e_j \rightsquigarrow e \,\} \\
= & \{ \text{ definition } \mathsf{time}\text{; Lemma 6.11 } \} \\
& \mathsf{time}(\sigma\,\sigma_i', e) \quad .
\end{array}
$$

The proof that $H_1 \cup H_1' = H_1''$ is presented below. The proof for $H_2 \cup H_2' = H_2''$ is similar, but simpler, and is omitted.

$$
\begin{array}{rl}
& \quad \{\, t{+}t_j \mid \exists\, X \subseteq E' : X \overset{t}{\mapsto}{}' e \;\wedge\; X \cap \overline{[\sigma_i']} = \{\, e_j \,\}\,\} \\
& \cup \{\, t{+}t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \{\, e_j \,\}\,\} \\
= & \{ \text{ Definition 2.28 } \} \\
& \quad \{\, t{+}t_j \mid \exists\, X \subseteq E' : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \varnothing \;\wedge\; X \cap \overline{[\sigma_i']} = \{\, e_j \,\}\,\} \\
& \cup \{\, t{+}t_j \mid \exists\, X \subseteq E' : X \overset{t}{\mapsto} e \;\wedge\; X = \varnothing \;\wedge\; X \cap \overline{[\sigma_i']} = \{\, e_j \,\}\,\} \\
& \cup \{\, t{+}t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \{\, e_j \,\}\,\} \\
= & \{\ E' \subseteq E;\ X \cap \overline{[\sigma]} = \varnothing \ \} \\
& \quad \{\, t{+}t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \varnothing \;\wedge\; X \cap \overline{[\sigma\,\sigma_i']} = \{\, e_j \,\}\,\} \\
& \cup \{\, t{+}t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \{\, e_j \,\}\,\} \\
= & \{ \text{ calculus } \} \\
& \{\, t{+}t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma\,\sigma_i']} = \{\, e_j \,\}\,\} \quad .
\end{array}
$$

□

We now have the following correctness result for the remainder of an urgent event structure. Note that the correctness criterion is identical to that of timed remainders (cf. Theorem 4.24) except that we require $\sigma\,\sigma'$ to be time-consistent.

6.13. THEOREM. *Correctness of urgent remainder*

For $\sigma \in T_U(\Psi)$ and σ' a sequence of timed events satisfying $tc(\sigma\,\sigma')$:

1. $\sigma' \in T_U(\Psi[\sigma]) \iff \sigma\,\sigma' \in T_U(\Psi)$
2. $\sigma' \in T_U(\Psi[\sigma]) \;\Rightarrow\; L_U(\overline{\sigma})$ is a prefix of $L_U(\overline{\sigma\,\sigma'})$.

PROOF. Let $\Psi = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ with $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$ and $\Psi[\sigma] = \Psi' = \langle \mathcal{E}', \mathcal{D}', \mathcal{T}', \mathcal{U}' \rangle$ with $\mathcal{E}' = (E', \rightsquigarrow', \mapsto', l')$.

1. '$\Rightarrow$': Assume that $\sigma \in T_U(\Psi)$ and $\sigma' \in T_U(\Psi')$. We prove that $\sigma\,\sigma' \in T_U(\Psi)$ by systematically checking the conditions of being a timed event trace (see Definition 6.3).

   (a) $[\sigma\,\sigma'] \in T(\mathcal{E})$. Given that $[\sigma] \in T(\mathcal{E})$ and $[\sigma'] \in T(\mathcal{E}')$ this follows directly from Theorem 2.30.

   (b) $\forall\, i : \neg\mathcal{U}(e_i) \;\Rightarrow\; t_i \geqslant \mathsf{time}((\sigma\,\sigma')_i, e_i) \;\wedge\; \mathcal{U}(e_i) \;\Rightarrow\; t_i = \mathsf{time}((\sigma\,\sigma')_i, e_i)$. We consider the second conjunct; the proof of the first conjunct is conducted in a similar way and is omitted. We derive:

$$
\begin{array}{ll}
& \forall\, i : \mathcal{U}(e_i) \;\Rightarrow\; t_i = \mathsf{time}((\sigma\,\sigma')_i, e_i) \\
\Leftrightarrow & \{ \text{ domain split } \} \\
& \quad (\forall\, 0 < i \leqslant |\,\sigma\,| : \mathcal{U}(e_i) \;\Rightarrow\; t_i = \mathsf{time}((\sigma\,\sigma')_i, e_i)) \\
& \wedge\; (\forall\, |\,\sigma\,| < i \leqslant |\,\sigma\,\sigma'\,| : \mathcal{U}(e_i) \;\Rightarrow\; t_i = \mathsf{time}((\sigma\,\sigma')_i, e_i)) \\
\Leftrightarrow & \{ \text{ calculus } \} \\
& \quad (\forall\, 0 < i \leqslant |\,\sigma\,| : \mathcal{U}(e_i) \;\Rightarrow\; t_i = \mathsf{time}(\sigma_i, e_i)) \\
& \wedge\; (\forall\, 0 < j \leqslant |\,\sigma'\,| : \mathcal{U}(e_j) \;\Rightarrow\; t_j = \mathsf{time}(\sigma\,\sigma'_j, e_j)) \\
\Leftrightarrow & \{ \text{ calculus; } \mathcal{U}(e) = \mathcal{U}'(e) \text{ for } e \in E'\text{; Lemma 6.12 } \} \\
& \quad (\forall\, 0 < i \leqslant |\,\sigma\,| : \mathcal{U}(e_i) \;\Rightarrow\; t_i = \mathsf{time}(\sigma_i, e_i)) \\
& \wedge\; (\forall\, 0 < j \leqslant |\,\sigma'\,| : \mathcal{U}'(e_j) \;\Rightarrow\; t_j = \mathsf{time}'(\sigma'_j, e_j)) \\
\Leftarrow & \{ \text{ Definition 6.3 } \} \\
& \sigma \in T_U(\Psi) \;\wedge\; \sigma' \in T_U(\Psi') \quad .
\end{array}
$$

   (c) For the third constraint of being a timed event trace we have

$$
\begin{array}{ll}
& \forall\, i, e : e \in \mathsf{en}([(\sigma\,\sigma')_i]) \;\wedge\; \mathcal{U}(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}((\sigma\,\sigma')_i, e) \\
\Leftrightarrow & \{ \text{ domain split } \} \\
& (\forall\, 0 < i \leqslant |\,\sigma\,|, e : e \in \mathsf{en}([(\sigma\,\sigma')_i]) \;\wedge\; \mathcal{U}(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}((\sigma\,\sigma')_i, e)) \;\wedge \\
& (\forall\, |\,\sigma\,| < i \leqslant |\,\sigma\,\sigma'\,|, e : e \in \mathsf{en}([(\sigma\,\sigma')_i]) \;\wedge\; \mathcal{U}(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}((\sigma\,\sigma')_i, e)) \\
\Leftrightarrow & \{ \text{ calculus; } \mathcal{U}(e) = \mathcal{U}'(e) \text{ for } e \in E' \} \\
& (\forall\, 0 < i \leqslant |\,\sigma\,|, e : e \in \mathsf{en}([\sigma_i]) \;\wedge\; \mathcal{U}(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}(\sigma_i, e)) \;\wedge \\
& (\forall\, 0 < j \leqslant |\,\sigma'\,|, e : e \in \mathsf{en}([\sigma\,\sigma'_j]) \;\wedge\; \mathcal{U}'(e) \;\Rightarrow\; t_j \leqslant \mathsf{time}(\sigma\,\sigma'_j, e)) \\
\Leftrightarrow & \{ \text{ Lemma 6.12; Lemma 6.11 } \} \\
& (\forall\, 0 < i \leqslant |\,\sigma\,|, e : e \in \mathsf{en}([\sigma_i]) \;\wedge\; \mathcal{U}(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}(\sigma_i, e)) \;\wedge \\
& (\forall\, 0 < j \leqslant |\,\sigma'\,|, e : e \in \mathsf{en}'([\sigma'_j]) \;\wedge\; \mathcal{U}'(e) \;\Rightarrow\; t_j \leqslant \mathsf{time}'(\sigma'_j, e)) \\
\Leftarrow & \{ \text{ Definition 6.3 } \} \\
& \sigma \in T_U(\Psi) \;\wedge\; \sigma' \in T_U(\Psi') \quad .
\end{array}
$$

   (d) $\sigma\,\sigma'$ is time-consistent by assumption.

   This concludes the proof that $\sigma\,\sigma' \in T_U(\Psi)$.

   '$\Leftarrow$': the proof for this direction can be provided along the same lines as the proof for $\Rightarrow$ using Lemma 6.12 and Lemma 6.11.

2. Let $\sigma' \in T_U(\Psi')$. From 1. it follows that $\sigma\,\sigma' \in T_U(\Psi)$, so $L_U(\overline{\sigma\,\sigma'})$ exists. Evidently, we have $\overline{\sigma} \subseteq \overline{\sigma\,\sigma'}$ and $\prec^*_{\overline{\sigma}} \subseteq \prec^*_{\overline{\sigma\,\sigma'}}$. Since $\sigma\,\sigma' \in T_U(\Psi)$ and Lemma 6.6 it follows that no event in $\overline{\sigma}'$ precedes (under $\prec^*_{\overline{\sigma\,\sigma'}}$) an event in $\overline{\sigma}$. This proves that $L_U(\overline{\sigma})$ is a prefix of $L_U(\overline{\sigma\,\sigma'})$.

□

6.3 A timed process algebra including urgency

This section extends the simple timed process algebra PA_T with an urgency operator. Section 6.3.1 introduces the syntax of the resulting timed formalism PA_U. Section 6.3.2 defines the causality-based semantics of PA_U and Section 6.3.3 presents an event-based operational semantics of PA_U. Since we consider a time-consistent setting we use the transition model of Chapter 6 based on separate time- and action transitions for this purpose. The consistency between these two semantics is proven in Section 6.5.

6.3.1 Syntax

In order to have a means to express urgency PA_T is extended with an *urgency* operator, denoted $\mathcal{U}_U()$, for $U \subseteq \mathsf{Act}^\tau$. Let PA_T^+ denote the resulting formalism.

6.14. DEFINITION. (*Timed process algebra with urgency* PA_T^+)

$$B ::= \mathbf{0} \;\big|\; \surd \;\big|\; (t)\ a\,;B \;\big|\; B+B \;\big|\; B\,||_G\,B \;\big|\; B[H] \;\big|\; B \setminus G \;\big|\; B >\!\!> B \;\big|\; B\,[\!>B \;\big|\; \mathcal{U}_U(B).$$

□

$\mathcal{U}_U(B)$ behaves like B except that actions in U are *forced* to happen as soon as they are enabled. If U is a singleton set, $\{\,a\,\}$ say, we simply write $\mathcal{U}_a()$ instead of $\mathcal{U}_{\{a\}}()$. Notice that U may contain also internal action τ. $\mathcal{U}_U()$ is a generalization of the urgency operator $\hat{}$ introduced by Brinksma *et al.* [28], where $\hat{a}$ denotes action a that is forced to happen urgently. $\hat{}$ is restricted to be only applied to actions whose occurrence can be controlled completely internally. Here, urgency can involve several participants and is strongly influenced by the more general notion of urgency in proposals for timed extensions of LOTOS by Bolognesi *et al.* [18, 19] and similar work by Klusener, inspired by [18], in the setting of real-time ACP [86].

6.15. EXAMPLE. Consider $\mathcal{U}_c(a\,;((t_1)\,b\,;B_1 + (t_2)\,c\,;B_2))$.

After the occurrence of a it specifies a choice between $b\,;B_1$ and $c\,;B_2$. The first behaviour is enabled t_1 time units after a's occurrence, the second behaviour after t_2 time units. When b is performed before the second argument is enabled (i.e., $t_b \in t_a + [t_1, t_2]$) the entire behaviour subsequently behaves like B_1. Otherwise, precisely t_2 time units after the appearance of a it behaves like $c\,;B_2$, since c is urgent. □

Urgent interactions are forced to happen once all participants are ready for it. E.g., in

$$B = a\,;(3)\ c\,;\mathbf{0}\,||_c\,b\,;((2)\ d\,;\mathbf{0} + (5)\ c\,;\mathbf{0})$$

c can occur at any $t_c \geqslant max(t_a{+}3, t_b{+}5)$ provided d has not yet appeared. If c has not yet occurred, d can occur from $t_b{+}2$ on. In $\mathcal{U}_c(B)$ action c is forced to happen at $t_c = max(t_a{+}3, t_b{+}5)$ in case d has not yet appeared at that time. That is, d is prevented to occur at any time later than t_c, and can only occur in the interval $[t_b{+}2, t_c]$. At time t_c a nondeterministic choice between c and d occurs—urgency does not impose a priority in this case.

Once made urgent, actions cannot be used for synchronization any further. Without such a restriction, expressions like $B = \mathcal{U}_b((2)\ b)\,||_b\,\mathcal{U}_b((1)\ b)$ would be allowed. Conforming to the principle that an urgent action happens as soon as all participants are ready for it, $(b, 2)$ would be a trace of B. This would cause a delay of action b in the right component, contradicting its (local) urgency. The fact that we do not allow synchronizations on urgent events is captured by a syntactical constraint on behaviours which is formulated as follows. As a subsidiary notion we introduce a function that determines syntactically the set of urgent actions of a behaviour.

6.16. Definition. For $B \in \mathsf{PA}_T^+$, function $\mathsf{Urgent} : \mathsf{PA}_T^+ \longrightarrow \mathcal{P}(\mathsf{Act}^\tau)$ is defined as:

$$
\begin{array}{rcl}
\mathsf{Urgent}(B) & \triangleq & \varnothing \text{ for } B \in \{\, \mathbf{0}, \surd \,\} \\
\mathsf{Urgent}((t)\ a\,;\,B) & \triangleq & \mathsf{Urgent}(B) \\
\mathsf{Urgent}(B_1 \ \mathsf{op}\ B_2) & \triangleq & \mathsf{Urgent}(B_1) \cup \mathsf{Urgent}(B_2) \text{ for } \ \mathsf{op}\ \in \{\, +,\ ||_G\,, >>, [>\,\} \\
\mathsf{Urgent}(B \setminus G) & \triangleq & \begin{cases} (\mathsf{Urgent}(B) \setminus G) \cup \{\,\tau\,\} & \text{if } \mathsf{Urgent}(B) \cap G \neq \varnothing \\ \mathsf{Urgent}(B) & \text{otherwise} \end{cases} \\
\mathsf{Urgent}(B[H]) & \triangleq & \{\, H(a) \mid a \in \mathsf{Urgent}(B) \,\} \\
\mathsf{Urgent}(\mathcal{U}_U(B)) & \triangleq & \mathsf{Urgent}(B) \cup U.
\end{array}
$$

□

6.17. Definition. (*Temporal process algebra* PA_U)

PA_U is the largest subset of PA_T^+ such that any subexpression B' of $B \in \mathsf{PA}_U$ satisfies:

$$B' = B_1\,||_G\,B_2 \ \Rightarrow\ (G \,\cap\, \mathsf{Urgent}(B_1) = \varnothing \ \wedge\ G \,\cap\, \mathsf{Urgent}(B_2) = \varnothing) \quad .$$

□

6.3.2 Causality-based semantics

In this section we give a causality-based semantics to PA_U. We do so by defining a mapping $\mathcal{E}_U[\![\ ]\!] : \mathsf{PA}_U \longrightarrow \mathsf{EBES}_U$. Let $\mathcal{E}_U[\![\, B_i \,]\!] = \Psi_i = \langle \Gamma_i, \mathcal{U}_i \rangle$, for $i{=}1, 2$. Then:

6.18. Definition. (*Causality-based semantics of* PA_U)

Let $\mathcal{E}_U[\![\ ]\!] : \mathsf{PA}_U \longrightarrow \mathsf{EBES}_U$ be defined as follows:

$$
\begin{array}{rcl}
\mathcal{E}_U[\![\, \mathbf{0} \,]\!] & \triangleq & \langle \mathcal{E}_T[\![\, \mathbf{0} \,]\!], \varnothing \rangle \\
\mathcal{E}_U[\![\, \surd \,]\!] & \triangleq & \langle \mathcal{E}_T[\![\, \surd \,]\!], \{\, (e_\delta, \text{false}) \,\} \rangle
\end{array}
$$

$$\begin{array}{rcl}
\mathcal{E}_U[\![\,(t)\ a\,;\,B_1\,]\!] & \triangleq & \langle \mathcal{E}_T[\![\,(t)\ a\,;\,B_1\,]\!], \mathcal{U}_1 \cup \{\,(e_a, \text{false})\,\}\rangle \\
\mathcal{E}_U[\![\,B_1\ \mathsf{op}\ B_2\,]\!] & \triangleq & \langle \mathcal{E}_T[\![\,B_1\ \mathsf{op}\ B_2\,]\!], \mathcal{U}_1 \cup \mathcal{U}_2\rangle \text{ for } \mathsf{op}\ \in \{\,+, >\!\!>, [>\,\} \\
\mathcal{E}_U[\![\,\mathsf{op}\ B_1\,]\!] & \triangleq & \langle \mathcal{E}_T[\![\,\mathsf{op}\ B_1\,]\!], \mathcal{U}_1\rangle \text{ for } \mathsf{op}\ \in \{\,\backslash, [\,]\,\} \\
\mathcal{E}_U[\![\,\mathcal{U}_U(B_1)\,]\!] & \triangleq & \langle \mathcal{E}_T[\![\,B_1\,]\!], \mathcal{U}\rangle \text{ where } \mathcal{U}(e) = \mathcal{U}_1(e)\ \vee\ (l_1(e) \in U) \\
\mathcal{E}_U[\![\,B_1 \,||_G\, B_2\,]\!] & \triangleq & \langle \mathcal{E}_T[\![\,B_1 \,||_G\, B_2\,]\!], \mathcal{U}\rangle \text{ where} \\
\mathcal{U}((e_1, e_2)) & = & \mathcal{U}_1(e_1)\ \vee\ \mathcal{U}_2(e_2) \text{ with } \mathcal{U}_i(*) = \text{false, for } i{=}1,2.
\end{array}$$

□

It is easy to check that due to the syntactical constraints of Definition 6.17 we have for $||_G$ that $(e_1 \neq * \ \wedge\ e_2 \neq *) \ \Rightarrow\ \neg\mathcal{U}((e_1, e_2))$, since in this case e_1 and e_2 synchronize. It is also not difficult to check that for all $B \in \mathsf{PA}_U$ we have that $\mathcal{E}_U[\![\,B\,]\!]$ is an urgent event structure.

6.19. Example. In Figure 6.4 the urgent event structures corresponding to the following expressions are depicted:

(a) $\mathcal{U}_b((2)\ a\,;\,(4)\ b\,;\,\mathbf{0} \,||_b\, (7)\ b\,;\,\mathbf{0})$,

(b) $((2)\ a\,;\,(7)\ x\,;\,\mathbf{0} \,|||\, \mathcal{U}_y((4)\ a\,;\,(11)\ y\,;\,\mathbf{0})) \,||_a\, ((5)\ a\,;\,(2)\ b\,;\,\mathbf{0})$, and

(c) $\mathcal{U}_{y_1}(a_1\,;\,((t_1)\ x\,;\,\mathbf{0} + (d_1)\ y_1\,;\,\mathbf{0})) \,||_x\, \mathcal{U}_{y_2}(a_2\,;\,((t_2)\ x\,;\,\mathbf{0} + (d_2)\ y_2\,;\,\mathbf{0}))$.

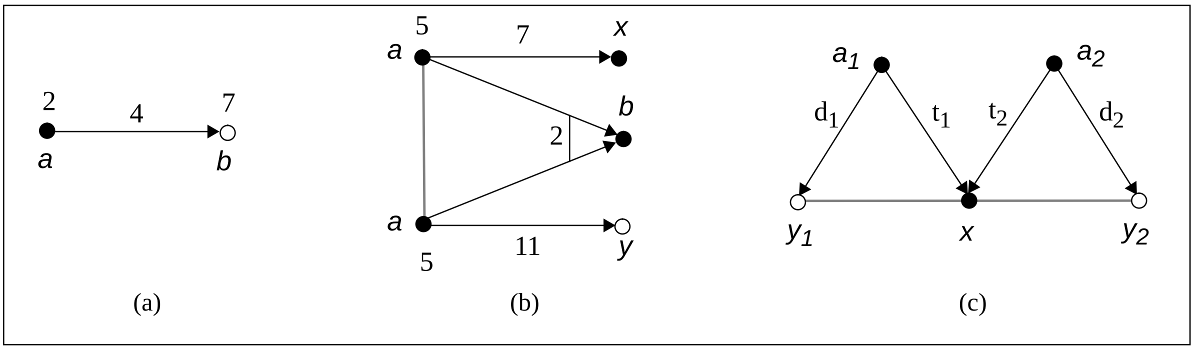


Figure 6.4: Examples of semantics of urgent behaviours.

For urgent event structure (c) we have that if (x, t_x) belongs to a timed event trace then $t_{a_1}{+}t_1 \leqslant t_x \leqslant t_{a_1}{+}d_1 \ \wedge\ t_{a_2}{+}t_2 \leqslant t_x \leqslant t_{a_2}{+}d_2$. □

6.3.3 Event-based operational semantics for PA_U

This section extends the timed event transition system of Section 5.4 with urgency. The relations $\rightsquigarrow$ and $\longrightarrow$ are defined as the smallest relations closed under all inference rules of Section 5.4 and the rules for $\mathcal{U}_U(B)$ defined below.

As a subsidiary notion, let $d_{min}(a, B)$ determine for initial action a in B the minimal time at which a can appear. The interpretation of $d_{min}(a, B) = \infty$ is that B is not able to perform an a action initially.

6.20. DEFINITION. Function $d_{min} : \mathsf{Act}^{\tau,\delta} \times \mathsf{PA}_U \longrightarrow \mathsf{Time}^\infty$ is defined as:

$$
\begin{array}{rcl}
d_{min}(a, \mathbf{0}) & \triangleq & \infty \\
d_{min}(a, \surd) & \triangleq & \begin{cases} \infty & \text{if } a \neq \delta \\ 0 & \text{if } a = \delta \end{cases} \\
d_{min}(a, (t)\, b\, ;\, B) & \triangleq & \begin{cases} \infty & \text{if } a \neq b \\ t & \text{if } a = b \end{cases} \\
d_{min}(a, B_1 + B_2) & \triangleq & min(d_{min}(a, B_1), d_{min}(a, B_2)) \\
d_{min}(a, B_1 >> B_2) & \triangleq & \begin{cases} \infty & \text{if } a = \delta \\ d_{min}(a, B_1) & \text{if } a \notin \{\, \tau, \delta \,\} \\ min(d_{min}(a, B_1), d_{min}(\delta, B_1)) & \text{if } a = \tau \end{cases} \\
d_{min}(a, B_1 \,[>\, B_2) & \triangleq & min(d_{min}(a, B_1), d_{min}(a, B_2)) \\
d_{min}(a, B_1 \,||_G\, B_2) & \triangleq & \begin{cases} min(d_{min}(a, B_1), d_{min}(a, B_2)) & \text{if } a \notin G^\delta \\ max(d_{min}(a, B_1), d_{min}(a, B_2)) & \text{if } a \in G^\delta \end{cases} \\
d_{min}(a, B \setminus G) & \triangleq & \begin{cases} Min\{\, d_{min}(b, B) \mid b \in G^\tau \,\} & \text{if } a = \tau \\ \infty & \text{if } a \in G \\ d_{min}(a, B) & \text{if } a \notin G^\tau \end{cases} \\
d_{min}(a, B[H]) & \triangleq & Min\{\, d_{min}(b, B) \mid a = H(b) \,\} \\
d_{min}(a, \mathcal{U}_U(B)) & \triangleq & d_{min}(a, B).
\end{array}
$$

□

Here it is assumed that min, max and their generalizations over sets of events are defined on Time^∞ in the obvious way. E.g., $min(t, \infty) \triangleq t$ and $max(t, \infty) \triangleq \infty$.

Urgency
If B permits time to pass with some amount, then $\mathcal{U}_U(B)$ is able to do the same provided that there is no urgent action in U that can be performed by B at any time earlier. Thus, the effect of the urgency operator is to prevent the passage of time as an alternative to the occurrence of an action in the urgency set U. If B can perform (e, a) and evolve into B' then so can $\mathcal{U}_U(B)$, evolving into $\mathcal{U}_U(B')$.

$$
\frac{\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle}{\langle \mathcal{U}_U(B), t\rangle \rightsquigarrow \langle \mathcal{U}_U(B'), t'\rangle} \quad (\forall\, a \in U : t' - t \leqslant d_{min}(a, B))
$$

$$
\frac{\langle B, t\rangle \xrightarrow{(\xi, a)} \langle B', t\rangle}{\langle \mathcal{U}_U(B), t\rangle \xrightarrow{(\xi, a)} \langle \mathcal{U}_U(B'), t\rangle}
$$

For convenience we have listed all rules for time transitions in Table 6.1 and all rules for action transitions in Table 6.2 .

6.21. EXAMPLE. Consider $B = \mathcal{U}_b(B')$ with $B' = (2)\ a\, ;\, (1)\ b\, ;\, \mathbf{0} \,||_b\, (0)\ b\, ;\, \mathbf{0}$. (For simplicity we omit the occurrence identifiers.) It follows that $d_{min}(a, B) = 2$ and $d_{min}(b, B) =$

$$\frac{}{\langle \mathbf{0}, t\rangle \rightsquigarrow \langle \mathbf{0}, t'\rangle}\ (t' \geqslant t) \qquad \frac{}{\langle (t')\, a_\xi\, ;\, B, t\rangle \rightsquigarrow \langle (t' \ominus (t''{-}t))\, a_\xi\, ;\, B, t''\rangle}\ (t'' \geqslant t)$$

$$\frac{}{\langle \surd_\xi, t\rangle \rightsquigarrow \langle \surd_\xi, t'\rangle}\ (t' \geqslant t) \qquad \frac{\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \ \wedge\ \langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle}{\langle B_1 + B_2, t\rangle \rightsquigarrow \langle B_1' + B_2', t'\rangle}$$

$$\frac{\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle}{\langle B_1 >> B_2, t\rangle \rightsquigarrow \langle B_1' >> B_2, t'\rangle} \qquad \frac{\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \ \wedge\ \langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle}{\langle B_1\, [> B_2, t\rangle \rightsquigarrow \langle B_1'\, [> B_2', t'\rangle}$$

$$\frac{\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle}{\langle B \setminus G, t\rangle \rightsquigarrow \langle B' \setminus G, t'\rangle} \qquad \frac{\langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \ \wedge\ \langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle}{\langle B_1 \,||_G\, B_2, t\rangle \rightsquigarrow \langle B_1' \,||_G\, B_2', t'\rangle}$$

$$\frac{\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle}{\langle B[H], t\rangle \rightsquigarrow \langle B'[H], t'\rangle} \qquad \frac{\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle}{\langle \mathcal{U}_U(B), t\rangle \rightsquigarrow \langle \mathcal{U}_U(B'), t'\rangle}\ (C)$$

Table 6.1: Time transition rules for PA_U where C equals $\forall\, a \in U : t'{-}t \leqslant d_{min}(a, B)$.

$max(\infty, 0) = \infty$. Assume that a happens at time 7, say. Then we infer for the component behaviours of B':

$$\langle (2)\ a\, ;\, (1)\ b\, ;\, \mathbf{0}, 0\rangle \rightsquigarrow \langle (0)\ a\, ;\, (1)\ b\, ;\, \mathbf{0}, 7\rangle \text{ and } \langle (0)\ b\, ;\, \mathbf{0}, 0\rangle \rightsquigarrow \langle (0)\ b\, ;\, \mathbf{0}, 7\rangle \quad .$$

Using the inference rules for $\rightsquigarrow$ for parallel composition and urgency we obtain

$$\langle B, 0\rangle \rightsquigarrow \langle \mathcal{U}_b((0)\ a\, ;\, (1)\ b\, ;\, \mathbf{0} \,||_b\, (0)\ b\, ;\, \mathbf{0}), 7\rangle \quad .$$

By the inference rules for $\longrightarrow$ for parallel composition ($a \notin G^\delta$) and urgency we get

$$\langle \mathcal{U}_b((0)\ a\, ;\, (1)\ b\, ;\, \mathbf{0} \,||_b\, (0)\ b\, ;\, \mathbf{0}), 7\rangle \xrightarrow{a} \langle \mathcal{U}_b((1)\ b\, ;\, \mathbf{0} \,||_b\, (0)\ b\, ;\, \mathbf{0}), 7\rangle \quad .$$

Let us denote $\mathcal{U}_b((1)\ b\, ;\, \mathbf{0} \,||_b\, (0)\ b\, ;\, \mathbf{0})$ by B''. It follows by Definition 6.20 that $d_{min}(b, B'') = 1$. Due to the inference rule for $\rightsquigarrow$, behaviour B'' allows the passage of time for at most 1 time unit only. By this mechanism it is enforced that b happens precisely at time 8. □

6.22. Example. Let $B = \mathcal{U}_U((2)\ a\, ;\, (1)\ b\, ;\, \mathbf{0} \,||_b\, (0)\ b\, ;\, \mathbf{0}\, [> (7)\ c\, ;\, \mathbf{0})$ with $U = \{a, b\}$. (Again, event identifiers are omitted.) Using Definition 6.20 we have $d_{min}(a, B) = 2$, and $d_{min}(b, B) = \infty$. We then have the following derivation:

$$\begin{array}{ll} & \langle \mathcal{U}_U((2)\ a\, ;\, (1)\ b\, ;\, \mathbf{0} \,||_b\, (0)\ b\, ;\, \mathbf{0}\, [> (7)\ c\, ;\, \mathbf{0}), 0\rangle \\ \rightsquigarrow & \{\ \text{(timed action-prefix), (choice), (parallel composition), (urgency)}\ \} \\ & \langle \mathcal{U}_U((0)\ a\, ;\, (1)\ b\, ;\, \mathbf{0} \,||_b\, (0)\ b\, ;\, \mathbf{0}\, [> (5)\ c\, ;\, \mathbf{0}), 2\rangle \\ \xrightarrow{a} & \{\ \text{(timed action-prefix), (parallel composition), (urgency)}\ \} \end{array}$$

$$\frac{}{\langle \surd_\xi, t\rangle \xrightarrow{(\xi,\delta)} \langle \mathbf{0}, t\rangle} \qquad \frac{}{\langle (0)\, a_\xi\,;\, B, t\rangle \xrightarrow{(\xi,a)} \langle B, t\rangle}$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle}{\langle B_1 + B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle} \qquad \frac{\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}{\langle B_1 + B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle}{\langle B_1 >> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1' >> B_2, t\rangle}\ (a \neq \delta) \qquad \frac{\langle B_1, t\rangle \xrightarrow{(\xi,\delta)} \langle B_1', t\rangle}{\langle B_1 >> B_2, t\rangle \xrightarrow{(\xi,\tau)} \langle B_2, t\rangle}$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,\delta)} \langle B_1', t\rangle}{\langle B_1\, [> B_2, t\rangle \xrightarrow{(\xi,\tau)} \langle B_1', t\rangle} \qquad \frac{\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}{\langle B_1\, [> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle}{\langle B_1\, [> B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_1'\, [> B_2, t\rangle} \quad (a \neq \delta)$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle}{\langle B_1\, ||_G\, B_2, t\rangle \xrightarrow{((\xi,*),a)} \langle B_1'\, ||_G\, B_2, t\rangle} \quad (a \notin G^\delta)$$

$$\frac{\langle B_2, t\rangle \xrightarrow{(\xi,a)} \langle B_2', t\rangle}{\langle B_1\, ||_G\, B_2, t\rangle \xrightarrow{((*,\xi),a)} \langle B_1\, ||_G\, B_2', t\rangle} \quad (a \notin G^\delta)$$

$$\frac{\langle B_1, t\rangle \xrightarrow{(\xi,a)} \langle B_1', t\rangle \ \wedge\ \langle B_2, t\rangle \xrightarrow{(\psi,a)} \langle B_2', t\rangle}{\langle B_1\, ||_G\, B_2, t\rangle \xrightarrow{((\xi,\psi),a)} \langle B_1'\, ||_G\, B_2', t\rangle} \quad (a \in G^\delta)$$

$$\frac{\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle}{\langle B \setminus G, t\rangle \xrightarrow{(\xi,a)} \langle B' \setminus G, t\rangle}\ (a \notin G) \qquad \frac{\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle}{\langle B \setminus G, t\rangle \xrightarrow{(\xi,\tau)} \langle B' \setminus G, t\rangle}\ (a \in G)$$

$$\frac{\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle}{\langle B[H], t\rangle \xrightarrow{(\xi,H(a))} \langle B'[H], t\rangle} \qquad \frac{\langle B, t\rangle \xrightarrow{(\xi,a)} \langle B', t\rangle}{\langle \mathcal{U}_U(B), t\rangle \xrightarrow{(\xi,a)} \langle \mathcal{U}_U(B'), t\rangle}$$

Table 6.2: Action transition rules for PA_U.

$$
\begin{array}{ll}
& \langle \mathcal{U}_U((1)\ b\,;\,\mathbf{0}\,||_b\,(0)\ b\,;\,\mathbf{0}\,[>\ (5)\ c\,;\,\mathbf{0}), 2\rangle \\
\rightsquigarrow & \{\ \text{(timed action-prefix), (choice), (parallel composition), (urgency)}\ \} \\
& \langle \mathcal{U}_U((0)\ b\,;\,\mathbf{0}\,||_b\,(0)\ b\,;\,\mathbf{0}\,[>\ (4)\ c\,;\,\mathbf{0}), 3\rangle \\
\xrightarrow{b} & \{\ \text{(timed action-prefix), (choice), (synchronization), (urgency)}\ \} \\
& \langle \mathcal{U}_U(\mathbf{0}\,||_b\,\mathbf{0}), 3\rangle \quad .
\end{array}
$$

Since this is the only allowed derivation (apart from intermediate $\rightsquigarrow$ transitions) it follows that c will never happen. B corresponds to the urgent event structure in Figure 6.2(b). □

We conclude this section by considering the properties time determinism, action persistency, and time additivity for PA_U. It turns out that the introduction of urgency does not disturb these properties.

6.23. THEOREM. *Action persistency, time determinism, and time additivity*

For all $B, B', B'' \in \mathsf{PA}_U$, $t, t', t'' \in \mathsf{Time}$ we have

1. $\langle B, t\rangle \rightsquigarrow \langle B, t\rangle$
2. $\big(\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle \ \wedge\ \langle B, t\rangle \rightsquigarrow \langle B'', t'\rangle\big) \ \Rightarrow\ B' = B''$
3. $\big(\langle B, t\rangle \xrightarrow{(e,a)} \ \wedge\ \langle B, t\rangle \rightsquigarrow \langle B', t'\rangle\big) \ \Rightarrow\ \langle B', t'\rangle \xrightarrow{(e,a)}$
4. $\langle B, t\rangle \rightsquigarrow \langle B', t{+}(t'{+}t'')\rangle \iff \big(\exists\, B'' : \langle B, t\rangle \rightsquigarrow \langle B'', t{+}t'\rangle \rightsquigarrow \langle B', t{+}(t'{+}t'')\rangle\big)$.

PROOF. All properties can be proven by induction on the structure of B. We only have to consider the urgency construct; for the other constructs the inference rules are unchanged and the proof is provided in Chapter 5. For the sake of brevity we only provide the proof of action persistency (3.). The proofs for the other properties are rather similar. Let $B = \mathcal{U}_U(B_1)$ and assume the theorem holds for B_1.

$$
\begin{array}{ll}
& \langle \mathcal{U}_U(B_1), t\rangle \xrightarrow{(e,a)} \ \wedge\ \langle \mathcal{U}_U(B_1), t\rangle \rightsquigarrow \langle \mathcal{U}_U(B_1'), t'\rangle \\
\Leftrightarrow & \{\ \text{SOS-rules for urgency}\ \} \\
& \langle B_1, t\rangle \xrightarrow{(e,a)} \ \wedge\ \langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \ \wedge\ (\forall\, b \in U : t'{-}t \leqslant d_{min}(b, B_1)) \\
\Rightarrow & \{\ \text{induction hypothesis}\ \} \\
& \langle B_1', t'\rangle \xrightarrow{(e,a)} \\
\Leftrightarrow & \{\ \text{SOS-rule } (\longrightarrow) \text{ for urgency}\ \} \\
& \langle \mathcal{U}_U(B_1'), t'\rangle \xrightarrow{(e,a)} \quad .
\end{array}
$$

□

6.4 Is urgency captured faithfully?

This section proves the correctness of the d_{min} function, in the sense that urgency is captured in a way corresponding to our intuition about what urgency should be. Some of the correctness results are essential to provide a timed event trace semantics of PA_U which is used in Section 6.5.3 for proving the consistency between the denotational and the event-based operational semantics. We prove the following properties:

- the time determined by $d_{min}(a, B)$ corresponds to the earliest moment at which initial action a can be performed by B
- urgent actions in B can only be performed as soon as they are possible, i.e., once enabled, their execution cannot be postponed
- $d_{min}(a, B) = \infty$ corresponds to the fact that B is not able to perform action a initially
- actions can only be performed by B provided there is no urgent action in B that could occur earlier, and
- if B advances t time units then d'_{min} of the resulting behaviour equals $d_{min} \ominus t$.

The proofs of these properties are all by induction on the structure of expressions. As an illustration we provide only proofs for three properties; the proofs of the other properties are conducted in a similar way and are left to the diligent reader.

The first theorem confirms that the time determined by the function $d_{min}(a, B)$ indeed corresponds to the minimal time at which initial action a can be performed by B.

6.24. THEOREM. $\forall\, B_0 \in \mathsf{PA}_U, t_0 \in \mathsf{Time}, a \in \mathsf{Act}^{\tau,\delta} : \langle B_0, t_0\rangle \xrightarrow{(e,a,t)}_* \Rightarrow t \geqslant t_0 + d_{min}(a, B_0)$.

PROOF. By induction on the structure of B_0 with base cases $\mathbf{0}$, $\surd$, and action-prefix.
Base: For $B_0 = \mathbf{0}$ the theorem trivially holds as the premise of the theorem does not hold. For $B_0 = \surd$ it is easy to check that the theorem holds as δ can be performed at any time and $d_{min}(\delta, \surd) = 0$. For $B_0 = (t')\ b\,;\, B_1$ we derive:

$$
\begin{array}{ll}
& \langle (t')\ b\,;\, B_1, t_0\rangle \xrightarrow{(e,a,t)}_* \\
\Leftrightarrow & \{\ \text{Definitions 6.20 and 5.24}\ \} \\
& \quad (\exists\, B' : \langle (t')\ b\,;\, B_1, t_0\rangle \rightsquigarrow \langle B', t\rangle \xrightarrow{(e,a)}) \\
& \wedge\ (b = a\ \Rightarrow\ d_{min}(a, B_0) = t'\ \wedge\ b \neq a\ \Rightarrow\ d_{min}(a, B_0) = \infty) \\
\Rightarrow & \{\ \text{SOS-rules for} \rightsquigarrow \text{and} \longrightarrow\ \} \\
& \quad (\exists\, B' : t \geqslant t_0 + t'\ \wedge\ B' = (0)\ b\,;\, B_1\ \wedge\ a = b) \\
& \wedge\ (b = a\ \Rightarrow\ d_{min}(a, B_0) = t'\ \wedge\ b \neq a\ \Rightarrow\ d_{min}(a, B_0) = \infty) \\
\Rightarrow & \{\ \text{calculus}\ \} \\
& t \geqslant t_0 + d_{min}(a, B_0) \quad .
\end{array}
$$

Induction Step: Assume the lemma holds for B_1 and B_2. We only consider parallel composition and urgency. The proofs for the other constructs are similar and are omitted.

1. For $B_0 = B_1 \,||_G\, B_2$ we derive:

$$
\begin{array}{ll}
& \langle B_1 \,||_G\, B_2, t_0\rangle \xrightarrow{(e,a,t)}_* \\
\Leftrightarrow & \{\ \text{Definition 5.24}\ \} \\
& \exists\, B' : \langle B_1 \,||_G\, B_2, t_0\rangle \rightsquigarrow \langle B', t\rangle \xrightarrow{(e,a)} \\
\Leftrightarrow & \{\ \text{SOS-rules for} \rightsquigarrow\ \} \\
& \exists\, B_1', B_2' : \langle B_1 \,||_G\, B_2, t_0\rangle \rightsquigarrow \langle B_1' \,||_G\, B_2', t\rangle \xrightarrow{(e,a)}
\end{array}
$$

$$\begin{array}{ll}
\Leftrightarrow & \{ \text{ SOS-rules for } \rightsquigarrow \} \\
& \exists\, B_1', B_2' : \langle B_1, t_0 \rangle \rightsquigarrow \langle B_1', t \rangle \;\wedge\; \langle B_2, t_0 \rangle \rightsquigarrow \langle B_2', t \rangle \;\wedge\; \langle B_1' \,||_G\, B_2', t \rangle \xrightarrow{(e,a)} \quad .
\end{array}$$

At this point in the derivation we distinguish between two cases: $a \in G^\delta$ and $a \notin G^\delta$. For completeness we consider both cases.

(a) For $a \in G^\delta$ we deduce starting from the result of the derivation above:

$$\begin{array}{ll}
\Leftrightarrow & \{ \text{ SOS-rules for } \longrightarrow \;;\; a \in G^\delta \} \\
& \exists\, B_1', B_2' : \langle B_1' \,||_G\, B_2', t \rangle \xrightarrow{((e,e'),a)} \;\wedge \\
& \langle B_1, t_0 \rangle \rightsquigarrow \langle B_1', t \rangle \xrightarrow{(e,a)} \;\wedge\; \langle B_2, t_0 \rangle \rightsquigarrow \langle B_2', t \rangle \xrightarrow{(e',a)} \\
\Rightarrow & \{ \text{ calculus ; Definition 5.24 } \} \\
& \langle B_1, t_0 \rangle \xrightarrow{(e,a,t)}_* \;\wedge\; \langle B_2, t_0 \rangle \xrightarrow{(e,a,t)}_* \\
\Rightarrow & \{ \text{ induction hypothesis } \} \\
& t \geqslant t_0 + d_{min}(a, B_1) \;\wedge\; t \geqslant t_0 + d_{min}(a, B_2) \\
\Leftrightarrow & \{ \text{ calculus } \} \\
& t \geqslant t_0 + max(d_{min}(a, B_1), d_{min}(a, B_2)) \\
\Leftrightarrow & \{ \text{ Definition 6.20 } (a \in G^\delta) \} \\
& t \geqslant t_0 + d_{min}(a, B_1 \,||_G\, B_2) \quad .
\end{array}$$

(b) For $a \notin G^\delta$ we infer starting from the result of the derivation above:

$$\begin{array}{ll}
\Leftrightarrow & \{ \text{ SOS-rules for } \longrightarrow \;;\; a \notin G^\delta \} \\
& \quad \exists\, B_1', B_2' : \langle B_1, t_0 \rangle \rightsquigarrow \langle B_1', t \rangle \;\wedge\; \langle B_2, t_0 \rangle \rightsquigarrow \langle B_2', t \rangle \;\wedge \\
& \quad (\langle B_1', t \rangle \xrightarrow{(e,a)} \langle B_1'', t \rangle \;\wedge\; \langle B_1' \,||_G\, B_2', t \rangle \xrightarrow{((e,*),a)}) \\
& \vee\; (\langle B_2', t \rangle \xrightarrow{(e,a)} \langle B_2'', t \rangle \;\wedge\; \langle B_1' \,||_G\, B_2', t \rangle \xrightarrow{((*,e),a)}) \\
\Rightarrow & \{ \text{ calculus ; Definition 5.24 } \} \\
& \langle B_1, t_0 \rangle \xrightarrow{(e,a,t)}_* \;\vee\; \langle B_2, t_0 \rangle \xrightarrow{(e,a,t)}_* \\
\Rightarrow & \{ \text{ induction hypothesis } \} \\
& t \geqslant t_0 + d_{min}(a, B_1) \;\vee\; t \geqslant t_0 + d_{min}(a, B_2) \\
\Leftrightarrow & \{ \text{ calculus } \} \\
& t \geqslant t_0 + min(d_{min}(a, B_1), d_{min}(a, B_2)) \\
\Leftrightarrow & \{ \text{ Definition 6.20 } (a \notin G^\delta) \} \\
& t \geqslant t_0 + d_{min}(a, B_1 \,||_G\, B_2) \quad .
\end{array}$$

2. For $B_0 = \mathcal{U}_U(B_1)$ we derive:

$$\begin{array}{ll}
& \langle \mathcal{U}_U(B_1), t_0 \rangle \xrightarrow{(e,a,t)}_* \\
\Leftrightarrow & \{ \text{ Definition 5.24 } \} \\
& \exists\, B' : \langle \mathcal{U}_U(B_1), t_0 \rangle \rightsquigarrow \langle B', t \rangle \xrightarrow{(e,a)} \\
\Rightarrow & \{ \text{ SOS-rules for } \rightsquigarrow \text{ and } \longrightarrow;\; B' = \mathcal{U}_U(B'') \} \\
& \exists\, e, B'' : \langle B_1, t_0 \rangle \rightsquigarrow \langle B'', t \rangle \;\wedge\; (\forall\, b \in U : t - t_0 \leqslant d_{min}(b, B_1)) \;\wedge\; \langle \mathcal{U}_U(B''), t \rangle \xrightarrow{(e,a)} \\
\Rightarrow & \{ \text{ SOS-rule for } \longrightarrow \}
\end{array}$$

$$
\begin{aligned}
& \exists\, e, B'' : \langle B_1, t_0 \rangle \rightsquigarrow \langle B'', t \rangle \xrightarrow{(e,a)} \ \wedge\ (\forall\, b \in U : t - t_0 \leqslant d_{min}(b, B_1)) \\
\Leftrightarrow\ & \{ \text{ Definition 5.24 } \} \\
& \langle B_1, t_0 \rangle \xrightarrow{(e,a,t)}_* \ \wedge\ (\forall\, b \in U : t - t_0 \leqslant d_{min}(b, B_1)) \\
\Rightarrow\ & \{ \text{ induction hypothesis } \} \\
& t \geqslant t_0 + d_{min}(a, B_1) \ \wedge\ (\forall\, b \in U : t \leqslant t_0 + d_{min}(b, B_1)) \\
\Leftrightarrow\ & \{ \text{ Definition 6.20 } \} \\
& t \geqslant t_0 + d_{min}(a, \mathcal{U}_U(B_1)) \ \wedge\ (\forall\, b \in U : t \leqslant t_0 + d_{min}(b, \mathcal{U}_U(B_1))) \quad .
\end{aligned}
$$

□

It is interesting to note that for $a \in U$ we obtain from the result just above that $t = t_0 + d_{min}(a, \mathcal{U}_U(B_1))$. This confirms that actions in U can only be performed as soon as they are possible and forms the basis for the following theorem. It says that any urgent action in behaviour B can only be performed as soon as possible:

6.25. THEOREM. $\forall\, B \in \mathsf{PA}_U, t' \in \mathsf{Time}, a \in \mathsf{Urgent}(B) : \langle B, t' \rangle \xrightarrow{(e,a,t)}_* \Rightarrow t = t' + d_{min}(a, B)$.

PROOF. By induction on the structure of B. Straightforward and omitted. □

As a next result we prove that $d_{min}(a, B) = \infty$ indeed corresponds to the fact that B is not able to initially perform action a.

6.26. THEOREM. $\forall\, B \in \mathsf{PA}_U, a \in \mathsf{Act}^{\tau,\delta} : d_{min}(a, B) = \infty \Longleftrightarrow \left(\forall\, t, t' : \langle B, t \rangle \xrightarrow{(e,a,t')}\!\!\!\!\!\!\not\ \ _*\right)$.

PROOF. By induction on the structure of B with base cases $\mathbf{0}$, $\surd$, and action prefix.
Base : For $B = \mathbf{0}$, $d_{min}(a, B) = \infty$ for all a. The theorem trivially holds as $\mathbf{0}$ is not able to perform any action. For $B = \surd$, $d_{min}(a, B) = \infty$ for $a \neq \delta$ and $d_{min}(\delta, B) = 0$. Since $\surd$ is only able to perform δ, this proves the case. For $B = (t)\ b\,;\, B_1$ we have that $d_{min}(a, B) = \infty$ if $a \neq b$. As B is able to initially only perform action b, the theorem evidently holds for this case.
Induction Step : Assume the theorem holds for B_1 and B_2. We only consider the proof for choice and parallel composition. The proofs for the other cases are quite similar and therefore omitted.

1. For $B = B_1 + B_2$ we derive:

$$
\begin{aligned}
& d_{min}(a, B_1 + B_2) = \infty \\
\Leftrightarrow\ & \{ \text{ Definition 6.20 } \} \\
& min(d_{min}(a, B_1), d_{min}(a, B_2)) = \infty \\
\Leftrightarrow\ & \{ \text{ calculus } \} \\
& d_{min}(a, B_1) = \infty \ \wedge\ d_{min}(a, B_2) = \infty \\
\Leftrightarrow\ & \{ \text{ induction hypothesis } \} \\
& (\forall\, t, t' : \langle B_1, t \rangle \xrightarrow{(e,a,t')}\!\!\!\!\!\!\not\ \ _*) \ \wedge\ (\forall\, t, t' : \langle B_2, t \rangle \xrightarrow{(e,a,t')}\!\!\!\!\!\!\not\ \ _*) \\
\Leftrightarrow\ & \{ \text{ SOS-rules for } \rightsquigarrow \text{ and } \longrightarrow \} \\
& \forall\, t, t' : \langle B_1 + B_2, t \rangle \xrightarrow{(e,a,t')}\!\!\!\!\!\!\not\ \ _* \quad .
\end{aligned}
$$

2. For $B_1 \,||_G\, B_2$ we distinguish between $a \in G^\delta$ and $a \notin G^\delta$. The proof for $a \notin G^\delta$ is quite similar to the proof for $+$, so we concentrate on $a \in G^\delta$.

$$
\begin{aligned}
& d_{min}(a, B_1 \,||_G\, B_2) = \infty \\
\Leftrightarrow \quad & \{ \text{ Definition 6.20 } (a \in G^\delta) \ \} \\
& max(d_{min}(a, B_1), d_{min}(a, B_2)) = \infty \\
\Leftrightarrow \quad & \{ \text{ calculus } \} \\
& d_{min}(a, B_1) = \infty \ \vee \ d_{min}(a, B_2) = \infty \\
\Leftrightarrow \quad & \{ \text{ induction hypothesis } \} \\
& (\forall t, t' : \langle B_1, t\rangle \xrightarrow{(e,a,t')}\!\!\!\!\!\!\not\ \ _*) \ \vee \ (\forall t, t' : \langle B_2, t\rangle \xrightarrow{(e',a,t')}\!\!\!\!\!\!\not\ \ _*) \\
\Leftrightarrow \quad & \{ \text{ SOS-rules for } \rightsquigarrow \text{ and } \longrightarrow \ (a \in G^\delta) \ \} \\
& \forall t, t' : \langle B_1 \,||_G\, B_2, t\rangle \xrightarrow{((e,e'),a,t')}\!\!\!\!\!\!\not\ \ _* \quad .
\end{aligned}
$$

□

As a next property we have that action a can only be performed by behaviour B provided there is no urgent event in B with a smaller d_{min}.

6.27. THEOREM. $\forall B \in \mathsf{PA}_U : \langle B, t\rangle \xrightarrow{(e,a,t')}_* \Rightarrow t' \leqslant t + Min\{\, d_{min}(b, B) \mid b \in \mathsf{Urgent}(B) \,\}$.

PROOF. Straightforward by induction on B. □

We conclude by proving that the intertwining of $\rightsquigarrow$ and d_{min} is as one would expect. More precisely, if B at time t can perform a then B' at time t', obtained from B by the passage of $t'-t$ time units, can perform a at $d_{min}(a, B) \ominus (t'-t)$. Let $\infty - x = \infty$.

6.28. THEOREM. $\forall B, B' \in \mathsf{PA}_U, t, t' \in \mathsf{Time}$:

$$
\Big(\langle B, t\rangle \xrightarrow{(e,a,t_a)}_* \ \wedge \ \langle B, t\rangle \rightsquigarrow \langle B', t'\rangle\Big) \ \Rightarrow \ d_{min}(a, B') = d_{min}(a, B) \ominus (t'-t) \quad .
$$

PROOF. By induction on the structure of B with base cases $\mathbf{0}$, $\surd$, and action-prefix.
Base: For $B = \mathbf{0}$ the theorem trivially holds as $\langle \mathbf{0}, t\rangle$ cannot perform any action. $B = \surd$ can only perform δ and under $\rightsquigarrow$ evolve into $\surd$. Since $d_{min}(\delta, \surd) = 0$ the theorem holds for this case. Let $B = (t'')\ a\,;B_1$. We have $d_{min}(a, B) = t''$ and $d_{min}(b, B) = \infty$, for $b \neq a$. Assume $\langle B, t\rangle \rightsquigarrow \langle B', t'\rangle$. By the inference rules for $\rightsquigarrow$ we have $B' = (t'' \ominus (t'-t))\ a\,; B_1$, and it follows $d_{min}(a, B') = t'' \ominus (t'-t) = d_{min}(a, B) \ominus (t'-t)$ and $d_{min}(b, B') = \infty = d_{min}(b, B) \ominus (t'-t)$.
Induction Step: Assume the theorem holds for B_1 and B_2. We only provide the proof for synchronization, the proofs for the other cases are similar and omitted. Let $B = B_1 \,||_G\, B_2$ and assume $a \in G^\delta$. We then derive

$$
\begin{aligned}
& \langle B_1 \,||_G\, B_2, t\rangle \xrightarrow{(e,a,t_a)}_* \ \wedge \ \langle B_1 \,||_G\, B_2, t\rangle \rightsquigarrow \langle B', t'\rangle \\
\Leftrightarrow \quad & \{ \text{ SOS-rules for } \longrightarrow \text{ and } \rightsquigarrow \ (a \in G^\delta); \text{ let } e = (e_1, e_2) \ \} \\
& \langle B_1, t\rangle \xrightarrow{(e_1,a,t_a)}_* \ \wedge \ \langle B_1, t\rangle \rightsquigarrow \langle B_1', t'\rangle \ \wedge \ \langle B_2, t\rangle \xrightarrow{(e_2,a,t_a)}_* \ \wedge \ \langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle \\
\Rightarrow \quad & \{ \text{ induction hypothesis } \} \\
& d_{min}(a, B_1') = d_{min}(a, B_1) \ominus (t'-t) \ \wedge \ d_{min}(a, B_2') = d_{min}(a, B_2) \ominus (t'-t)
\end{aligned}
$$

$\Leftrightarrow$ { Definition 6.20 ($a \in G^\delta$) }
$d_{min}(a, B'_1 \,||_G\, B'_2) = max(d_{min}(a, B_1) \ominus (t'-t), d_{min}(a, B_2) \ominus (t'-t))$
$\Leftrightarrow$ { $max(x \ominus z, y \ominus z) = max(x, y) \ominus z$; Definition 6.20 ($a \in G^\delta$) }
$d_{min}(a, B'_1 \,||_G\, B'_2) = d_{min}(a, B_1 \,||_G\, B_2) \ominus (t'-t)$. □

6.5 Correspondence with causality-based semantics

The main aim of this section is to prove the consistency between the denotational semantics of PA_U in terms of urgent event structures and its event-based operational semantics as induced by the inference rules for $\rightsquigarrow$ and $\longrightarrow$. The consistency proof is carried out in two steps, similar as in Chapter 5 where we dealt with PA_T. First, an (operational) characterization, denoted $\mathcal{T}'_U[\![B]\!]$, is provided of the set of traces of the tuple $\langle B, t \rangle$ as generated by the event-based operational semantics. This is done in Section 6.5.1. Here, the main difficulty is to correctly characterize the set of timed event traces of $\mathcal{U}_U(B)$ without using the d_{min} function that is used in the inference rules for this construct. In Section 6.5.2 a second, though denotational, characterization (denoted $\mathcal{T}_U[\![B]\!]$) is presented of the set of traces as generated by $\rightsquigarrow$ and $\longrightarrow$. The main reason for providing a second characterization is to facilitate the consistency proof; it follows that both characterizations denote identical sets of timed event traces, i.e., $\mathcal{T}_U = \mathcal{T}'_U$. Finally, in Section 6.5.3 it is shown that the set of timed event traces of urgent event structure $\mathcal{E}_U[\![B]\!]$ coincides with $\mathcal{T}_U[\![B]\!]$. This proves the consistency between the causality-based and operational semantics of PA_U.

6.5.1 Operational characterization of timed event traces

The following lemma characterizes the timed traces (under $\longrightarrow_*$) of $\langle B, t \rangle$ where $B \in \mathsf{PA}_U$ in an operational way. The presence of urgency has an important impact on the characterization of timed traces for $B_1 + B_2$ and $B_1 \,[>\, B_2$; it is not difficult to check that the characterizations for the other operators in PA_U are equal to those for PA_T (cf. Lemma 5.25). For $+$ and $[>$ states can be reached (under $\longrightarrow_*$) for which there is an outgoing branch labelled with an urgent action the timing of which avoids the occurrence of a competitive alternative. E.g., for

$$(2)\ a\,;\ (5)\ b\,;\ \mathbf{0} + \mathcal{U}_c((7)\ c\,;\ \mathbf{0})$$

$(a, 8)\,(b, 13)$ is not a legal trace since c will prevent a from occurring at any time later than 7. In general, a trace σ ($\sigma \neq \varepsilon$) of B_1 is also a trace of $B_1 + B_2$ provided B_2 cannot initially perform an urgent action at any time earlier than the time of the *first* event in σ. By symmetry, an analogous reasoning applies to traces of B_2.

Replacing $+$ by $[>$ in the above behaviour expression yields:

$$(2)\ a\,;\ (5)\ b\,;\ \mathbf{0}\,[>\, \mathcal{U}_c((7)\ c\,;\ \mathbf{0})$$

Here, c will prevent a from occurring at any time later than 7. In general, a trace σ ($\sigma \neq \varepsilon$) of B_1 is also (part of) a trace of $B_1 \,[\!> B_2$ provided that for *each* event e_i in σ behaviour B_2 cannot initially perform an urgent action at any time earlier than t_i. For

$$\mathcal{U}_a((2)\ a\,;\,(5)\ b\,;\,\mathbf{0})\,[\!>\,(7)\ c\,;\,\mathbf{0}$$

$(c, 7)$ is not a trace since a is forced at time 2 and should precede c. In general, trace $\sigma_1\,\sigma_2$ with σ_i a trace of B_i is a trace of $B_1 \,[\!> B_2$ iff σ_1 does not contain a successful termination event, and if for the first event e_1 in σ_2 there does not exist an urgent action in B_1 after σ_1 that could occur earlier than t_1.

It is technically convenient to introduce a function that determines the minimal time instant at which behaviour B at time t can perform an urgent event.

6.29. DEFINITION. $\mathsf{mt}(B, t) \triangleq \mathit{Min}\{\, t_a \mid \exists\, a \in \mathsf{Urgent}(B) : \langle B, t\rangle \xrightarrow{(e_a, a, t_a)}_* \}$. $\square$

The timed event traces generated by $\longrightarrow_*$ can now be characterized as follows. We only provide full characterizations for choice, disrupt, and urgency. For the other constructs the characterization of Lemma 5.25 remains to hold.

6.30. LEMMA. For trace σ, behaviours B, B_1 and $B_2 \in \mathsf{PA}_U$, and $t, t'' \in \mathsf{Time}$ we have:

1. $\langle B_1 + B_2, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff either
   (i) $\sigma = \varepsilon \ \wedge\ \langle B_1, t\rangle \xrightarrow{\varepsilon}_* \langle B_1', t'\rangle \ \wedge\ \langle B_2, t\rangle \xrightarrow{\varepsilon}_* \langle B_2', t'\rangle \ \wedge\ B' = B_1' + B_2'$, or
   (ii) $\langle B_1, t\rangle \xrightarrow{\sigma}_* \langle B_1', t'\rangle \ \wedge\ B' = B_1' \ \wedge\ \sigma = (e_a, a, t_a)\,\sigma'' \ \wedge\ t_a \leqslant \mathsf{mt}(B_2, t)$, or
   (iii) $\langle B_2, t\rangle \xrightarrow{\sigma}_* \langle B_2', t'\rangle \ \wedge\ B' = B_2' \ \wedge\ \sigma = (e_a, a, t_a)\,\sigma'' \ \wedge\ t_a \leqslant \mathsf{mt}(B_1, t)$.
2. $\langle B_1 \,[\!> B_2, t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff either
   (i) $\sigma = \varepsilon \ \wedge\ \langle B_1, t\rangle \xrightarrow{\varepsilon}_* \langle B_1', t'\rangle \ \wedge\ \langle B_2, t\rangle \xrightarrow{\varepsilon}_* \langle B_2', t'\rangle \ \wedge\ B' = B_1' \,[\!> B_2'$, or
   (ii) $\sigma = \sigma'\,(e, \delta, t') \ \wedge\ \langle B_1, t\rangle \xrightarrow{\sigma}_* \langle B_1', t'\rangle \ \wedge\ t' \leqslant \mathsf{mt}(B_2, t) \ \wedge\ B' = B_1'$, or
   (iii) $\sigma = (e, a, t_a)\,\sigma' \ \wedge\ \langle B_2, t\rangle \xrightarrow{\sigma}_* \langle B_2', t'\rangle \ \wedge\ t_a \leqslant \mathsf{mt}(B_1, t) \ \wedge\ B' = B_2'$, or
   (iv) $\sigma = \sigma_1\,(e, a, t_a)\,\sigma_2 \ \wedge\ \langle B_1, t\rangle \xrightarrow{\sigma_1 (e, a, t_a)}_* \langle B_1', t_a\rangle \ \wedge\ a \neq \delta \ \wedge\ t_a \leqslant \mathsf{mt}(B_2, t) \ \wedge$ $\langle B_2, t_a\rangle \xrightarrow{\sigma_2}_* \langle B_2', t'\rangle \ \wedge\ (\sigma_2 = (e, b, t_b)\,\sigma' \ \Rightarrow\ t_b \leqslant \mathsf{mt}(B_1', t_a) \ \wedge\ B' = B_2') \ \wedge$ $(\sigma_2 = \varepsilon \ \Rightarrow\ B' = B_1')$
3. $\langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma}_* \langle B', t'\rangle$ iff $\langle B, t\rangle \xrightarrow{\sigma}_* \langle B'', t'\rangle \ \wedge\ B' = \mathcal{U}_U(B'')$ and
   $\forall\, 0 < i \leqslant |\,\sigma\,| : (\forall\, a \in U, t_a < t_i : \langle B, t\rangle \overset{\sigma_i (e, a, t_a)}{\not\longrightarrow}_*)$.

PROOF. The proof is by induction on the length of σ. As an illustration we provide the proof for urgency. The proofs for $+$ and $[\!>$ are similar and omitted. Consider $\mathcal{U}_U(B)$.
Base: For $\sigma = \varepsilon$ we derive:

$$\begin{array}{ll} & \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\varepsilon}_* \langle B', t'\rangle \\ \Leftrightarrow & \{\ \text{definition } \xrightarrow{\varepsilon}_*\ \} \\ & \langle \mathcal{U}_U(B), t\rangle \rightsquigarrow \langle B', t'\rangle \end{array}$$

$$
\begin{array}{ll}
\Leftrightarrow & \{ \text{ SOS-rules for } \rightsquigarrow \} \\
& \langle B,t\rangle \rightsquigarrow \langle B'',t'\rangle \ \wedge \ B' = \mathcal{U}_U(B'') \\
\Leftrightarrow & \{ \text{ definition } \xrightarrow{\varepsilon}_* \} \\
& \langle B,t\rangle \xrightarrow{\varepsilon}_* \langle B'',t'\rangle \ \wedge \ B' = \mathcal{U}_U(B'') \quad .
\end{array}
$$

Induction Step: '$\Rightarrow$' : assume the lemma holds for n and suppose $\sigma = \sigma' \ (e_{n+1}, a_{n+1}, t_{n+1})$ with $\sigma' = (e_1, a_1, t_1) \dots (e_n, a_n, t_n)$, $n \geqslant 0$. The proof is by contradiction. That is, assume that for some i, $0 < i \leqslant n{+}1$ we have that

$$
\langle B,t\rangle \xrightarrow{\sigma_i\ (e,a,t_a)}_* \quad , \tag{6.1}
$$

for $a \in U$ and $t_a < t_i$. Because σ' is an event trace of $\langle \mathcal{U}_U(B), t\rangle$ it follows from the induction hypothesis that $i > n$, since for all $i \leqslant n$ (6.1) does not hold. Thus, $i = n{+}1$. We derive starting from (6.1):

$$
\begin{array}{ll}
& \langle B,t\rangle \xrightarrow{\sigma_i\ (e,a,t_a)}_* \\
\Leftrightarrow & \{ \text{ definition } \xrightarrow{\sigma}_*;\ i = n{+}1 \ \} \\
& \exists\, C : \langle C,t_n\rangle \xrightarrow{(e,a,t_a)}_* \\
\Rightarrow & \{ \text{ Theorem 6.24 } \} \\
& t_a \geqslant t_n + d_{min}(a,C) \\
\Rightarrow & \{ \ t_a < t_{n+1} \ ; \text{ calculus } \} \\
& d_{min}(a,C) < t_{n+1} - t_n \quad .
\end{array}
$$

Thus from the assumption it follows that $d_{min}(a,C) < t_{n+1} - t_n$, for $a \in U$. We now infer:

$$
\begin{array}{ll}
& \langle \mathcal{U}_U(B),t\rangle \xrightarrow{\sigma}_* \langle B',t'\rangle \\
\Leftrightarrow & \{ \text{ definition } \xrightarrow{\sigma}_* \text{ using that } \sigma = \sigma' \ (e_{n+1},a_{n+1},t_{n+1}) \ \} \\
& \exists\, B'' : \langle \mathcal{U}_U(B),t\rangle \xrightarrow{\sigma'}_* \langle B'',t_n\rangle \xrightarrow{(e_{n+1},a_{n+1},t_{n+1})}_* \langle B',t_{n+1}\rangle \\
\Leftrightarrow & \{ \text{ SOS-rules for } \rightsquigarrow \text{ and } \longrightarrow \ \} \\
& \exists\, C,D : \langle \mathcal{U}_U(B),t\rangle \xrightarrow{\sigma'}_* \langle \mathcal{U}_U(C),t_n\rangle \xrightarrow{(e_{n+1},a_{n+1},t_{n+1})}_* \langle \mathcal{U}_U(D),t_{n+1}\rangle \\
\Leftrightarrow & \{ \text{ induction hypothesis; SOS-rules for } \rightsquigarrow \text{ and } \longrightarrow \ \} \\
& \langle B,t\rangle \xrightarrow{\sigma'}_* \langle C,t_n\rangle \xrightarrow{(e_{n+1},a_{n+1},t_{n+1})}_* \langle D,t_{n+1}\rangle \ \wedge \ (\forall\, a \in U : t_{n+1}{-}t_n < d_{min}(a,C)).
\end{array}
$$

This, however, contradicts with the fact that for $a \in U$ we have $t_{n+1} - t_n > d_{min}(a,C)$ as derived above. Contradiction.

'$\Leftarrow$' : assume the lemma holds for n ($n \geqslant 0$) and suppose $\sigma = \sigma' \ (e_{n+1}, a_{n+1}, t_{n+1})$ with $\sigma' = (e_1, a_1, t_1) \dots (e_n, a_n, t_n)$. We then derive:

$$
\begin{array}{ll}
& \langle B,t\rangle \xrightarrow{\sigma}_* \langle B',t'\rangle \ \wedge \ (\forall\, 0 < i \leqslant |\,\sigma\,| : (\forall\, t_a < t_i, a \in U : \langle B,t\rangle \overset{\sigma_i\ (e,a,t_a)}{\not\longrightarrow}_*)) \\
\Leftrightarrow & \{ \text{ definition } \xrightarrow{\sigma}_* \ ; \ \sigma = \sigma' \ (e_{n+1},a_{n+1},t_{n+1}) \ ; \ t' = t_{n+1} \ \} \\
& \quad \langle B,t\rangle \xrightarrow{\sigma'}_* \langle B'',t_n\rangle \xrightarrow{(e_{n+1},a_{n+1},t_{n+1})}_* \langle B',t_{n+1}\rangle \\
& \wedge \ (\forall\, 0 < i \leqslant |\,\sigma'\,| : (\forall\, t_a < t_i, a \in U : \langle B,t\rangle \overset{\sigma'_i\ (e,a,t_a)}{\not\longrightarrow}_*)) \\
& \wedge \ (\forall\, t_a < t_{n+1}, a \in U : \langle B,t\rangle \overset{\sigma'\ (e,a,t_a)}{\not\longrightarrow}_*)
\end{array}
$$

$$
\begin{array}{ll}
\Rightarrow & \{ \text{ induction hypothesis } \} \\
& \quad \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma'}_* \langle \mathcal{U}_U(B''), t_n\rangle \ \wedge\ \langle B'', t_n\rangle \xrightarrow{(e_{n+1}, a_{n+1}, t_{n+1})}_* \langle B', t_{n+1}\rangle \\
& \wedge\ (\forall\, t_a < t_{n+1}, a \in U : \langle B, t\rangle \overset{\sigma'\ (e,a,t_a)}{\not\longrightarrow}_*) \\
\Leftrightarrow & \{ \text{ Definition 5.24; calculus } \} \\
& \quad \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma'}_* \langle \mathcal{U}_U(B''), t_n\rangle \ \wedge\ \langle B'', t_n\rangle \rightsquigarrow \langle B''', t_{n+1}\rangle \xrightarrow{(e_{n+1}, a_{n+1})} \langle B', t_{n+1}\rangle \\
& \wedge\ \neg\, (\exists\, t_a < t_{n+1}, a \in U : \langle B, t\rangle \xrightarrow{\sigma'\ (e,a,t_a)}_*) \\
\Leftrightarrow & \{\ \langle B, t\rangle \xrightarrow{\sigma'}_* \langle B'', t_n\rangle\ \} \\
& \quad \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma'}_* \langle \mathcal{U}_U(B''), t_n\rangle \ \wedge\ \langle B'', t_n\rangle \rightsquigarrow \langle B''', t_{n+1}\rangle \xrightarrow{(e_{n+1}, a_{n+1})} \langle B', t_{n+1}\rangle \\
& \wedge\ \neg\, (\exists\, t_a < t_{n+1}, a \in U : \langle B'', t_n\rangle \xrightarrow{(e,a,t_a)}_*) \\
\Rightarrow & \{ \text{ Theorem 6.24 } \} \\
& \quad \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma'}_* \langle \mathcal{U}_U(B''), t_n\rangle \ \wedge\ \langle B'', t_n\rangle \rightsquigarrow \langle B''', t_{n+1}\rangle \xrightarrow{(e_{n+1}, a_{n+1})} \langle B', t_{n+1}\rangle \\
& \wedge\ \neg\, (\exists\, t_a < t_{n+1}, a \in U : t_a \geqslant t_n + d_{min}(a, B'')) \\
\Rightarrow & \{ \text{ calculus } \} \\
& \quad \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma'}_* \langle \mathcal{U}_U(B''), t_n\rangle \ \wedge\ \langle B'', t_n\rangle \rightsquigarrow \langle B''', t_{n+1}\rangle \xrightarrow{(e_{n+1}, a_{n+1})} \langle B', t_{n+1}\rangle \\
& \wedge\ (\forall\, a \in U : t_{n+1} - t_n \leqslant d_{min}(a, B'')) \\
\Leftrightarrow & \{ \text{ SOS-rules for } \rightsquigarrow \} \\
& \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma'}_* \langle \mathcal{U}_U(B''), t_n\rangle \rightsquigarrow \langle \mathcal{U}_U(B'''), t_{n+1}\rangle \ \wedge\ \langle B''', t_{n+1}\rangle \xrightarrow{(e_{n+1}, a_{n+1})} \langle B', t_{n+1}\rangle \\
\Leftrightarrow & \{ \text{ SOS-rule for } \longrightarrow \} \\
& \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma'}_* \langle \mathcal{U}_U(B''), t_n\rangle \rightsquigarrow \langle \mathcal{U}_U(B'''), t_{n+1}\rangle \xrightarrow{(e_{n+1}, a_{n+1})} \langle \mathcal{U}_U(B'), t_{n+1}\rangle \\
\Leftrightarrow & \{ \text{ Definition 5.24; } \sigma = \sigma'\ (e_{n+1}, a_{n+1}, t_{n+1})\ ;\ t' = t_{n+1}\ \} \\
& \langle \mathcal{U}_U(B), t\rangle \xrightarrow{\sigma}_* \langle \mathcal{U}_U(B'), t_{n+1}\rangle \quad .
\end{array}
$$

□

6.5.2 Denotational characterization of timed event traces

We now characterize denotationally the set of timed event traces of B obtained from applying the inference rules for $\rightsquigarrow$ and $\longrightarrow$, and prove that this characterization coincides with the operational characterization of the previous section. For the purpose of the consistency proof it suffices to only consider $\langle B, 0\rangle$. We use B as an abbreviation of $\langle B, 0\rangle$. For technical convenience we introduce

6.31. DEFINITION. $\mathsf{mt}'(B) \triangleq Min\{\, t_a \mid \exists\, a \in \mathsf{Urgent}(B) : (e_a, a, t_a) \in \mathcal{T}_U[\![\, B \,]\!]\,\}$. □

The denotational characterization of the set of timed traces of B is defined as follows:

6.32. DEFINITION. For $B \in \mathsf{PA}_U$ the set of timed traces of B, $\mathcal{T}_U[\![\, B \,]\!]$, is defined by:

1. $\mathcal{T}_U[\![\, \mathbf{0} \,]\!] \triangleq \{\, \varepsilon \,\}$
2. $\mathcal{T}_U[\![\, \surd_\xi \,]\!] \triangleq \{\, \varepsilon \,\} \cup \{\, (\xi, \delta, t) \mid t \in \mathsf{Time} \,\}$
3. $\mathcal{T}_U[\![\, (t)\ a_\xi\ ;\ B \,]\!] \triangleq \{\, (\xi, a, t')\, {}^{t'}[\, \sigma \,] \mid t' \geqslant t\ \wedge\ \sigma \in \mathcal{T}_U[\![\, B \,]\!] \,\} \cup \{\, \varepsilon \,\}$

4. $\mathcal{T}_U[\![B_1 + B_2]\!] \triangleq \{ (\xi, a, t)\, \sigma \in \mathcal{T}_U[\![B_1]\!] \mid t \leqslant \mathsf{mt}'(B_2) \} \cup$
$\{ (\xi, a, t)\, \sigma \in \mathcal{T}_U[\![B_2]\!] \mid t \leqslant \mathsf{mt}'(B_2) \} \cup \{ \varepsilon \}$
5. $\mathcal{T}_U[\![B_1 >> B_2]\!] \triangleq \{ \sigma_1\, (e, \tau, t)\, {}^t[\, \sigma_2 \,] \mid \sigma_1\, (e, \delta, t) \in \mathcal{T}_U[\![B_1]\!] \ \wedge\ \sigma_2 \in \mathcal{T}_U[\![B_2]\!] \}$
$\cup \{ \sigma \in \mathcal{T}_U[\![B_1]\!] \mid \sigma \neq \sigma'\, (e, \delta, t) \}$
6. $\mathcal{T}_U[\![B_1 [> B_2]\!] \triangleq \{ \sigma\, (e, \delta, t) \in \mathcal{T}_U[\![B_1]\!] \mid t \leqslant \mathsf{mt}'(B_2) \} \cup \{ \varepsilon \} \cup$
$\{ (e, a, t)\, \sigma \in \mathcal{T}_U[\![B_2]\!] \mid t \leqslant \mathsf{mt}'(B_1) \} \cup$
$\{ \sigma_1\, \sigma_2 \mid \sigma_1 = \sigma_1'\, (e, a, t) \in \mathcal{T}_U[\![B_1]\!] \ \wedge\ a \neq \delta \ \wedge\ t \leqslant \mathsf{mt}'(B_2) \ \wedge$
$\sigma_2 \in \mathcal{T}_U[\![B_2]\!] \ \wedge\ (\sigma_2 = (e', b, t')\, \sigma_2' \ \Rightarrow\ t' \geqslant t \ \wedge$
$(\forall\, c \in \mathsf{Urgent}(B_1), t'' < t' : \sigma_1\, (e', c, t'') \notin \mathcal{T}_U[\![B_1]\!])) \}$
7. $\mathcal{T}_U[\![B[H]]\!] \triangleq \{ \sigma \mid \exists\, \sigma' \in \mathcal{T}_U[\![B]\!] : \sigma = \sigma'[H] \}$
8. $\mathcal{T}_U[\![B \setminus G]\!] \triangleq \{ \sigma \mid \exists\, \sigma' \in \mathcal{T}_U[\![B]\!] : \sigma = \sigma' \setminus G \}$
9. $\mathcal{T}_U[\![B_1 ||_G B_2]\!] \triangleq \{ \sigma \in (\overline{\mathcal{T}_U[\![B_1]\!]} \bowtie_G \overline{\mathcal{T}_U[\![B_2]\!]})^+ \mid \pi_i(\sigma) \in \mathcal{T}_U[\![B_i]\!] \text{ for } i{=}1,2 \}$
10. $\mathcal{T}_U[\![\mathcal{U}_U(B)]\!] \triangleq \{ \sigma \in \mathcal{T}_U[\![B]\!] \mid \forall\, i : (\forall\, e, a \in U, t_a{<}t_i : \sigma_i\, (e, a, t_a) \notin \mathcal{T}_U[\![B]\!]) \}$.

$\square$

6.33. LEMMA. $\forall\, B \in \mathsf{PA}_U : \mathcal{T}_U[\![B]\!] = \{ \sigma \mid \exists\, B', t' : \langle B, 0 \rangle \xrightarrow{\sigma}_* \langle B', t' \rangle \}$.

PROOF. Straightforward but tedious by induction on the structure of B. $\square$

6.5.3 Consistency between causality-based and operational semantics

We now come to the following consistency result between the causality-based semantics of PA_U and the event-based operational semantics.

6.34. THEOREM. $\forall\, B \in \mathsf{PA}_U : T_U(\mathcal{E}_U[\![B]\!]) = \mathcal{T}_U[\![B]\!]$.

PROOF. The proof is by induction on the structure of B.
Base: For $B = \mathbf{0}$ and $B = \surd$ the theorem trivially holds.
Induction Step: Assume the theorem holds for B_1 and B_2. Let $\Psi = \mathcal{E}_U[\![B]\!]$ and $\Psi_i = \mathcal{E}_U[\![B_i]\!] = \langle (E_i, \rightsquigarrow_i, \mapsto_i, l_i), \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$ (i=1, 2).

1. $B = (t)\ a_\xi\, ; B_1$. We have $\Psi = \langle \mathcal{E}_T[\![B]\!], \mathcal{U}_1 \cup \{ (\xi, \text{false}) \} \rangle$. Event ξ is nonurgent and all urgent events in Ψ_1 can only occur after the occurrence of ξ, and thus cannot prevent ξ from appearing from a certain time on. The non-empty timed traces σ of Ψ are thus of the form $(\xi, a, t_a)\, {}^{t_a}[\, \sigma' \,]$ with $\sigma' \in T_U(\Psi_1)$ and $t_a \geqslant t$ (see also proof of Lemma 5.17). By the induction hypothesis we have $T_U(\Psi_1) = \mathcal{T}_U[\![B_1]\!]$. This proves the case.

2. $B = B_1 + B_2$. In Ψ mutual conflicts are introduced between all initial events of Ψ_1 and Ψ_2. This means that initial urgent events of Ψ_1 are put into conflict with (all) initial events of Ψ_2 (and vice versa for urgent events in Ψ_2 and initial events in Ψ_1), and as a result may prevent the occurrence of these initial events in Ψ_2 after a certain time. For e_1, e_2 initial events of Ψ_1 and Ψ_2, respectively, such that $\mathcal{U}_1(e_1)$ event e_2 becomes excluded in Ψ by e_1 from time t on,

$\mathcal{D}_1(e_1) < t$. Thus we derive:

$$
\begin{aligned}
& T_U(\mathcal{E}_U[\![B_1 + B_2]\!]) \\
= \quad & \{ \text{ see discussion above } \} \\
& \quad T_U(\Psi_1) \setminus \{ (e,a,t)\,\sigma \mid \exists\, e' \in E_2 : \mathcal{U}_2(e') \ \wedge\ e \rightsquigarrow e' \ \wedge\ \mathcal{D}_2(e') < t \} \\
& \cup T_U(\Psi_2) \setminus \{ (e,a,t)\,\sigma \mid \exists\, e' \in E_1 : \mathcal{U}_1(e') \ \wedge\ e \rightsquigarrow e' \ \wedge\ \mathcal{D}_1(e') < t \} \\
= \quad & \{ \text{ calculus } \} \\
& \quad \{ (e,a,t)\,\sigma \in T_U(\Psi_1) \mid \neg(\exists\,(e',b,t') \in T_U(\Psi_2) : \mathcal{U}_2(e') \ \wedge\ t' < t) \} \\
& \cup \{ (e,a,t)\,\sigma \in T_U(\Psi_2) \mid \neg(\exists\,(e',b,t') \in T_U(\Psi_1) : \mathcal{U}_1(e') \ \wedge\ t' < t) \} \cup \{ \varepsilon \} \\
= \quad & \{ \text{ induction hypothesis } ; \mathcal{U}(e) \ \Rightarrow\ l(e) \in \mathsf{Urgent}(B) \ \} \\
& \quad \{ (e,a,t)\,\sigma \in \mathcal{T}_U[\![B_1]\!] \mid \neg(\exists\,(e',b,t') \in \mathcal{T}_U[\![B_2]\!] : b \in \mathsf{Urgent}(B_2) \ \wedge\ t' < t) \} \\
& \cup \{ (e,a,t)\,\sigma \in \mathcal{T}_U[\![B_2]\!] \mid \neg(\exists\,(e',b,t') \in \mathcal{T}_U[\![B_1]\!] : b \in \mathsf{Urgent}(B_1) \ \wedge\ t' < t) \} \cup \{ \varepsilon \} \\
= \quad & \{ \text{ Definition 6.31 } \} \\
& \{ (e,a,t)\,\sigma \in \mathcal{T}_U[\![B_1]\!] \mid t \leqslant \mathsf{mt}'(B_2) \} \cup \{ (e,a,t)\,\sigma \in \mathcal{T}_U[\![B_2]\!] \mid t \leqslant \mathsf{mt}'(B_1) \} \cup \{ \varepsilon \} \\
= \quad & \{ \text{ Definition 6.32 } \} \\
& \mathcal{T}_U[\![B_1 + B_2]\!] \quad .
\end{aligned}
$$

3. $B = B_1 >> B_2$. Similar to the nonurgent case, timed traces of Ψ are either (i) traces of Ψ_1 that do not end with a successful termination event δ (this is equal to saying that no δ should occur in this trace), or (ii) traces of the form $\sigma_1\,(e,\tau,t)\,{}^t[\,\sigma_2\,]$ for $\sigma_2 \in T_U(\Psi_2)$ and $\sigma_1\,(e,\delta,t) \in T_U(\Psi_1)$. The fact that events in Ψ_2 can only occur after the successful termination of Ψ_1 guarantees that urgent events in Ψ_2 do not affect the occurrence of events in Ψ_1 (and vice versa). Thus, $T_U(\Psi)$ equals

$$
\begin{aligned}
& \quad \{ \sigma \in T_U(\Psi_1) \mid \sigma \neq \sigma'\,(e,\delta,t) \} \\
& \cup \{ \sigma_1\,(e,\tau,t)\,{}^t[\,\sigma_2\,] \mid \sigma_1(e,\delta,t) \in T_U(\Psi_1) \ \wedge\ \sigma_2 \in T_U(\Psi_2) \} \quad .
\end{aligned}
$$

By the induction hypothesis it now directly follows that this equals

$$
\begin{aligned}
& \quad \{ \sigma \in \mathcal{T}_U[\![B_1]\!] \mid \sigma \neq \sigma'\,(e,\delta,t) \} \\
& \cup \{ \sigma_1\,(e,\tau,t)\,{}^t[\,\sigma_2\,] \mid \sigma_1\,(e,\delta,t) \in \mathcal{T}_U[\![B_1]\!] \ \wedge\ \sigma_2 \in \mathcal{T}_U[\![B_2]\!] \} \quad .
\end{aligned}
$$

By Definition 6.32 this equals $\mathcal{T}_U[\![B_1 >> B_2]\!]$.

4. $B = B_1\,[> B_2$. From the timed case without urgency (see Chapter 5) we know that traces of Ψ are either (i) traces of Ψ_1 that end with a successful termination event δ, or (ii) concatenations of (possibly empty) traces $\sigma_1 \in T_U(\Psi_1)$ and $\sigma_2 \in T_U(\Psi_2)$ where σ_1 does not contain a δ-event and where the timing of each event in σ_2 should exceed the maximal timing in σ_1. In the urgent case the asymmetric conflicts between the events in Ψ_1 and $init(\Psi_2)$ do affect the occurrence of events. That is, an event in Ψ_1 can happen only provided there is no (initial) urgent event in Ψ_2 that could occur earlier, and an (initial) event in Ψ_2 can happen provided there is no urgent event in Ψ_1 after σ_1 that could occur earlier. We now characterize set (i) and derive for this set:

$$
\begin{aligned}
& \{ \sigma\,(e,\delta,t) \in T_U(\Psi_1) \mid \neg(\exists\, i, e' \in init(\Psi_2) : \mathcal{U}_2(e') \ \wedge\ \mathcal{D}_2(e') < t_i) \} \\
= \quad & \{ \text{ all traces in } T_U(\Psi_1) \text{ are time-consistent } \}
\end{aligned}
$$

$$
\begin{aligned}
& \{ \sigma\,(e,\delta,t) \in T_U(\Psi_1) \mid \neg(\exists\, e' \in \mathit{init}(\Psi_2) : \mathcal{U}_2(e') \;\wedge\; \mathcal{D}_2(e') < t) \} \\
= \;\; & \{ \text{ calculus } \} \\
& \{ \sigma\,(e,\delta,t) \in T_U(\Psi_1) \mid \neg(\exists\,(e', l_2(e'), t') \in T_U(\Psi_2) : \mathcal{U}_2(e') \;\wedge\; t' < t) \} \\
= \;\; & \{ \text{ induction hypothesis; } \mathcal{U}_2(e') \;\Rightarrow\; l_2(e') \in \mathsf{Urgent}(B_2) \} \\
& \{ \sigma\,(e,\delta,t) \in \mathcal{T}_U[\![B_1]\!] \mid \neg(\exists\,(e', a, t') \in \mathcal{T}_U[\![B_2]\!] : a \in \mathsf{Urgent}(B_2) \;\wedge\; t' < t) \} \\
= \;\; & \{ \text{ Definition 6.31 } \} \\
& \{ \sigma\,(e,\delta,t) \in \mathcal{T}_U[\![B_1]\!] \mid t \leqslant \mathsf{mt}'(B_2) \} \quad .
\end{aligned}
$$

A similar derivation can be carried out for set (ii). By Definition 6.32 the union of the thus obtained sets is equal to $\mathcal{T}_U[\![B_1 [> B_2]\!]$.

5. $B = B_1 \setminus G$. Similar to the nonurgent case, the timed traces of Ψ are the timed traces in Ψ_1 where all action labels in G are renamed into τ. So, $T_U(\Psi) = \{ \sigma \mid \exists\, \sigma' \in T_U(\Psi_1) : \sigma = \sigma' \setminus G \}$. By the induction hypothesis this equals $\{ \sigma \mid \exists\, \sigma' \in \mathcal{T}_U[\![B_1]\!] : \sigma = \sigma' \setminus G \}$, which—by Definition 6.32—equals $\mathcal{T}_U[\![B_1 \setminus G]\!]$.

   The proof for relabelling is similar and omitted here.

6. $B = B_1 \,||_G\, B_2$. Since, according to Definition 4.26, $G \cap \mathsf{Urgent}(B_i) = \varnothing$, for i=1, 2, it is easy to check that no new (asymmetric) conflicts are introduced between urgent events in Ψ_1 and events in Ψ_2 (or vice versa). This means that, similar to the timed case, $\sigma \in T_U(\Psi)$ iff $\pi_i(\sigma) \in T_U(\Psi_i)$, for i=1, 2, and σ is time-consistent. So, $T_U(\Psi)$ equals

   $$\{ \sigma \in (\overline{T_U(\Psi_1)} \;\bowtie_G\; \overline{T_U(\Psi_2)})^+ \mid \pi_1(\sigma) \in T_U(\Psi_1) \;\wedge\; \pi_2(\sigma) \in T_U(\Psi_2) \}.$$

   By the induction hypothesis this equals

   $$\{ \sigma \in (\overline{\mathcal{T}_U[\![B_1]\!]} \bowtie_G \overline{\mathcal{T}_U[\![B_2]\!]})^+ \mid \pi_1(\sigma) \in \mathcal{T}_U[\![B_1]\!] \;\wedge\; \pi_2(\sigma) \in \mathcal{T}_U[\![B_2]\!] \}.$$

   By Definition 6.32 this equals $\mathcal{T}_U[\![B_1 \,||_G\, B_2]\!]$.

7. $B = \mathcal{U}_U(B_1)$. All events in Ψ_1 labelled with an action in U become urgent in Ψ. No additional conflicts and/or bundles are introduced. This means that a trace σ of Ψ_1 is also a trace of Ψ iff (i) each event e_i in σ with $l_1(e_i) \in U$ cannot be performed any earlier, and (ii) for each event e_i in σ there is not an enabled urgent event that could be performed earlier (cf. the constraints in Definition 6.3). We now derive

$$
\begin{aligned}
& T_U(\mathcal{E}_U[\![\mathcal{U}_U(B_1)]\!]) \\
= \;\; & \{ \text{ discussion above } \} \\
& \{ \sigma \in T_U(\Psi_1) \mid (\forall\,(e_i, a_i, t_i) \in \overline{\sigma} : a_i \in U \;\Rightarrow\; t_i = \mathsf{time}(\sigma_i, e_i)) \;\wedge \\
& \quad (\forall\, 0 < i \leqslant |\, \sigma \,| : e \in \mathsf{en}([\sigma_i]) \;\wedge\; l_1(e) \in U \;\Rightarrow\; t_i \leqslant \mathsf{time}(\sigma_i, e)) \} \\
= \;\; & \{ \text{ calculus } \} \\
& \{ \sigma \in T_U(\Psi_1) \mid (\forall\,(e_i, a_i, t_i) \in \overline{\sigma}, t < t_i : a_i \in U \;\Rightarrow\; \sigma_i\,(e_i, a_i, t) \notin T_U(\Psi_1)) \;\wedge \\
& \quad (\forall\, 0 < i \leqslant |\, \sigma \,|, t < t_i : l_1(e) \in U \;\Rightarrow\; \sigma_i\,(e, l_1(e), t) \notin T_U(\Psi_1)) \} \\
= \;\; & \{ \text{ calculus } \} \\
& \{ \sigma \in T_U(\Psi_1) \mid \forall\, i : (\forall\, e, t < t_i, a \in U : \sigma_i\,(e, a, t) \notin T_U(\Psi_1)) \} \\
= \;\; & \{ \text{ induction hypothesis } \}
\end{aligned}
$$

$$\{\sigma \in \mathcal{T}_U[\![B_1]\!] \mid \forall i : (\forall e, t < t_i, a \in U : \sigma_i\,(e,a,t) \notin \mathcal{T}_U[\![B_1]\!])\}$$
$$= \quad \{ \text{ Definition 6.32 } \}$$
$$\mathcal{T}_U[\![\mathcal{U}_U(B_1)]\!] \quad .$$

□

Let $\mathsf{TS}_U(B)$ be the timed event transition system obtained by applying the inference rules to B. For $\mathcal{E}_U[\![B]\!]$ a transition system ETS_U is constructed in the following way. States of the transition system for $\mathcal{E}_U[\![B]\!]$ are reachable urgent event structures (or, derivatives) of $\mathcal{E}_U[\![B]\!]$ with $\mathcal{E}_U[\![B]\!]$ being the initial state. There is a transition from Ψ to Ψ' if $\Psi' = \Psi[\sigma]$ for timed trace σ with $\mid \sigma \mid = 1$. (See Section 5.3 for a formalization of these issues.)

The previous theorem implies that $\mathsf{TS}_U(B)$ and $\mathsf{ETS}_U(\mathcal{E}_U[\![B]\!])$ are (timed) event trace equivalent. It is easy to check that for each transition $B \xrightarrow{(e,a,t)}_* B'$ there is a unique way in which this transition is derived from the SOS-rules for $\rightsquigarrow$ and $\longrightarrow$. Since—in addition—both (timed) event transition systems are deterministic it follows that:

6.35. THEOREM. $\forall B \in \mathsf{PA}_U : \mathsf{TS}_U(B) \sim \mathsf{ETS}_U(\mathcal{E}_U[\![B]\!])$.

PROOF. Similar to the proof of Theorem 2.46. □

6.6 Related work

This section discusses some related work in the literature that deals with the incorporation of a notion of urgency in a timed process algebra. We deliberately state 'notion of' since it appears that there are several closely related concepts around.

The basic timing ingredient in PA_U is the delay function, $(t)\ a\,;B$. It specifies that from t time units on since the occurrence of the causal predecessor of a (if any) the behaviour is able to engage in a. This type of time constraint is sometimes referred to as *unbounded idling* [112] or *loose time prefix* [105], since the time between the expiration of the specified delay and the occurrence of a is determined by the environment, and in principle may be unbounded. Opposed to this principle the notion of *time-stamped actions* has been proposed by, amongst others, Baeten & Bergstra [7]. For example, $[t]\ a\,;B$ specifies that a *must* occur at time t. In fact, this construct specifies a delay t, and in addition, that a must occur at t—*local* urgency, so to say.

Urgent and nonurgent actions are incorporated by Bolognesi & Lucidi [18] within a (discrete) timed variant of LOTOS. In their proposal each action is nonurgent by default, but can be made urgent—like in our case. Opposed to the proposal of this chapter they do allow synchronization of urgent actions. Such synchronizations only succeed if all participants are ready to engage in the interaction at the same time instant. If this is not the case then a so-called *timelock* appears, a situation in which the progress of time is blocked as a result of which the entire system may halt execution.

Bolognesi *et al.* [19] generalize the notion of urgency (in a continuous time setting) by introducing the **time** operator. **time** $a(t_1, t_2)$ **in** B denotes behaviour B in which a must occur in interval $[t_1, t_2]$ once it is enabled. $\mathcal{U}_a(B)$ is akin to (**time** $a(0,0)$ **in** B) $\backslash a$. In order to constrain the passage of time in the inference rules for **time** $a(t_1, t_2)$ **in** B, Bolognesi *et al.* use a function $age(a, B)$ which determines the maximal (rather than the minimal) time at which B can perform initial action a. (It was pointed out by Bolognesi that the operational semantics of PA_U strongly resembles an (unpublished) proposal by Bolognesi & Schneider [20] to integrate timed LOTOS [18] and timed CSP [133].)

Klusener [86] introduces a real-time variant of ACP, called ACPur, and provides an operational semantics using separate time and action transitions. Rather than using a function like d_{min} to block the passage of time in presence of urgent actions, he uses negative premises. Klusener defines several notions of bisimulation for ACPur and presents an axiomatization for it.

In other approaches (for instance, Hansson & Jonsson [65], Hennessy & Regan [67] and Schneider [133]) a weaker notion of urgency, often referred to as *maximal progress*, is present. The notion of maximal progress (or eagerness) is weaker than urgency as it 'ignores urgency in the context of choice'. That is, if actions happen they happen as soon as possible, but they cannot prevent the occurrence of other actions after a certain amount of time (like urgent actions do). In most formalisms maximal progress is coupled with hiding ($\backslash$). In these formalisms internalized events become eager, and eager events are internal. The rational for this is that when an event is internalized (i.e., hidden from the environment), no further interaction on this event can take place, no further delays will be imposed by the environment, and thus there is no reason to delay its execution any further. On the one hand, the maximal progress assumption applied to internal events alone is not sufficiently expressive (why can't non-internal events be eager?), and on the other hand, it is a bit restrictive—when specifying an unreliable communication service that may lose messages, the loss of a message is usually modelled by an internal event, but we are not interested in specifying when this message is physically lost!

6.7 Conclusion

The notion of urgency was introduced by Bolognesi & Lucidi [18] in the context of (discrete) timed LOTOS and later by Bolognesi *et al.* [19] in a dense timed setting. In this chapter we have investigated the incorporation of urgency in the setting of event structures by distinguishing between urgent and nonurgent events. The resulting model has been used to provide a denotational semantics to a timed process algebra that includes an urgency operator akin to the one proposed for timed LOTOS. The corresponding event-based operational semantics turned out to strongly resemble the inference rules in [19]. The main difference is that we do not allow synchronization on urgent actions, while in [19] this is possible at the prize of possible timelocks.

7 The real-time module

In this chapter we generalize timed event structures by equipping events and bundles with sets of time instants and use urgent events for the sole purpose of modelling timeout mechanisms (thus restricting urgent event structures). An event e with set T of time instants denotes that e can only occur at some $t \in T$ since the start of the system. T associated with bundle $X \mapsto e$ denotes that the time between the occurrence of an event in X and the appearance of e should equal t, for some $t \in T$. The result is a causality-based model allowing the specification of minimal, maximal and, for instance, periodic time constraints. This chapter generalizes the theory of Chapter 4 and uses urgent events in a controlled way. It investigates how the more expressive model, baptized real-time event structures, can be used as a vehicle to provide a semantics to a real-time process algebra including timeout and watchdog operators.

7.1 Introduction

In Chapter 4 we introduced a simple timed variant of event structures by associating a single time instant to events and bundles. These time instants specify only lower bounds of occurrence and do not allow for constraining the latest point in time at which an event may occur. In addition, this model does not allow to specify timeout mechanisms, a necessary ingredient for describing real-time systems. Therefore, in this chapter we propose a model, called *real-time* event structures, which allows to specify upper bounds of occurrence (in addition to lower bounds) and allows to specify timeouts.

Let us first reconsider the timed event structure model. An event e with delay t denotes that e can happen from t time units on since the start of the system. This is, in fact, a shorthand notation for event e equipped with a set T of time instants, $T = \{ t' \mid t' \geqslant t \}$, with the interpretation that e can happen at any time instant in T. Of course, a similar observation can be made for bundle delays. In this chapter we generalize this point of view by allowing arbitrary *sets* of time instants to be assigned to events and bundles. In this way, it is not only possible to specify the minimal time at which an event can occur, but also the latest time at which it can occur.

In order to specify *timeouts* we use urgent events, like we introduced in Chapter 6. Opposed to PA_U, the process algebra of Chapter 6, that allows to enforce an arbitrary action in an expression to be urgent we restrict the introduction of urgency to timeout mechanisms only. In this way, the model of Chapter 6 can be simplified significantly, while suiting our purposes.

We will also show how *timed interrupts* (or *watchdogs* [112]) can be modelled without using urgent events.

This chapter is organized as follows. In Section 7.2 the notion of real-time event structures is introduced and it is investigated to what extent the results and definitions related to timed event structures still apply. Section 7.3 extends PA_T by generalizing the delay function and incorporating timeout and watchdog operators; it presents a causality-based and event-based operational semantics for the resulting formalism and shows the correspondence between them. Section 7.4 discusses related work in the field of extending partial-order models with time. Finally, Section 7.5 presents the conclusions of this chapter.

7.2 Real-time event structures

7.1. NOTATION. For $x, y \in \mathsf{Time}$ let $[x, y]$ abbreviate $\{ t \mid x \leqslant t \leqslant y \}$ and $(x, y]$ abbreviate $\{ t \mid x < t \leqslant y \}$. For $x \in \mathsf{Time}$ and $y \in \mathsf{Time}^\infty$ let (x, y) be a shorthand for $\{ t \mid x < t < y \}$ and $[x, y)$ a shorthand for $\{ t \mid x \leqslant t < y \}$. $[x, \infty)$ is often abbreviated as x. □

In order to facilitate the specification of other than minimal time constraints we replace event and bundle delays by arbitrary, and possibly infinite, *sets* of time instants. The interpretation of $\{ e_a \} \overset{T}{\mapsto} e_b$, where T is a set of time instants, is that if e_a happens at t_a, then e_b is possible at $t_a + t$, for any $t \in T$. Notice that for $T = [t, \infty)$ the interpretation of $\{ e_a \} \overset{T}{\mapsto} e_b$ is equal to $\{ e_a \} \overset{t}{\mapsto} e_b$ in the model of Chapter 4.

For events that have more than one bundle pointing to them we take the following interpretation. Consider $\{ e_a \} \overset{T}{\mapsto} e_c$ and $\{ e_b \} \overset{T'}{\mapsto} e_c$. Then, if e_a happens at time t_a and e_b at time t_b, then e_c is enabled at any $t \in (t_a{+}T) \cap (t_b{+}T')$, where $t{+}T$ denotes $\{ t{+}t' \mid t' \in T \}$. (If $T = [t, \infty)$ and $T' = [t', \infty)$ then $(t_a{+}T) \cap (t_b{+}T') = [max(t_a{+}t, t_b{+}t'), \infty)$; so the synchronization principle of Chapter 4 is retained.) When the intersection of two (or more) sets of time instants is empty this means that the event at hand cannot occur at any time and will be permanently disabled.

The interpretation of an event with delay T is that e can happen at some time $t \in T$ since the start of the system. As before we use $\mathcal{D}$ and $\mathcal{T}$ to associate delays (which are now sets of time instants) to events and bundles, respectively.

In order to be able to model timeouts we equip the model with urgent events (like in Chapter 6). In Chapter 6 we introduced urgent event structures and did not constrain the introduction of urgent events in the model. As we have shown, urgent events may have a global impact which alleviates one of the interesting characteristics of event structures, viz. the *locality* aspect. This resulted in a rather complex characterization of timed event trace: in order to decide whether an event can happen the 'global' state of the entire system is used (cf. the third and fourth constraint of Definition 6.3). In this chapter we restrict the introduction of urgent events thus yielding a simpler model. Later on in this chapter we will show that this 'weakened' variant of urgency suffices to model timeouts and watchdogs.

Let $X \rightsquigarrow e'$ abbreviate $(\forall\, e'' \in X : e'' \rightsquigarrow e')$. Note that $\varnothing \rightsquigarrow e'$ for all e'.

7.2. Definition. (*Real-time event structure*)

A *real-time* event structure is a quadruple $\langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ with

- $\mathcal{E}$, an (extended bundle) event structure $(E, \leadsto, \mapsto, l)$
- $\mathcal{D} : E \longrightarrow \mathcal{P}(\mathsf{Time}^\infty)$, the *event delay* function
- $\mathcal{T} : \mapsto \longrightarrow \mathcal{P}(\mathsf{Time}^\infty)$, the *bundle delay* function
- $\mathcal{U} : E \longrightarrow \mathsf{Bool}$, the *urgency* predicate

such that for all $e \in E$ with $\mathcal{U}(e)$:

1. $\forall\, e' \in E, X \subseteq E : ((e' \leadsto e \;\vee\; e \leadsto e') \;\wedge\; X \mapsto e) \;\Rightarrow\; (X \mapsto e' \;\vee\; X \leadsto e')$
2. $\exists\, t \in \mathsf{Time} : \mathcal{D}(e) \subseteq [t,t] \;\vee\; (\exists\, X \subseteq E : X \overset{T}{\mapsto} e \;\wedge\; T \subseteq [t,t])$.

□

The first constraint requires that the enablings of an urgent event e are either contained in the enablings of an event e' that it disables, i.e., $e' \leadsto e$, or that an enabling of e is disabled by e' (the case $e \leadsto e'$ is identical). This constraint enforces that as soon as e' is enabled either e is also enabled (provided e is not disabled in another way), or is permanently disabled, since some enabling of e is disabled (by e'). As a result the global impact of urgent events is limited; see also the discussion in Section 6.2.1. Thus, in order to decide whether e' can occur—once it is enabled—it suffices to consider the local (and urgent) disablings of e'.

The second constraint ensures that urgent events are enabled at at most a single time instant. The motivation for this constraint is that urgent events are used for the sole purpose of modelling timeouts, and timeouts typically can appear at a single time instant only.

Note that event and bundle delays may be infinite sets of time instants, but also empty sets of time instants. We usually will use intervals and combinations (unions and intersections) of them. For depicting real-time event structures we use the same conventions as for timed event structures. We use Λ to denote a real-time event structure and EBES_R to denote the class of real-time event structures. We use T to range over $\mathcal{P}(\mathsf{Time}^\infty)$.

7.2.1 Timed event traces

Given a sequence σ of timed events and an enabled event e, that is $e \in \mathsf{en}([\sigma])$, let $\mathsf{time}(\sigma, e)$ denote the set of time instants at which e can occur.

7.3. Definition. For σ a sequence of timed events $(e_1, t_1) \ldots (e_n, t_n)$ with $e_i \in E$, $t_i \in \mathsf{Time}$, for all $0 < i \leqslant n$, and $e \in \mathsf{en}([\sigma])$, let

$$\begin{aligned}\mathsf{time}(\sigma, e) \triangleq {} & \bigcap\,(\{\, \mathcal{D}(e) \,\} \cup H_1 \cup H_2) \text{ where}\\ & H_1 = \{\, t_j + T \mid \exists\, X \subseteq E : X \overset{T}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \{\, e_j \,\}\,\}\\ & H_2 = \{\, [t_j, \infty) \mid \exists\, e_j \in \overline{[\sigma]} : e_j \leadsto e \,\} \quad .\end{aligned}$$

□

For P a set of sets of time instants let $\bigcap P \triangleq \{\, t \mid \forall T \in P : t \in T \,\}$.

7.4. Lemma. For all sequences σ of timed events and $e \in E$ we have:

$$e \in \mathsf{en}([\sigma]) \ \wedge\ \mathcal{U}(e) \ \Rightarrow \ |\,\mathsf{time}(\sigma, e)\,| \leqslant 1 \quad .$$

Proof. This follows directly from the second constraint of Definition 7.2 and the definition of time. □

In the sequel we will use for urgent event e $\mathsf{time}(\sigma, e)$ as a value, rather than as a set of time instants, whenever appropriate. We use ∞ as the value of $\varnothing$.

7.5. Definition. (*Timed event trace (revisited)*)

A *timed event trace* of real-time event structure $\Lambda = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ is a sequence σ of timed events $(e_1, t_1) \ldots (e_n, t_n)$ with $e_i \in E$, $t_i \in \mathsf{Time}$, for all $0 < i \leqslant n$, satisfying

1. $e_1 \ldots e_n \in T(\mathcal{E})$
2. $\forall\, i : t_i \in \mathsf{time}(\sigma_i, e_i)$
3. $\forall\, i, e : (e \in \mathsf{en}([\sigma_i]) \ \wedge\ \mathcal{U}(e) \ \wedge\ (e_i \rightsquigarrow e \ \vee\ e \rightsquigarrow e_i)) \ \Rightarrow\ t_i \leqslant \mathsf{time}(\sigma_i, e)$.

□

The first two constraints are self-explanatory. The third constraint is justified as follows. First, consider $e_i \rightsquigarrow e$, for urgent e. Then the third constraint takes care of the fact that urgent event e prevents the events that it disables (such as e_i) to occur after a certain time. That is, event e_i can occur at time t_i provided there is no enabled urgent event e that disables e_i and that must occur before t_i. In the case that $e \rightsquigarrow e_i$ and both e and e_i are enabled, the third constraint ensures that e_i can only occur if urgent e cannot occur earlier. If e could occur earlier it should precede (and cause) e_i.

7.6. Example. For the following sequences of timed events the conditions are given under

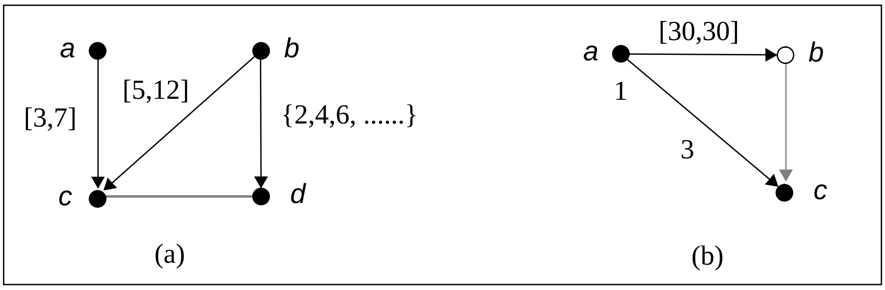


Figure 7.1: Two example real-time event structures.

which they are timed event traces of Figure 7.1(a):

$$(e_a, t_a)\,(e_b, t_b)\,(e_d, t_d) \quad \text{if} \quad t_d \in \{\, t_b{+}2, t_b{+}4, \ldots \,\}, \ \text{and}$$
$$(e_a, t_a)\,(e_b, t_b)\,(e_c, t_c) \quad \text{if} \quad max(t_a{+}3, t_b{+}5) \leqslant t_c \leqslant min(t_a{+}7, t_b{+}12).$$

For Figure 7.1(b) we obtain:

$$\begin{array}{rl} (e_a, t_a)\,(e_c, t_c) & \text{if} \quad t_a \geqslant 1 \ \wedge \ t_a + 3 \leqslant t_c \leqslant t_a + 30, \ \text{and} \\ (e_a, t_a)\,(e_b, t_b)\,(e_c, t_c) & \text{if} \quad t_a \geqslant 1 \ \wedge \ t_b = t_a + 30 \ \wedge \ t_c \geqslant max(t_a + 3, t_b). \end{array}$$

□

Like for the simple timed case, timed event traces do respect causality, but not necessarily time. Ill-timed traces only appear as a result of the interleaving of causally independent events. Let $T_R(\Lambda)$ denote the set of timed event traces of Λ.

7.7. THEOREM. *Ill-timed theorem (revisited)*

For $t' < t$: $\sigma\ (e, t)(e', t')\ \sigma' \in T_R(\Lambda) \ \Rightarrow\ \sigma\ (e', t')(e, t)\ \sigma' \in T_R(\Lambda)$.

PROOF. Let $\sigma^1 = \sigma\ (e, t)(e', t')\ \sigma'$ and $\sigma^2 = \sigma\ (e', t')(e, t)\ \sigma'$. Let $t' < t$ and $\sigma^1 \in T_R(\Lambda)$. The proof is by contradiction. Suppose $\sigma^2 \notin T_R(\Lambda)$. This can only be because one of the following reasons:

1. $[\sigma^2] \notin T(\mathcal{E})$. Identically to the proof of Theorem 4.9 this leads to a contradiction.
2. $\exists\, j : t_j \notin \mathsf{time}(\sigma^2_j, e_j)$. In a similar way as in the proof of Theorem 4.9 it can be proven that this leads to a contradiction.
3. $\exists\, j, e'' : e'' \in \mathsf{en}([\sigma^2_j])$ with $\mathcal{U}(e'')$ such that $e_j \rightsquigarrow e''$ (or $e'' \rightsquigarrow e_j$) and $t_j > \mathsf{time}(\sigma^2_j, e'')$. For event e_j in σ or σ' this leads to a contradiction; otherwise $\sigma^1 \notin T_R(\Lambda)$. It remains to check $e_j = e$ and $e_j = e'$:
   (a) $e_j \rightsquigarrow e'' \ \wedge\ e_j = e$. Then $e'' = e'$ is the interesting case; if $e'' \neq e'$ we would have $\sigma^1 \notin T_R(\Lambda)$, which is a contradiction. If $e'' = e'$ then $e' \in \mathsf{en}([\sigma^2_j])$ which is impossible since e' is an event in the prefix of σ^2 of e. Contradiction.
   (b) $e_j \rightsquigarrow e'' \ \wedge\ e_j = e'$. Then $e'' = e$ is the interesting case; if $e'' \neq e$ we would have $\sigma^1 \notin T_R(\Lambda)$, which is a contradiction. If $e'' = e$ then $e' \rightsquigarrow e$ and e could not precede e' in σ^1, so $\sigma^1 \notin T_R(\Lambda)$. Contradiction.
   (c) $e'' \rightsquigarrow e_j \ \wedge\ e_j = e$. For e'' in σ or e'' in σ' this would contradict $\sigma^1 \in T_R(\Lambda)$. So, let $e'' = e'$. Then $e' \rightsquigarrow e$ which means that e could not precede e' in σ^1. Contradiction.
   (d) $e'' \rightsquigarrow e_j \ \wedge\ e_j = e'$. Again, the interesting case is $e'' = e$; the other cases contradict $\sigma^1 \in T_R(\Lambda)$. Then $e \notin \mathsf{en}([\sigma^2_j])$ since e' disables e. Contradiction.

□

7.2.2 Families of lposets

As an underlying semantical model for real-time event structures we use lposets. The lposets of Λ, denoted $L_R(\Lambda)$, are generated in the same way as for timed event structures, cf. Definition 4.18.

7.8. DEFINITION. (*Lposets of a real-time event structure*)

For $\Lambda \in \mathsf{EBES}_R : L_R(\Lambda) \ \triangleq\ \{\, \langle \overline{\sigma}, \bigcap_{\sigma' \in [\sigma]_{\sim_T}} <^*_{\sigma'}, l \restriction \overline{\sigma} \rangle \mid \sigma \in T_R(\Lambda)\,\}$. □

We sometimes use $L_R(\sigma)$ to denote $\langle \overline{\sigma}, \bigcap_{\sigma' \in [\sigma]_{\sim_T}} <^*_{\sigma'}, l \restriction \overline{\sigma} \rangle$.

For real-time event structures we have that having the same families of lposets is equivalent to having the same sets of timed event traces.

7.9. THEOREM. $\forall\, \Lambda, \Lambda' \in \mathsf{EBES}_R : L_R(\Lambda) = L_R(\Lambda') \iff T_R(\Lambda) = T_R(\Lambda')$.

PROOF. Straightforward and omitted. □

In the real-time setting the untimed lposets of Λ are not necessarily equal to the lposets of the corresponding untimed event structure $\varphi(\Lambda)$ (i.e., $\mathcal{E}$). The reason for this is that events—though causally enabled—may not appear since there is no time instant at which they can occur (or because an urgent event prevents them from occurring). E.g., if Λ consists of a single event e with $\mathcal{D}(e) = \varnothing$ then $L_R(\Lambda)$ only consists of the empty lposet whereas the corresponding untimed event structure also has lposet $\boxed{e}$.

7.10. LEMMA. $\forall\, \Lambda \in \mathsf{EBES}_R : L(\Lambda) \subseteq L(\varphi(\Lambda))$.

PROOF. Straightforward from the fact that $\forall\, \sigma \in T_R(\Lambda) : [\sigma] \in T(\varphi(\Lambda))$. □

7.2.3 Real-time remainder

The remainder of a real-time event structure is defined as a straightforward generalization of Definition 4.22.

7.11. DEFINITION. (*Real-time remainder*)

The *remainder* of real-time event structure $\Lambda = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ after timed event trace σ, is $\Lambda[\sigma] = \langle \mathcal{E}', \mathcal{D}', \mathcal{T}', \mathcal{U}' \rangle$ where

- $\mathcal{E}' = \mathcal{E}[[\sigma]] = (E', \rightsquigarrow', \mapsto', l')$
- $\forall\, e \in E' : \mathcal{D}'(e) = \bigcap\, (\{\, \mathcal{D}(e) \,\} \cup H_1 \cup H_2)$ with
$$H_1 = \{\, t_j + T \mid \exists\, X \subseteq E : X \overset{T}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \{\, e_j \,\}\,\} \text{ and}$$
$$H_2 = \{\, [t_j, \infty) \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow e \,\}$$
- $\mathcal{T}' = (\mathcal{T} \restriction \mapsto') \cup \{\, ((\varnothing, e), T) \mid \varnothing \mapsto' e \,\}$ for some $T \in \mathcal{P}(\mathsf{Time}^\infty)$
- $\mathcal{U}' = \mathcal{U} \restriction E'$.

□

The fact that $\Lambda[\sigma]$ is a real-time event structure follows from:

7.12. LEMMA. $\forall\, \Lambda \in \mathsf{EBES}_R, \sigma \in T_R(\Lambda) : \Lambda[\sigma] \in \mathsf{EBES}_R$.

PROOF. Let $\Lambda = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ with $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$, and $\Lambda' = \Lambda[\sigma]$. It follows directly that $\mathcal{E}' \in \mathsf{EBES}$, since $\mathcal{E}[[\sigma]] \in \mathsf{EBES}$ for all $\mathcal{E} \in \mathsf{EBES}$ and $[\sigma] \in T(\mathcal{E})$. For each $e \in E'$, functions $\mathcal{D}'$ and $\mathcal{U}'$ are defined, and $\mathcal{T}'$ is a total function on the bundles of Λ'. It remains to verify that the constraints of Definition 7.2 are satisfied. Let $e \in E'$ with $\mathcal{U}'(e)$.

1. $\forall e' \in E', X \subseteq E' : ((e' \rightsquigarrow' e \vee e \rightsquigarrow' e') \wedge X \mapsto' e) \Rightarrow (X \mapsto' e' \vee X \rightsquigarrow' e')$. Distinguish between $e' \rightsquigarrow' e$ and $e \rightsquigarrow' e'$.

   (a) $e' \rightsquigarrow' e$. Then, by Definition 2.28, we have $e' \rightsquigarrow e$. Since $\mathcal{U}'(e)$ we have $\mathcal{U}(e)$. Now let $X \mapsto' e$. If $X \mapsto e$ then we also have that $X \mapsto e'$, since $\Lambda \in \mathsf{EBES}_R$. In addition, since $X \mapsto e$ and $X \mapsto' e$ it follows that $X \cap \overline{[\sigma]} = \varnothing$, and so $X \mapsto' e'$. In case $X \mapsto' e$, but $X \mapsto e$ does not exist, then $X \mapsto' e$ is a new bundle, and it follows from Definition 2.28 that $X = \varnothing$. But then $X \rightsquigarrow' e'$ since $\varnothing \rightsquigarrow' e'$ for all e'.

   (b) $e \rightsquigarrow' e'$. Similar to the case $e' \rightsquigarrow' e$.

2. $\exists\, t : \mathcal{D}'(e) \subseteq [t,t] \vee (\exists\, X \subseteq E' : X \overset{T}{\mapsto}{}' e \wedge T \subseteq [t,t])$. Since $\mathcal{U}'(e)$ we have $\mathcal{U}(e)$. Suppose $\mathcal{D}(e) \subseteq [t,t]$. By the definition of remainder it follows directly that $\mathcal{D}'(e) \subseteq [t,t]$. Suppose $X \overset{T}{\mapsto} e$ with $T \subseteq [t,t]$. If $X \mapsto' e$ then the bundle delay is unaffected and the constraint is satisfied. In case $(X,e) \not\in \mapsto'$ it follows from Definition 2.28 that $X \cap \overline{[\sigma]} \neq \varnothing$, say $X \cap \overline{[\sigma]} = \{\, e_j \,\}$. But then $\mathcal{D}(e)$ will be intersected by $t_j + T$, and since $T \subseteq [t,t]$ then it follows that $\mathcal{D}'(e) \subseteq [t',t']$ for $t' = t_j + t$. This proves the case.

$\square$

The following correctness result concerning real-time remainders is analogous to the correctness results for timed and urgent remainders.

7.13. THEOREM. *Correctness of real-time remainder*

For $\sigma \in T_R(\Lambda)$ and σ' a sequence of timed events:

1. $\sigma' \in T_R(\Lambda[\sigma]) \Longleftrightarrow \sigma\,\sigma' \in T_R(\Lambda)$
2. $\sigma' \in T_R(\Lambda[\sigma]) \;\Rightarrow\; L_R(\sigma)$ is a prefix of $L_R(\sigma\,\sigma')$.

PROOF. Let $\Lambda = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ and $\Lambda' = \Lambda[\sigma] = \langle \mathcal{E}', \mathcal{D}', \mathcal{T}', \mathcal{U}' \rangle$ for $\sigma \in T_R(\Lambda)$ where $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$ and $\mathcal{E}' = (E', \rightsquigarrow', \mapsto', l')$.

1. '$\Rightarrow$': Assume $\sigma' \in T_R(\Lambda')$. We prove that $\sigma\,\sigma' \in T_R(\Lambda)$ by systematically checking the constraints of being a timed event trace (cf. Definition 7.5).

   (a) $[\sigma\,\sigma'] \in T(\mathcal{E})$. Given that $[\sigma] \in T(\mathcal{E})$ and $[\sigma'] \in T(\mathcal{E}')$ this follows directly from Theorem 2.30.

   (b) $\forall\, i : t_i \in \mathsf{time}((\sigma\,\sigma')_i, e_i)$. This is proven in a similar way as in the proof of Theorem 6.13.

   (c) For the third constraint of Definition 7.5 we derive for all e:

$$
\begin{aligned}
&\forall\, i : e \in \mathsf{en}([(\sigma\,\sigma')_i]) \wedge \mathcal{U}(e) \wedge (e_i \rightsquigarrow e \vee e \rightsquigarrow e_i) \Rightarrow t_i \leqslant \mathsf{time}((\sigma\,\sigma')_i, e)\\
\Leftrightarrow\;& \{\text{ domain split }\}\\
&(\forall\, 0 < i \leqslant |\,\sigma\,| : e \in \mathsf{en}([(\sigma\,\sigma')_i]) \wedge \mathcal{U}(e) \wedge (e_i \rightsquigarrow e \vee e \rightsquigarrow e_i) \Rightarrow\\
&\qquad\qquad t_i \leqslant \mathsf{time}((\sigma\,\sigma')_i, e)) \;\wedge\\
&(\forall\, |\,\sigma\,| < i \leqslant |\,\sigma\,\sigma'\,| : e \in \mathsf{en}([(\sigma\,\sigma')_i]) \wedge \mathcal{U}(e) \wedge (e_i \rightsquigarrow e \vee e \rightsquigarrow e_i) \Rightarrow\\
&\qquad\qquad t_i \leqslant \mathsf{time}((\sigma\,\sigma')_i, e))\\
\Leftrightarrow\;& \{\text{ calculus }\}\\
&\quad(\forall\, i : e \in \mathsf{en}([\sigma_i]) \wedge \mathcal{U}(e) \wedge (e_i \rightsquigarrow e \vee e \rightsquigarrow e_i) \Rightarrow t_i \leqslant \mathsf{time}(\sigma_i, e))
\end{aligned}
$$

$$
\begin{aligned}
&\qquad \wedge\ (\forall j : e \in \mathsf{en}([\sigma\,\sigma'_j])\ \wedge\ \mathcal{U}(e)\ \wedge\ (e_j \rightsquigarrow e\ \vee\ e \rightsquigarrow e_j)\ \Rightarrow\ t_j \leqslant \mathsf{time}(\sigma\,\sigma'_j, e))\\
\Leftrightarrow\ &\ \{\ \text{Lemma 6.12; Lemma 6.11}\ \}\\
&\qquad\ (\forall i : e \in \mathsf{en}([\sigma_i])\ \wedge\ \mathcal{U}(e)\ \wedge\ (e_i \rightsquigarrow e\ \vee\ e \rightsquigarrow e_i)\ \Rightarrow\ t_i \leqslant \mathsf{time}(\sigma_i, e))\\
&\qquad \wedge\ (\forall j : e \in \mathsf{en}'([\sigma'_j])\ \wedge\ \mathcal{U}(e)\ \wedge\ (e_j \rightsquigarrow e\ \vee\ e \rightsquigarrow e_j) \Rightarrow t_j \leqslant \mathsf{time}'(\sigma'_j, e))\\
\Leftrightarrow\ &\ \{\ \mathcal{U}'(e) = \mathcal{U}(e) \text{ for } e \in E';\ \rightsquigarrow' = \rightsquigarrow \cap (E' \times E')\ \}\\
&\qquad\ (\forall i : e \in \mathsf{en}([\sigma_i])\ \wedge\ \mathcal{U}(e)\ \wedge\ (e_i \rightsquigarrow e\ \vee\ e \rightsquigarrow e_i)\ \Rightarrow\ t_i \leqslant \mathsf{time}(\sigma_i, e))\\
&\qquad \wedge\ (\forall j : e \in \mathsf{en}'([\sigma'_j])\ \wedge\ \mathcal{U}'(e)\ \wedge\ (e_j \rightsquigarrow' e\ \vee\ e \rightsquigarrow' e_j)\ \Rightarrow\ t_j \leqslant \mathsf{time}'(\sigma'_j, e))\\
\Leftarrow\ &\ \{\ \text{Definition 7.5}\ \}\\
&\quad\ \sigma \in T_R(\Lambda)\ \wedge\ \sigma' \in T_R(\Lambda')\quad .
\end{aligned}
$$

'$\Leftarrow$': the proof for this case goes along similar lines as the proof for '$\Rightarrow$'.

2. The proof of this theorem is analogous to the proof of Theorem 4.24.

□

7.2.4 Transformation rules

The first rule allows for the transformation of (particular) events that are impossible due to timing constraints into events that can never occur due to causal conditions that can never be met ($\star$ denotes an arbitrary element of $\mathcal{P}(\mathsf{time}^\infty)$). Since we can always safely remove events with an empty bundle pointing to them, this rule is considered to be useful. The second rule facilitates the removal of sub-bundles and is a generalization of a similar rule for the simple timed case. The third rule allows for the recalculation of event delays. T associated to bundle set X means that $\bigcup_{e \in X} \mathcal{D}(e)$ equals T. $T{+}T'$ equals $\bigcup_{t \in T}(t + T')$, where $\bigcup_{t \in T}(t + T')$ equals $\varnothing$ if $T = \varnothing$. Note that these rules do not depend on the fact whether e is urgent or not (except that in the first rule $\star$ should equal $\varnothing$ or $[t, t]$, for some $t \in \mathsf{Time}$, if e is urgent; otherwise the result is not necessarily a real-time event structure).

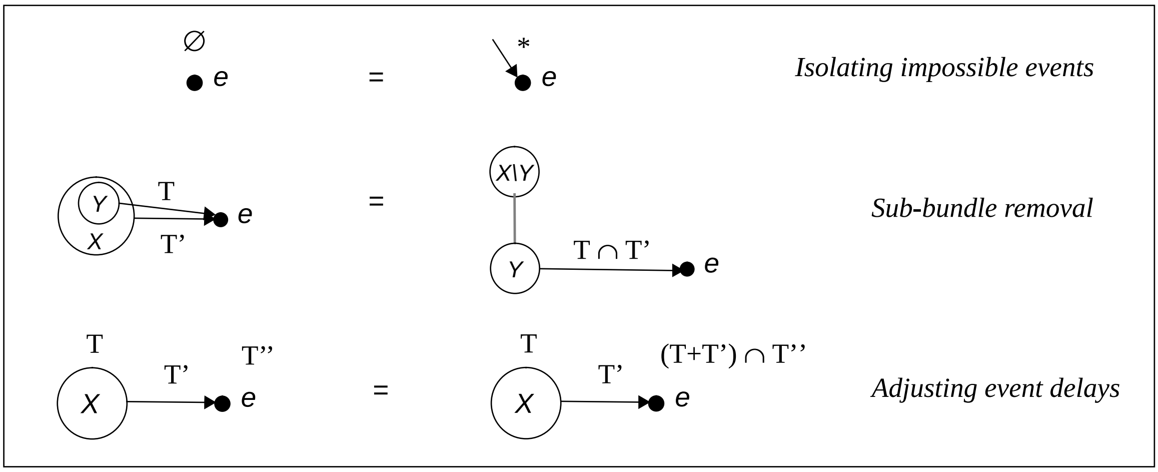


Figure 7.2: Some transformation rules for real-time event structures.

7.14. THEOREM. Real-time event structure $\langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ is lposet-equivalent with

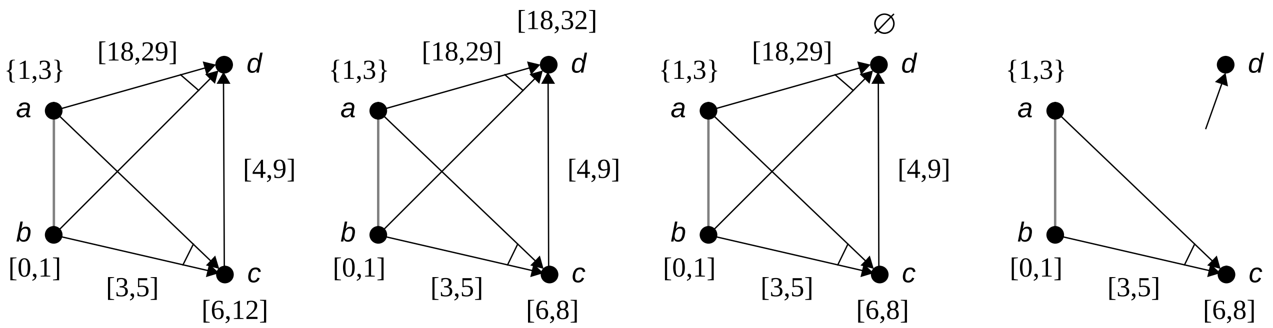
1. $\langle (E, \rightsquigarrow, \mapsto \cup \{ (\varnothing, e) \}, l), (\mathcal{D} \setminus \{ (e, \varnothing) \}) \cup \{ (e, \star) \}, \mathcal{T} \cup \{ ((\varnothing, e), \star) \}, \mathcal{U} \rangle$, if $\mathcal{D}(e) = \varnothing$.
2. $\langle (E, \rightsquigarrow, \mapsto \setminus \{ (X, e) \}, l), \mathcal{D}, (\mathcal{T} \setminus \{ ((Y, e), T), ((X, e), T') \}) \cup \{ ((Y, e), T \cap T') \}, \mathcal{U} \rangle$ if $Y \subseteq X \ \wedge \ Y \overset{T}{\mapsto} e \ \wedge \ X \overset{T'}{\mapsto} e$.
3. $\langle (E, \rightsquigarrow, \mapsto, l), (\mathcal{D} \setminus \{ (e, T'') \}) \cup \{ (e, (\bigcup_{e' \in X} \mathcal{D}(e') + T') \cap T'') \}, \mathcal{T}, \mathcal{U} \rangle$ if $X \overset{T'}{\mapsto} e \ \wedge \ \mathcal{D}(e) = T''$.

PROOF. Similar to the proof of Theorem 4.25. □

The next transformation rule is a consequence of the third and first rules of Theorem 7.14 and the rule for untimed event structures which allows to eliminate bundles pointing to impossible events:

This rule can be proven by first transforming the event delay of e into $\varnothing$ according to the adjusting event delays rule, then introducing an empty bundle pointing to e according to the first rule and subsequently eliminating the impossible event e according to the rule superfluous bundles for untimed event structures (see Chapter 2).

7.15. EXAMPLE. The application of the transformation rules of Figure 7.2 is shown by the following example:



(a) (b) (c) (d)

Suppose we initially have real-time event structure (a). In the first step we consider the bundles originating from $\{ e_a, e_b \}$ and adjust the event delays of e_c and e_d according to the third transformation rule. This yields (b). Then we apply the same rule to $\{ e_c \} \mapsto e_d$, resulting in (c). Finally, we can remove all bundles pointing to e_d according to the above derived transformation rule, and obtain (d). □

7.3 A real-time process algebra

This section introduces an extension of PA_T by generalizing the delay function and introducing a timeout and watchdog operator. In the first section we introduce the syntax; a causality-based semantics is provided in the second section. Subsequently, an event-based operational semantics is presented in the same spirit as in Chapter 5 using timed actions (and using separate time and action transitions), and the consistency of this semantics with respect to the denotational semantics is proven.

7.3.1 Syntax

Let $t \in \mathsf{Time}$ and $T \in \mathcal{P}(\mathsf{Time}^\infty)$. The syntax of the real-time process algebra PA_R is defined as follows.

7.16. Definition. (*Real-time process algebra* PA_R)

$$B \ ::= \ \mathbf{0} \ \big| \ \surd \ \big| \ (T)\ a\,;B \ \big| \ B + B \ \big| \ B\,||_G\,B \ \big| \ B[H] \ \big| \ B \setminus G \ \big| \ B >> B \ \big| \ B\,[> B \ \big| \ B \overset{t}{\rhd} B \ \big| \ B \overset{t}{\blacktriangleright} B. \qquad \Box$$

The precedences of the composition operators are, in decreasing binding order: ; , + and $\rhd$, $||$, $[>$ and $\blacktriangleright$, $>>$, $\setminus$ and $[]$. Parentheses are omitted if this does not introduce ambiguities.

The delay function that expresses the relative delay of an action associates to an action a set T of time instants. Behaviour $a\,;(T)\ b\,;\mathbf{0}$ is able to engage in b at t time units since the occurrence of a, for $t \in T$. That is, if a happens at t_a then b can happen at $t_a{+}t$ for some $t \in T$. Like for PA_T we allow arithmetic expressions on sets of time instants. We abbreviate $([t,\infty))\ a$ by $(t)\ a$, and $(0)\ a$ simply by a.

$B_1 \overset{t}{\rhd} B_2$ is a *timeout* operator; initially the behaviour behaves like B_1, but if B_1 does not perform any action before or at t (since the enabling of this behaviour) then the control is passed to B_2. $B_1 \overset{t}{\rhd} B_2$ can be considered as a timed generalization of $B_1 + B_2$: if during $[0,t)$ behaviour B_1 performs an action then the choice is resolved in favour of B_1, if it does not perform any action in $[0,t]$ then the choice is resolved in favour of B_2. At time t a nondeterministic choice appears between B_1 and B_2.

$B_1 \overset{t}{\blacktriangleright} B_2$ is a *watchdog* operator; initially the behaviour behaves like B_1 but at time t control is passed to B_2 provided B_1 is not yet successfully terminated. $B_1 \overset{t}{\blacktriangleright} B_2$ is a timed generalization of $B_1\,[> B_2$: B_1 is aborted at time t by B_2 provided that B_1 has not successfully terminated.

Note that in $B_1 \overset{t}{\rhd} B_2$ control is passed to B_2 only if B_1 does not perform any action—either internal or not—before (or at) t, whereas in $B_1 \overset{t}{\blacktriangleright} B_2$ control is passed to B_2 at time t, regardless of the activities of B_1 until time t (with the exception of termination).

The synchronization principle for PA_R is identical to that in PA_T and PA_U: an action can only occur when all participants are ready to engage in it. Thus, in behaviour

$$a\,;(T_1)\ b\,;\mathbf{0}\,||_{\{a,b\}}\,a\,;(T_2)\ b\,;\mathbf{0}$$

b is enabled at any time in $t_a{+}T_1 \cap t_a{+}T_2 = t_a + (T_1 \cap T_2)$.

Notice that by means of synchronization actions may become impossible due to incompatible timing constraints in the participating behaviours. For instance, if $T_1 \cap T_2 = \varnothing$ in the example just above, b can never occur.

7.3.2 Causality-based semantics

In this section we show how a causality-based semantics can be given to PA_R using real-time event structures. We define a mapping $\mathcal{E}_R[\![\ ]\!] : \mathsf{PA}_R \longrightarrow \mathsf{EBES}_R$. For convenience we use the denotational semantics $\mathcal{E}'[\![\ ]\!]$ for the untimed case which is defined in Chapter 2. In addition we use:

7.17. DEFINITION. $\Phi_R : \mathsf{PA}_R \longrightarrow \mathsf{PA}$ is defined as follows:

$$\begin{array}{rcl}
\Phi_R(\mathbf{0}) & \triangleq & \mathbf{0} \\
\Phi_R(\surd) & \triangleq & \surd \\
\Phi_R((T)\ a\,;\,B) & \triangleq & a\,;\,\Phi_R(B) \\
\Phi_R(B_1\ \mathsf{op}\ B_2) & \triangleq & \Phi_R(B_1)\ \mathsf{op}\ \Phi_R(B_2) \text{ for } \mathsf{op}\ \in \{\,+,\ ||_G\,,>>,\ [>\,\} \\
\Phi_R(\mathsf{op}\ B) & \triangleq & \mathsf{op}\ \Phi_R(B) \text{ for } \mathsf{op}\ \in \{\,\backslash, [\,]\,\} \\
\Phi_R(B_1 \overset{t}{\rhd} B_2) & \triangleq & \Phi_R(B_1) + \tau\,;\,\Phi_R(B_2) \\
\Phi_R(B_1 \overset{t}{\blacktriangleright} B_2) & \triangleq & \Phi_R(B_1)\,[>\Phi_R(B_2).
\end{array}$$

□

So, Φ_R associates to a real-time behaviour B its corresponding untimed behaviour $\Phi_R(B)$ by simply omitting all time annotations in B and converting $\rhd$ and $\blacktriangleright$ into $+$ and $[>$, respectively. The purpose of the internal event introduced by the timeout operator will be explained later on.

In the rest of this section let $\mathcal{E}_R[\![\,B_i\,]\!] = \Lambda_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$, for i=1, 2, with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ and $E_1 \cap E_2 = \varnothing$. The functions *init* and *exit* which denote the set of initial and termination events, respectively, are defined for event structures in Chapter 2 and are used for real-time event structures in the same way. Let E_U denote the universe of events.

7.18. DEFINITION. (*Real-time semantics of* **0**, $\surd$, *and* $(T)\ a\,;$)

$$\begin{array}{rcl}
\mathcal{E}_R[\![\,\mathbf{0}\,]\!] & \triangleq & \langle \mathcal{E}'[\![\,\Phi_R(\mathbf{0})\,]\!], \varnothing, \varnothing, \varnothing \rangle \\
\mathcal{E}_R[\![\,\surd\,]\!] & \triangleq & \langle \mathcal{E}'[\![\,\Phi_R(\surd)\,]\!], \{\,(e_\delta, \mathsf{Time}^\infty)\,\}, \varnothing, \{\,(e_\delta, \text{false})\,\} \rangle \\
\mathcal{E}_R[\![\,(T)\ a\,;\,B_1\,]\!] & \triangleq & \langle (E, \rightsquigarrow_1, \mapsto, l_1 \cup \{\,(e_a, a)\,\}), \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle \text{ where} \\
E & = & E_1 \cup \{\,e_a\,\} \text{ for some } e_a \in E_U \setminus E_1 \\
\mapsto & = & \mapsto_1\ \cup (\{\,\{\,e_a\,\}\,\} \times E_1) \\
\mathcal{D} & = & \{\,(e_a, T)\,\} \cup (E_1 \times \{\,\mathsf{Time}^\infty\,\}) \\
\mathcal{T} & = & \mathcal{T}_1 \cup \{\,((\{\,e_a\,\}, e), \mathcal{D}_1(e)) \mid e \in E_1\,\} \\
\mathcal{U} & = & \mathcal{U}_1 \cup \{\,(e_a, \text{false})\,\}.
\end{array}$$

□

The semantics of $\mathbf{0}$ and $\surd$ is self-explanatory. For timed action-prefix the semantics closely resembles that for the simple timed case, i.e., PA_T, except that now bundles are introduced between the new event e_a and *all* events in $\mathcal{E}_R[\![B_1]\!]$. In this way it is guaranteed that the resulting structure is indeed a real-time event structure: it satisfies the first constraint on urgent events of Definition 7.2. A similar approach is taken for >>, see just below. For the other operators the semantics is a straightforward generalization of the denotational semantics for the simple timed case.

7.19. DEFINITION. (*Real-time semantics of* $\backslash$, $[\,]$, $+$, $>>$ *and* $[>$)

$$
\begin{array}{rcl}
\mathcal{E}_R[\![B_1 + B_2]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi_R(B_1 + B_2)]\!], \mathcal{D}_1 \cup \mathcal{D}_2, \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle \\
\mathcal{E}_R[\![B_1 \,[> B_2]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi_R(B_1 \,[> B_2)]\!], \mathcal{D}_1 \cup \mathcal{D}_2, \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle \\
\mathcal{E}_R[\![\mathsf{op}\, B_1]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi_R(\mathsf{op}\, B_1)]\!], \mathcal{D}_1, \mathcal{T}_1, \mathcal{U}_1 \rangle \text{ for } \mathsf{op} \in \{ \backslash, [\,] \} \\
\mathcal{E}_R[\![B_1 >> B_2]\!] & \triangleq & \langle (E_1 \cup E_2, \rightsquigarrow, \mapsto, l), \mathcal{D}, \mathcal{T}, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle \text{ where} \\
\rightsquigarrow & = & \rightsquigarrow_1 \,\cup\, \rightsquigarrow_2 \,\cup \{ (e, e') \mid e, e' \in exit(\Lambda_1) \,\wedge\, e \neq e' \} \\
\mapsto & = & \mapsto_1 \,\cup\, \mapsto_2 \,\cup (\{ exit(\Lambda_1) \} \times E_2) \\
l & = & ((l_1 \cup l_2) \setminus (exit(\Lambda_1) \times \{ \delta \})) \cup (exit(\Lambda_1) \times \{ \tau \}) \\
\mathcal{D} & = & \mathcal{D}_1 \cup (E_2 \times \{ \mathsf{Time}^\infty \}) \\
\mathcal{T} & = & \mathcal{T}_1 \cup \mathcal{T}_2 \cup \{ ((exit(\Lambda_1), e), \mathcal{D}_2(e)) \mid e \in E_2 \}.
\end{array}
$$

□

For parallel composition the set of time instants associated to a bundle equals the intersection of the delays associated to the bundles we get by projecting on the i-th components (i=1, 2) of the events in the bundle, if this projection yields a bundle in $\mathcal{E}_R[\![B_i]\!]$. The delay of an event is the intersection of the delays of its components that are different from $*$. Finally, an event is urgent once one of its components is urgent (where $*$ is treated as nonurgent).

7.20. DEFINITION. (*Real-time semantics of* $||_G$)

$$
\begin{array}{rcl}
\mathcal{E}_R[\![B_1 \,||_G\, B_2]\!] & \triangleq & \langle \mathcal{E}'[\![\Phi_R(B_1 \,||_G\, B_2)]\!], \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle \text{ where} \\
\mathcal{D}((e_1, e_2)) & = & \mathcal{D}_1(e_1) \,\cap\, \mathcal{D}_2(e_2) \text{ with } \mathcal{D}_i(*) = \mathsf{Time}^\infty \\
\mathcal{T}((X, (e_1, e_2))) & = & \mathcal{T}_1((pr_1(X), e_1)) \,\cap\, \mathcal{T}_2((pr_2(X), e_2)) \\
 & & \quad \text{with } \mathcal{T}_i((\varnothing, e_i)) = \mathsf{Time}^\infty, \text{ for } i{=}1,2 \\
\mathcal{U}((e_1, e_2)) & = & \mathcal{U}_1(e_1) \,\vee\, \mathcal{U}_2(e_2) \text{ with } \mathcal{U}_i(*) = \text{false}, \text{for } i{=}1,2.
\end{array}
$$

□

Now we consider the denotational semantics for the two new operators $\rhd$ and $\blacktriangleright$. We start with the timeout operator.

In $\mathcal{E}_R[\![B_1 \overset{t}{\rhd} B_2]\!]$ a new internal, urgent event e_τ is introduced that models the expiration of the timer (i.e., e_τ models a timeout). Since either the timer expires or B_1 performs an initial action before (or at) t, event e_τ is put in mutual conflict with all initial events of $\mathcal{E}_R[\![B_1]\!]$. The events of $\mathcal{E}_R[\![B_2]\!]$ can only occur after the timeout; this is modelled in the same way as for action-prefix: a bundle $\{ e_\tau \} \mapsto e$ is introduced for all $e \in \mathcal{E}_R[\![B_2]\!]$. (Again bundles are introduced to *all* events in $\mathcal{E}_R[\![B_2]\!]$ in order to guarantee that this yields a real-time event structure.) The delay of these bundles is determined as in the action-prefix case. The event delay of e_τ becomes $[t, t]$ such that it can only occur at t time units since the enabling of $\mathcal{E}_R[\![B_1 \overset{t}{\rhd} B_2]\!]$. So, $\mathcal{E}_R[\![B_1 \overset{t}{\rhd} B_2]\!]$ equals $\mathcal{E}_R[\![B_1 + ([t,t])\tau \,;\, B_2]\!]$ where τ is urgent.

7.21. DEFINITION. (*Real-time semantics of* $\rhd$)

$$\begin{array}{rcl}
\mathcal{E}_R[\![B_1 \overset{t}{\rhd} B_2]\!] & \triangleq & \langle (E, \rightsquigarrow, \mapsto, l), \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle \text{ where} \\
E & = & E_1 \cup E_2 \cup \{ e_\tau \} \text{ for some } e_\tau \in E_U \setminus (E_1 \cup E_2) \\
\rightsquigarrow & = & \rightsquigarrow_1 \;\cup\; \rightsquigarrow_2 \;\cup (\mathit{init}(\Lambda_1) \times \{ e_\tau \}) \cup (\{ e_\tau \} \times \mathit{init}(\Lambda_1)) \\
\mapsto & = & \mapsto_1 \;\cup\; \mapsto_2 \;\cup (\{ \{ e_\tau \} \} \times E_2) \\
l & = & l_1 \cup l_2 \cup \{ (e_\tau, \tau) \} \\
\mathcal{D} & = & \mathcal{D}_1 \cup \{ (e_\tau, [t,t]) \} \cup (E_2 \times \{ \mathsf{Time}^\infty \}) \\
\mathcal{T} & = & \mathcal{T}_1 \cup \mathcal{T}_2 \cup \{ ((\{ e_\tau \}, e), \mathcal{D}_2(e)) \mid e \in E_2 \} \\
\mathcal{U} & = & \mathcal{U}_1 \cup \mathcal{U}_2 \cup \{ (e_\tau, \text{true}) \}.
\end{array}$$

□

7.22. EXAMPLE. Figure 7.3 shows how $\Lambda_1 \overset{12}{\rhd} \Lambda_2$ is constructed from Λ_1 and Λ_2.

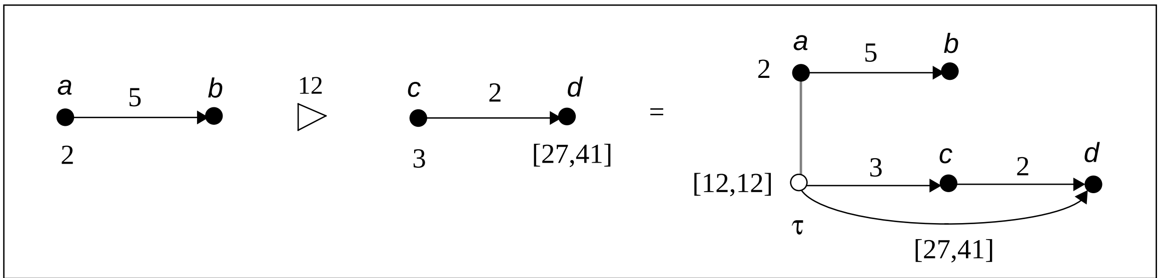


Figure 7.3: Example of timed semantics for timeout operator (I).

As a second example consider $B_1 = ([3,7])\ a\,;\,(2)\ b\,;\,\mathbf{0} \,|||\, (6)\ c\,;\,\mathbf{0}$ and $B_2 = ([4,32))\ d\,;\,\mathbf{0}$. Figure 7.4 illustrates how $\mathcal{E}_R[\![B_1 \overset{6}{\rhd} B_2]\!]$ is constructed from $\mathcal{E}_R[\![B_1]\!]$ and $\mathcal{E}_R[\![B_2]\!]$. □

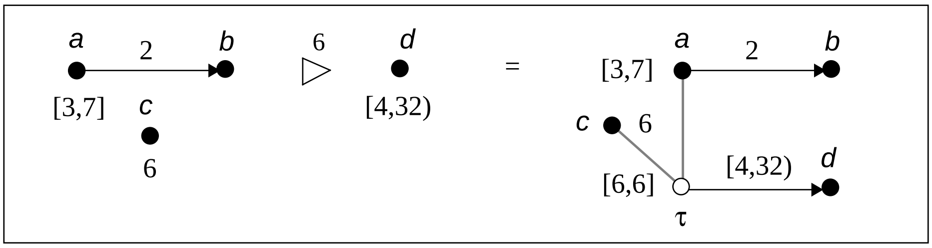
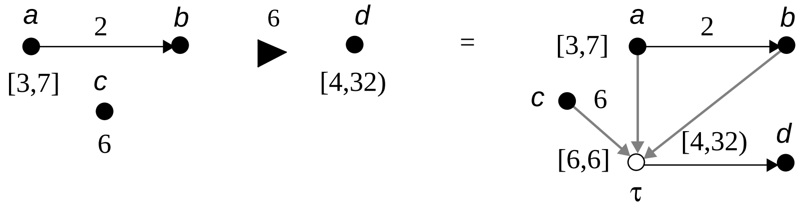


Figure 7.4: Example of timed semantics for timeout operator (II).

A similar approach could be taken for the watchdog operator: for $\mathcal{E}_R[\![B_1 \overset{t}{\blacktriangleright} B_2]\!]$ introduce a new urgent event e with delay $[t,t]$, let this event precede all events in $\mathcal{E}_R[\![B_2]\!]$, and introduce a conflict $e' \rightsquigarrow e$ for all events e in $\mathcal{E}_R[\![B_1]\!]$ such that at time t it is guaranteed that B_1 is interrupted; for the other attributes do the same as for B_1 [> τ_e ; B_2.

This recipe would, for example, result for $B_1 \overset{6}{\blacktriangleright} B_2$, where B_1 and B_2 are taken from Example 7.22, in:



There is, however, also a possibility to model $B_1 \overset{t}{\blacktriangleright} B_2$ in a simpler way without using urgent events. Consider $\mathcal{E}_R[\![B_1 [> B_2]\!]$, i.e., the real-time event structure of B_1 [> B_2, and (i) restrict all event delays in $\mathcal{E}_R[\![B_1]\!]$ by $[0, t]$ ensuring that these events can only occur at time t at the latest, and (ii) postpone all events in $\mathcal{E}_R[\![B_2]\!]$ by time t such that these events can only occur from t on.

7.23. Definition. (*Real-time semantics of* $\blacktriangleright$)

$$\begin{array}{rcl} \mathcal{E}_R[\![B_1 \overset{t}{\blacktriangleright} B_2]\!] & \triangleq & \mathcal{E}'[\![\Phi_R(B_1 \overset{t}{\blacktriangleright} B_2)]\!], \mathcal{D}, \mathcal{T}_1 \cup \mathcal{T}_2, \mathcal{U}_1 \cup \mathcal{U}_2 \rangle \text{ where} \\ \mathcal{D} & = & \{ (e, \mathcal{D}_1(e) \cap [0,t]) \mid e \in E_1 \} \cup \{ (e, t+\mathcal{D}_2(e)) \mid e \in E_2 \}. \end{array}$$

□

7.24. Example. Figure 7.5 shows how $\Lambda_1 \overset{6}{\blacktriangleright} \Lambda_2$ is constructed from Λ_1 and Λ_2. The reader is invited to compare this figure with Figure 7.4. □

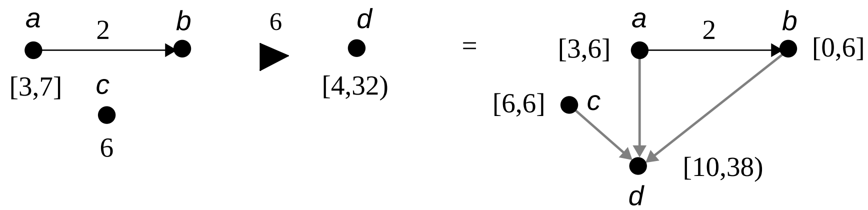


Figure 7.5: Example of timed semantics for watchdog operator.

7.3.3 Properties

The results in this section are all relative to $\Lambda = \mathcal{E}_R[\![B]\!] = \langle (E, \rightsquigarrow, \mapsto, l), \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ for $B \in \mathsf{PA}_R$.

7.25. Lemma. $\forall\, e \in E : \mathcal{U}(e) \;\Rightarrow\; l(e) = \tau$.

Proof. Straightforward, since urgent events are only introduced for $\rhd$, and the urgency of events is unaffected by $\mathcal{E}_R[\![\ ]\!]$ for all other syntactical constructs in PA_R. □

7.26. LEMMA. $\forall\, e, e' \in E, X \subseteq E : (\mathcal{U}(e) \;\wedge\; e' \rightsquigarrow e \;\wedge\; X \mapsto e) \Rightarrow X \mapsto e'$.

PROOF. By induction on the structure of B. Let $B \in \mathsf{PA}_R$ and $\Lambda_i = \mathcal{E}_R[\![\, B_i \,]\!] = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$ where $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2.
Base: For $B = \mathbf{0}$ and $B = \surd$ the lemma holds, since Λ does not contain any urgent event.
Induction Step: Assume the theorem holds for B_1 and B_2.

1. $B = (T)\ a\, ;\, B_1$. The new event e_a is not urgent and is not put in conflict with some urgent event, so it suffices to consider urgent events in Λ_1. Let $e \in E_1$ with $\mathcal{U}(e)$ (i.e., $\mathcal{U}_1(e)$), and $e' \in E_1$ such that $e' \rightsquigarrow e$ (i.e., $e' \rightsquigarrow_1 e$). Let $X \mapsto e$. If $X \mapsto_1 e$ then—by the induction hypothesis—we have $X \mapsto_1 e'$, and so $X \mapsto e'$. In case $X \mapsto e$ is a new bundle, then $X = \{\, e_a \,\}$ and it follows from $\mathcal{E}_R[\![\ ]\!]$ that also $\{\, e_a \,\} \mapsto e'$, since a bundle is introduced from e_a to all events in E_1.

2. $B = B_1 + B_2$. For non-initial events in Λ_1 and Λ_2 the lemma follows directly from the induction hypothesis, since these events are unaffected in Λ. *init*(Λ_1) and *init*(Λ_2) are put in mutual conflict, but since there is no bundle pointing to these events, the lemma follows directly.

3. $B = B_1 >> B_2$. The events in *exit*(Λ_1) are put in mutual conflict, but since these events are nonurgent (cf. Lemma 7.25) this does not violate the lemma. In addition, new bundles from *exit*(Λ_1) to all events in E_2 are introduced. It follows in the same way as for action-prefix that these bundles do not harm the lemma: if a bundle is introduced to an urgent event e in E_2 then the same bundle is introduced to all events that are disabled by e in E_2.

4. $B = B_1\, [> B_2$. The new conflicts between *init*(Λ_2) and *exit*(Λ_1) do not affect the lemma since all events in *exit*(Λ_1) are nonurgent (cf. Lemma 7.25). The other new conflicts are between E_1 and *init*(Λ_2). Suppose there is some $e \in$ *init*(Λ_2) with $\mathcal{U}_2(e)$. Since e is an initial event, no bundles are pointing to e and the lemma holds immediately. Since all other events in Λ are unaffected this proves the case.

5. $B = B_1 \setminus G$. For this case the lemma directly follows from the induction hypothesis. The same applies to relabelling.

6. $B = B_1 \,||_G\, B_2$. Suppose $e = (e_1, e_2) \in E$ such that $\mathcal{U}(e)$, and $e' = (e_1', e_2') \in E$ with $e' \rightsquigarrow e$. Since urgent events are internal (cf. Lemma 7.25), no synchronization of urgent events takes place; i.e., $e_1 = *$ and $e_2 \neq *$, or the reverse. By symmetry it suffices to consider $e_1 = *$ and $e_2 \neq *$. But then $e' \rightsquigarrow e$ implies $e_2' \rightsquigarrow_2 e_2$. Suppose $X \mapsto e$. Since $e = (*, e_2)$ it follows that $X = \{\, (e, e') \in E \mid e' \in X_2 \,\}$ where $X_2 \mapsto_2 e_2$. By induction hypothesis it follows that $X_2 \mapsto_2 e_2'$, and so, $X \mapsto e'$.

7. $B = B_1 \overset{t}{\rhd} B_2$. Similar to the proof for $+$ since the untimed event structure corresponding to B equals $\mathcal{E}'[\![\, B_1 + \tau_\xi\, ;\, B_2 \,]\!]$, where ξ is an urgent (timeout) event.

8. $B = B_1 \overset{t}{\blacktriangleright} B_2$. Similar to the proof for $B_1\, [> B_2$ since the untimed event structure corresponding to B equals $\mathcal{E}'[\![\, B_1\, [> B_2 \,]\!]$.

□

7.27. LEMMA. $\forall e, e' \in E, X \subseteq E : (\mathcal{U}(e) \ \wedge \ e \rightsquigarrow e' \ \wedge \ X \mapsto e) \Rightarrow (X \mapsto e' \vee X \rightsquigarrow e')$.

PROOF. By induction on the structure of B. Let $B \in \mathsf{PA}_R$ and $\Lambda_i = \mathcal{E}_R[\![B_i]\!] = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$ where $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2.
Base: For $B = \mathbf{0}$ and $B = \surd$ the lemma holds, since Λ does not contain any urgent event.
Induction Step: Assume the theorem holds for B_1 and B_2. We only consider the proofs for disrupt and parallel composition. For all other constructs the proof is very similar to the proof of Lemma 7.26.

1. $B = B_1 [> B_2$. We have $E = E_1 \cup E_2$ and $\mathcal{U} = \mathcal{U}_1 \cup \mathcal{U}_2$. Let $e \in E$.
   (a) Let $e \in E_1$ and suppose $\mathcal{U}_1(e)$. For $e \rightsquigarrow e'$ with $e' \in E_1$ we have $e \rightsquigarrow_1 e'$ and the lemma follows from the induction hypothesis (and the fact that bundles and conflicts in Λ_1 are retained in Λ). Let $e \rightsquigarrow e'$ but not $e \rightsquigarrow_1 e'$. Then we have $e' \in \mathit{init}(\Lambda_2)$. Suppose $X \mapsto e$. It follows from the definition of $\mathcal{E}'[\![\ ]\!]$ that then $X \mapsto_1 e$, so $X \subseteq E_1$. Since new conflicts are introduced between E_1 and $\mathit{init}(\Lambda_2)$ we have $(\forall e'' \in X : e'' \rightsquigarrow e')$, i.e., $X \rightsquigarrow e'$.
   (b) Let $e \in E_2$ and suppose $\mathcal{U}_2(e)$. If $e \notin \mathit{init}(\Lambda_2)$ neither new conflicts nor new bundles are introduced; for this case the lemma follows directly from the induction hypothesis. Assume $e \in \mathit{init}(\Lambda_2)$. Since there are no bundles pointing to e the lemma holds trivially.
2. $B = B_1 \,||_G\, B_2$. Suppose $e = (e_1, e_2) \in E$ and $e' = (e_1', e_2') \in E$ such that $\mathcal{U}(e)$ and $e \rightsquigarrow e'$. Assume $e_1 = *$ and $e_2 \neq *$. Then we have $e_2 \rightsquigarrow_2 e_2'$. Suppose $X \mapsto e$. Since $e = (*, e_2)$ we have $pr_1(X) = \varnothing$ and $pr_2(X) = X_2$ such that $X_2 \mapsto_2 e_2$. By induction hypothesis it follows $X_2 \mapsto_2 e_2' \ \vee \ X_2 \rightsquigarrow_2 e_2'$. But then we have, according to the definition of $\mathcal{E}'[\![\ ]\!]$, that $X \mapsto e'$ or $X \rightsquigarrow e'$. The proof for the case that $e_1 \neq *$ and $e_2 = *$ is obtained by exchanging the subscripts in the above proof.

□

7.28. LEMMA. For all $e \in E$ such that $\mathcal{U}(e)$ we have:

$$\exists\, t \in \mathsf{Time} : \mathcal{D}(e) \subseteq [t, t] \ \vee \ (\exists\, X \subseteq E : X \overset{T}{\mapsto} e \ \wedge \ T \subseteq [t, t]) \quad .$$

PROOF. By induction on the structure of B.
Base: For $B = \mathbf{0}$ and $B = \surd$ the lemma holds since Λ contains no urgent events.
Induction Step: Assume the lemma holds for B_1 and B_2. We provide the proof for action-prefix, choice, parallel composition, timeout, and watchdog. For the other operators the proof is conducted in a similar way.

1. $B = (T)\ a\,; B_1$. Suppose that $e \in E$ such that $\mathcal{U}(e)$. Then $e \in E_1$ and $\mathcal{U}_1(e)$, since e_a is nonurgent. If $X \overset{T'}{\mapsto}_1 e$ with $T' \subseteq [t, t]$ then this bundle remains in Λ with the same timing and so for this case the lemma holds. Now suppose $\mathcal{D}_1(e) \subseteq [t, t]$. The new bundle $\{\, e_a \,\} \mapsto e$ will become delay $\mathcal{D}_1(e)$, and so also in this case the lemma holds.
2. $B = B_1 + B_2$. For this case the lemma directly follows from the induction hypothesis.
3. $B = B_1 \,||_G\, B_2$. Let $e = (e_1, e_2) \in E$ such that $\mathcal{U}(e)$. Since no synchronizations on urgent events takes place we have $e_1 = * \ \wedge \ e_2 \neq *$, or the reverse. By symmetry, it suffices to consider $e_1 = * \ \wedge \ e_2 \neq *$. By the induction hypothesis we have that $\mathcal{D}_2(e_2) \subseteq [t, t]$ or that $X_2 \overset{T}{\mapsto}_2 e_2$ with $T \subseteq [t, t]$, for some t.

(a) Suppose $\mathcal{D}_2(e_2) \subseteq [t,t]$. From the definition of $\mathcal{E}_R[\![\ ]\!]$ it follows that $\mathcal{D}(e) = \mathcal{D}_1(e_1) \cap \mathcal{D}_2(e_2)$ which equals $\mathcal{D}_2(e_2)$, since $e_1 = *$ and $\mathcal{D}_1(*) = \mathsf{Time}^\infty$. So, then $\mathcal{D}(e) \subseteq [t,t]$.

(b) Suppose $X_2 \stackrel{T}{\mapsto}_2 e_2$ with $T \subseteq [t,t]$. Since $e_1 = *$ this means that $X \mapsto e$ with $pr_1(X) = \varnothing$ and $pr_2(X) = X_2$. According to the definition of $\mathcal{E}_R[\![\ ]\!]$ we have that $\mathcal{T}((X,e))$ equals $\mathcal{T}_1((pr_1(X),e_1)) \cap \mathcal{T}_2((pr_2(X),e_2))$ which equals (since $\mathcal{T}_1((\varnothing,e_1)) = \mathsf{Time}^\infty$) $\mathcal{T}_2((X_2,e_2)) = T$. So, $X \stackrel{T}{\mapsto} e$ with $T \subseteq [t,t]$.

4. $B = B_1 \stackrel{t}{\rhd} B_2$. Let $e \in E$ and suppose $\mathcal{U}(e)$. There are three different cases to be considered.

   (a) $e \in E_1$ and $\mathcal{U}_1(e)$. Since the delay of e and the bundle delays of bundles in Λ_1 are unaffected the lemma holds for this case by the induction hypothesis.

   (b) $e \in E_2$ and $\mathcal{U}_2(e)$. Here, the same arguments as for action-prefix apply; if $X \stackrel{T}{\mapsto}_2 e$ with $T \subseteq [t',t']$ for some t' then this bundle remains in Λ and so for this case the lemma holds, and in case $\mathcal{D}_2(e) \subseteq [t',t']$ a new bundle $\{ e_\tau \} \mapsto e$ is introduced and becomes delay $\mathcal{D}_2(e)$. So, the lemma also holds for this case.

   (c) $e = e_\tau$. For the new urgent event e_τ we have $\mathcal{D}(e_\tau) = [t,t]$.

5. $B = B_1 \stackrel{t}{\blacktriangleright} B_2$. Let $e \in E$ with $\mathcal{U}(e)$. Event and bundle delays in Λ_2 are unaffected, so for $e \in E_2$ the lemma follows from the induction hypothesis. Let $e \in E_1$. If $e \notin \mathit{init}(\Lambda_1)$ we have $\mathcal{D}(e) = \mathcal{D}_1(e)$ which, together with the fact that bundle delays in Λ_1 are unaffected, proves the case. If $e \in \mathit{init}(\Lambda_1)$ then $\mathcal{D}_1(e) \subseteq [t',t']$ by the induction hypothesis. But, since $\mathcal{D}(e) = \mathcal{D}_1(e) \cap [0,t]$, it also follows $\mathcal{D}(e) \subseteq [t',t']$.

□

7.29. THEOREM. $\forall\, B \in \mathsf{PA}_R : \mathcal{E}_R[\![\, B\,]\!] \in \mathsf{EBES}_R$.

PROOF. Let $B \in \mathsf{PA}_R$ and $\Lambda = \mathcal{E}_R[\![\, B\,]\!] = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$. It follows directly from the definition of $\mathcal{E}_R[\![\ ]\!]$ that $\mathcal{E} \in \mathsf{EBES}$ and that $\mathcal{D}$, $\mathcal{T}$, and $\mathcal{U}$ are total functions. From Lemma 7.26, Lemma 7.27, and Lemma 7.28 it follows that $\mathcal{E}_R[\![\, B\,]\!]$ satisfies the constraints of being a real-time event structure (cf. Definition 7.2). □

Notice that we do not have a (strong) backward compatibility result like Theorem 4.36, for two reasons: due to empty sets of time instants (e.g., due to synchronization) and the presence of urgent events, events may be permanently disabled in the timed sense, but not from a causality point of view. For example, $B = (\varnothing)\ a\,;\mathbf{0}$ has only an empty lposet, whereas $\Phi_R(B) = a\,;\mathbf{0}$ has an lposet in which an event labelled a occurs.

7.3.4 Event-based operational semantics for PA_R

This section defines a timed event transition system for PA_R. This is performed along the same lines as in Chapter 5. The differences with PA_T are (i) that a set of time instants is associated to an action in a timed action-prefix; (ii) the inclusion of a timeout and (iii) a watchdog operator. Besides the fact that—as in Chapter 5—all occurrences of action-prefix and successful termination are uniquely identified (by a Greek letter) we do the same for all occurrences of $\triangleright$. E.g., ξ in $B_1 \triangleright_\xi B_2$ represents the event identifier of the urgent event that models the timeout.

As a subsidiary notion let $\mathsf{ut}(B)$ denote the set of time instants at which B can initially perform an urgent event. Let PA_R^+ equal PA_R including the auxiliary ${}^t[\,]$ and ${}^t\{\,\}$ operators.

7.30. Definition. $\mathsf{ut} : \mathsf{PA}_R^+ \longrightarrow \mathcal{P}(\mathsf{Time})$ is defined by:

$$
\begin{array}{rcl}
\mathsf{ut}({}^t[\,B\,]) & \triangleq & \{\, t'{+}t \mid t' \in \mathsf{ut}(B)\,\} \\
\mathsf{ut}(B_1 \;\mathsf{op}\; B_2) & \triangleq & \mathsf{ut}(B_1) \cup \mathsf{ut}(B_2) \text{ for } \mathsf{op} \in \{\,+,\, [>,\, ||_G\,\} \\
\mathsf{ut}({}^t\{\,B\,\}) & \triangleq & \{\, t' \in \mathsf{ut}(B) \mid t' \geqslant t\,\} \\
\mathsf{ut}(B_1 >> B_2) & \triangleq & \mathsf{ut}(B_1) \\
\mathsf{ut}(\mathsf{op}\; B) & \triangleq & \mathsf{ut}(B) \text{ for } \mathsf{op} \in \{\,\backslash, [\,]\,\} \\
\mathsf{ut}(B_1 \overset{t}{\triangleright} B_2) & \triangleq & \mathsf{ut}(B_1) \cup \{\,t\,\} \\
\mathsf{ut}(B_1 \overset{t}{\blacktriangleright} B_2) & \triangleq & \mathsf{ut}(B_1) \cup \mathsf{ut}({}^t[\,B_2\,]).
\end{array}
$$

For all other syntactical constructs let $\mathsf{ut}(B) \triangleq \varnothing$. □

Let $\mathsf{mt}(B)$ abbreviate $\mathit{Min}(\mathsf{ut}(B))$, where Min of the empty set equals 0. We will later on prove the correctness of mt.

Table 7.1 presents the event-based inference rules for PA_R. For various operators the inference rules are identical to the rules for PA_T, see Table 5.1. We only discuss the inference rules that have been modified or introduced.

$(T)\ a_\xi\,;B$ can perform ξ at any time $t \in T$, while evolving into ${}^t[\,B\,]$.

The rules for $B_1 + B_2$ are somewhat adapted since (initial) urgent events in B_1 or B_2 can decide the choice. E.g., in

$$(12)\ a_\xi\,;\mathbf{0} + ((18)\ b_\psi\,;\mathbf{0} \overset{5}{\triangleright}_\chi \mathbf{0})$$

the event χ will occur at time 5, and resolve the choice in favour of B_2. In general, if B_1 performs an event at time t then $B_1 + B_2$ can perform the same provided that B_2 cannot perform an urgent event at any time earlier, i.e., if $t \leqslant \mathsf{mt}(B_2)$. By symmetry, a similar condition is obtained for B_2 performing an event. The inference rules for $[>$ are adjusted analogously.

If B_1 performs an event at time t', with $t' \leqslant t$, and evolves into B_1' then $B_1 \overset{t}{\triangleright}_\psi B_2$ can do the same; in this case the possibility that B_2 happens is dropped since B_1 has performed an action before (or at) time t. At time t the timeout event ψ can happen and the resulting behaviour is ${}^t[\,B_2\,]$, B_2 shifted t time units in advance. This can only be done if $t \leqslant \mathsf{mt}(B_1)$. This condition ensures that ψ is not performed if B_1 can perform an urgent event before t. E.g., in $(a\,;\mathbf{0} \overset{7}{\triangleright}_\xi \mathbf{0}) \overset{21}{\triangleright}_\psi \mathbf{0}$ it prevents ψ from happening (at time 21) without ξ being executed (at time 7).

If B_1 performs an event (which is not a successful termination event) at time t', with $t' \leqslant t$, and evolves into B_1' then $B_1 \overset{t}{\blacktriangleright} B_2$ can do the same while evolving into $B_1' \overset{t}{\blacktriangleright} B_2$; the possibility for disruption (at time t) by B_2 remains. If B_1 terminates successfully at time t', $t' \leqslant t$, disruption by B_2 becomes impossible (like for $B_1 [> B_2$). If B_2 performs an event at time t' and evolves into B_2' then $B_1 \overset{t}{\blacktriangleright} B_2$ can perform the same (provided B_1 cannot perform an urgent event earlier) and evolves into ${}^{t}[\, B_2' \,]$, B_2' shifted t time units in time.

7.31. EXAMPLE. Consider

$$B := \Big(((([3,7])\ a_\xi\,;\, \surd_\psi \,|||\, (14)\ b_\chi\,;\, \surd_\eta) >> ([1,12))\ c_\rho\,;\, \mathbf{0}\Big) \overset{17}{\blacktriangleright} ((1)\ d_\mu\,;\, \mathbf{0} \,|||\, ([3,\pi))\ f_\nu\,;\, \mathbf{0}).$$

Using the inference rules of Table 7.1 we derive

$$\begin{array}{l}
\quad \Big(((([3,7])\ a_\xi\,;\, \surd_\psi \,|||\, (14)\ b_\chi\,;\, \surd_\eta) >> ([1,12))\ c_\rho\,;\, \mathbf{0}\Big) \overset{17}{\blacktriangleright} ((1)\ d_\mu\,;\, \mathbf{0} \,|||\, ([3,\pi))\ f_\nu\,;\, \mathbf{0}) \\
\xrightarrow{((*,\chi),b,17)}\!\!\!\!\rightarrow \{\ \text{(timed action-prefix), (par-right), (enabling-left), (watchdog-left)}\ \} \\
\quad \Big(((([3,7])\ a_\xi\,;\, \surd_\psi \,|||\, {}^{17}[\, \surd_\eta \,]) >> ([1,12))\ c_\rho\,;\, \mathbf{0}\Big) \overset{17}{\blacktriangleright} ((1)\ d_\mu\,;\, \mathbf{0} \,|||\, ([3,\pi))\ f_\nu\,;\, \mathbf{0}) \\
\xrightarrow{((\xi,*),a,5)}\!\!\!\!\rightarrow \{\ \text{(timed action-prefix), (par-left), (enabling-left), (watchdog-left)}\ \} \\
\quad \Big(({}^{5}[\, \surd_\psi \,] \,|||\, {}^{17}[\, \surd_\eta \,]) >> ([1,12))\ c_\rho\,;\, \mathbf{0}\Big) \overset{17}{\blacktriangleright} ((1)\ d_\mu\,;\, \mathbf{0} \,|||\, ([3,\pi))\ f_\nu\,;\, \mathbf{0}) \\
\xrightarrow{(\nu,f,20)}\!\!\!\!\rightarrow \{\ \text{(timed action-prefix), (par-right), (watchdog-right)}\ \} \\
\quad {}^{17}[\, ((1)\ d_\mu\,;\, \mathbf{0} \,|||\, {}^{3}[\, \mathbf{0} \,]\,] \quad .
\end{array}$$

□

In order to define and prove the correctness of the mt function we let $\mathsf{UE}(B)$ denote the set of urgent events in B.

7.32. DEFINITION. Function $\mathsf{UE} : \mathsf{PA}_R^+ \longrightarrow \mathcal{P}(Ev)$ is defined as

$$\begin{array}{rcl}
\mathsf{UE}(B) & \triangleq & \varnothing \text{ for } B \in \{\, \mathbf{0}, \surd_\xi \,\} \\
\mathsf{UE}(\mathsf{op}\, B) & \triangleq & \mathsf{UE}(B) \text{ for } \mathsf{op} \in \{\, (T)\ a_\xi\,;\, , \backslash, [\,], {}^{t}[\,], {}^{t}\{\,\} \,\} \\
\mathsf{UE}(B_1\, \mathsf{op}\, B_2) & \triangleq & \mathsf{UE}(B_1) \cup \mathsf{UE}(B_2) \text{ for } \mathsf{op} \in \{\, +, >>, [>, \blacktriangleright \,\} \\
\mathsf{UE}(B_1 \,||_G\, B_2) & \triangleq & \{\, (e,*) \mid e \in \mathsf{UE}(B_1) \,\} \cup \{\, (*,e) \mid e \in \mathsf{UE}(B_2) \,\} \\
\mathsf{UE}(B_1 \overset{t}{\rhd}_\xi B_2) & \triangleq & \mathsf{UE}(B_1) \cup \mathsf{UE}(B_2) \cup \{\, \xi \,\}.
\end{array}$$

□

It is quite straightforward to prove by induction on the structure of B that $\mathsf{UE}(B)$ concurs with our intuition, i.e., if $\mathcal{E}_R[\![\, B \,]\!] = \langle (E, \rightsquigarrow, \mapsto, l), \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$ then we have $\mathsf{UE}(B) = \{\, e \in E \mid \mathcal{U}(e) \,\}$. The proof of this fact is left to the diligent reader.

The following lemma shows that $\mathsf{mt}(B)$ indeed corresponds to the minimal time at which B can perform an urgent event initially.

$$\frac{}{\surd_\xi \xrightarrow{(\xi,\delta,t)} \mathbf{0}}$$

$$\frac{}{(T)\, a_\xi \,;\, B \xrightarrow{(\xi,a,t)} {}^{t}[\,B\,]} \quad (t \in T) \qquad \frac{B \xrightarrow{(\xi,a,t)} B'}{{}^{t'}[\,B\,] \xrightarrow{(\xi,a,t+t')} {}^{t'}[\,B'\,]}$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)} B_1'}{B_1 + B_2 \xrightarrow{(\xi,a,t)} B_1'} \quad (t \leqslant \mathsf{mt}(B_2)) \qquad \frac{B_2 \xrightarrow{(\xi,a,t)} B_2'}{B_1 + B_2 \xrightarrow{(\xi,a,t)} B_2'} \quad (t \leqslant \mathsf{mt}(B_1))$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)} B_1'}{B_1 >> B_2 \xrightarrow{(\xi,a,t)} B_1' >> B_2} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{(\xi,\delta,t)} B_1'}{B_1 >> B_2 \xrightarrow{(\xi,\tau,t)} {}^{t}[\,B_2\,]}$$

$$\frac{B_2 \xrightarrow{(\xi,a,t)} B_2'}{B_1\, [> B_2 \xrightarrow{(\xi,a,t)} B_2'} \quad (t \leqslant \mathsf{mt}(B_1)) \qquad \frac{B_1 \xrightarrow{(\xi,\delta,t)} B_1'}{B_1\, [> B_2 \xrightarrow{(\xi,\delta,t)} B_1'} \quad (t \leqslant \mathsf{mt}(B_2))$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)} B_1'}{B_1\, [> B_2 \xrightarrow{(\xi,a,t)} B_1'\, [> {}^{t}\{\,B_2\,\}} \quad (a \neq \delta \ \wedge\ t \leqslant \mathsf{mt}(B_2))$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)} B_1'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,*),a,t)} B_1' \,||_G\, B_2} \quad (a \notin G^\delta) \qquad \frac{B_2 \xrightarrow{(\xi,a,t)} B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((*,\xi),a,t)} B_1 \,||_G\, B_2'} \quad (a \notin G^\delta)$$

$$\frac{B_1 \xrightarrow{(\xi,a,t)} B_1' \ \wedge\ B_2 \xrightarrow{(\psi,a,t)} B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,\psi),a,t)} B_1' \,||_G\, B_2'} \quad (a \in G^\delta)$$

$$\frac{B \xrightarrow{(\xi,a,t)} B'}{B \setminus G \xrightarrow{(\xi,a,t)} B' \setminus G} \quad (a \notin G) \qquad \frac{B \xrightarrow{(\xi,a,t)} B'}{B \setminus G \xrightarrow{(\xi,\tau,t)} B' \setminus G} \quad (a \in G)$$

$$\frac{B \xrightarrow{(\xi,a,t)} B'}{B[H] \xrightarrow{(\xi,H(a),t)} B'[H]} \qquad \frac{B \xrightarrow{(\xi,a,t)} B'}{{}^{t'}\{\,B\,\} \xrightarrow{(\xi,a,t)} {}^{t'}\{\,B'\,\}} \quad (t \geqslant t')$$

$$\frac{B_1 \xrightarrow{(\xi,a,t')} B_1'}{B_1 \overset{t}{\rhd}_\psi B_2 \xrightarrow{(\xi,a,t')} B_1'} \quad (t' \leqslant t) \qquad \frac{}{B_1 \overset{t}{\rhd}_\psi B_2 \xrightarrow{(\psi,\tau,t)} {}^{t}[\,B_2\,]} \quad (t \leqslant \mathsf{mt}(B_1))$$

$$\frac{B_1 \xrightarrow{(\xi,\delta,t')} B_1'}{B_1 \overset{t}{\blacktriangleright} B_2 \xrightarrow{(\xi,\delta,t')} B_1'} \quad (t' \leqslant t) \qquad \frac{B_2 \xrightarrow{(\xi,a,t')} B_2'}{B_1 \overset{t}{\blacktriangleright} B_2 \xrightarrow{(\xi,\tau,t+t')} {}^{t}[\,B_2'\,]} \quad (t \leqslant \mathsf{mt}(B_1))$$

$$\frac{B_1 \xrightarrow{(\xi,a,t')} B_1'}{B_1 \overset{t}{\blacktriangleright} B_2 \xrightarrow{(\xi,a,t')} B_1' \overset{t}{\blacktriangleright} B_2} \quad (a \neq \delta \ \wedge\ t' \leqslant t)$$

Table 7.1: Event-based operational semantics for PA_R.

7.33. LEMMA. $\forall B \in \mathsf{PA}_R^+ : (t \leqslant \mathsf{mt}(B)) \iff (\forall e \in \mathsf{UE}(B), t' < t : B \xrightarrow{(e,\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow)$.

PROOF. By induction on the structure of B, with base cases $\mathbf{0}$, $\surd$, and action-prefix.
Base: For $B = \mathbf{0}$, $B = \surd$ and $B = (T)\ a\,;\, B_1$ the lemma trivially holds, since B cannot perform an urgent event initially and $\mathsf{mt}(B)$ equals $Min(\varnothing) = \infty$.
Induction Step: Assume the lemma holds for B_1 and B_2. We consider the proof for timeout and parallel composition; the proofs for the other operators are conducted in a similar way.

1. $B = B_1 \overset{t''}{\rhd}_\psi B_2$. For this case we derive:

$$
\begin{aligned}
& t \leqslant \mathsf{mt}(B_1 \overset{t''}{\rhd}_\psi B_2) \\
\Leftrightarrow\ & \{ \text{ definition } \mathsf{mt} \ \} \\
& t \leqslant Min(\mathsf{ut}(B_1 \overset{t''}{\rhd}_\psi B_2)) \\
\Leftrightarrow\ & \{ \text{ Definition 7.30 } \} \\
& t \leqslant Min(\mathsf{ut}(B_1), t'') \\
\Leftrightarrow\ & \{ \text{ calculus; definition } \mathsf{mt} \ \} \\
& t \leqslant t'' \ \wedge \ t \leqslant \mathsf{mt}(B_1) \\
\Leftrightarrow\ & \{ \text{ SOS-rules for } \rhd\text{; induction hypothesis } \} \\
& (B_1 \overset{t''}{\rhd}_\psi B_2 \xrightarrow{(\psi,\tau,t'')}\!\!\!\!\twoheadrightarrow) \ \wedge \ t \leqslant t'' \ \wedge \ (\forall e \in \mathsf{UE}(B_1), t' < t : B_1 \xrightarrow{(e,\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow) \\
\Leftrightarrow\ & \{ \text{ SOS-rules for } \rhd \ \} \\
& (\forall t' < t : B_1 \overset{t''}{\rhd}_\psi B_2 \xrightarrow{(\psi,\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow) \ \wedge \ (\forall e \in \mathsf{UE}(B_1), t' < t : B_1 \overset{t''}{\rhd}_\psi B_2 \xrightarrow{(e,\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow) \\
\Leftrightarrow\ & \{ \text{ SOS-rules for } \rhd\text{; Definition 7.32 } \} \\
& (\forall e \in \mathsf{UE}(B), t' < t : B_1 \overset{t''}{\rhd}_\psi B_2 \xrightarrow{(e,\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow) \quad .
\end{aligned}
$$

2. $B = B_1 \,||_G\, B_2$. For this case we derive:

$$
\begin{aligned}
& t \leqslant \mathsf{mt}(B_1 \,||_G\, B_2) \\
\Leftrightarrow\ & \{ \text{ Definition 7.30; definition } \mathsf{mt}\text{; calculus } \} \\
& t \leqslant \mathsf{mt}(B_1) \ \wedge \ t \leqslant \mathsf{mt}(B_2) \\
\Leftrightarrow\ & \{ \text{ induction hypothesis } \} \\
& (\forall e \in \mathsf{UE}(B_1), t' < t : B_1 \xrightarrow{(e,\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow) \ \wedge \ (\forall e' \in \mathsf{UE}(B_2), t' < t : B_2 \xrightarrow{(e',\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow) \\
\Leftrightarrow\ & \{ \text{ SOS-rule for } ||_G \ (\tau \notin G^\delta) \ \} \\
& (\forall e \in (\mathsf{UE}(B_1) \times \{*\}) \cup (\{*\} \times \mathsf{UE}(B_2)), t' < t : B_1 \,||_G\, B_2 \xrightarrow{(e,\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow) \\
\Leftrightarrow\ & \{ \text{ Definition 7.32 } \} \\
& (\forall e \in \mathsf{UE}(B_1 \,||_G\, B_2), t' < t : B_1 \,||_G\, B_2 \xrightarrow{(e,\tau,t')}\!\!\!\!\!\!\not\twoheadrightarrow) \quad .
\end{aligned}
$$

□

For PA_T we had the nice property that when we take the transition system for B induced by $\twoheadrightarrow$ and abstract from the timing aspects and event identifiers then we obtain the standard transition system for $\Phi_T(B)$, the untimed variant of B (cf. Theorem 5.10). A similar result does not hold in the setting of PA_R. A counterexample is provided, for example, by the expression $([1,2])\ a\,;\, \mathbf{0} \,||_a\, (12)\ a\,;\,\mathbf{0}$ which in the timed case leads to a transition system only consisting of an initial state (since there is no time instant at which the interaction a succeeds), whereas if we omit the time annotations, yielding $a\,;\,\mathbf{0}\,||_a\, a\,;\,\mathbf{0}$, we obtain a possible transition labelled a from the initial state to state $\mathbf{0}\,||_a\,\mathbf{0}$.

7.3.5 Consistency between causality-based and operational semantics

In order to prove the consistency between the denotational and event-based operational semantics for PA_R we follow the same approach as in Chapters 5 and 6. We present a denotational characterization of the timed event traces of B that are generated by $\longrightarrow\!\!\!\!\rightarrow$ and prove that this characterization coincides with the event traces of $\mathcal{E}_R[\![\, B \,]\!]$.

The following predicate is true iff all events in σ have a timing of at most t.

7.34. Definition. For trace σ and $t \in \mathsf{Time}$ let $\mathsf{res}(t,\sigma) \triangleq (\forall\, e_i \in \overline{[\sigma]} : t_i \leqslant t)$. □

The set of timed event traces of B is defined in a denotational way as follows.

7.35. Definition. For $B \in \mathsf{PA}_R$ the set of timed traces of B, $\mathcal{T}_R[\![\, B \,]\!]$, is defined by:

1. $\mathcal{T}_R[\![\, \mathbf{0} \,]\!] \triangleq \{\, \varepsilon \,\}$
2. $\mathcal{T}_R[\![\, \surd_\xi \,]\!] \triangleq \{\, \varepsilon \,\} \cup \{\, (\xi,\delta,t) \mid t \in \mathsf{Time} \,\}$
3. $\mathcal{T}_R[\![\, (T)\ a_\xi\,;\, B \,]\!] \triangleq \{\, (\xi,a,t)\, {}^t[\,\sigma\,] \mid t \in T \ \wedge\ \sigma \in \mathcal{T}_R[\![\, B \,]\!] \,\} \cup \{\, \varepsilon \,\}$
4. $\mathcal{T}_R[\![\, B_1 + B_2 \,]\!] \triangleq \{\, (\xi,a,t)\ \sigma \in \mathcal{T}_R[\![\, B_1 \,]\!] \mid t \leqslant \mathsf{mt}(B_2) \,\} \cup$
   $\{\, (\xi,a,t)\ \sigma \in \mathcal{T}_R[\![\, B_2 \,]\!] \mid t \leqslant \mathsf{mt}(B_1) \,\} \cup \{\, \varepsilon \,\}$
5. $\mathcal{T}_R[\![\, B_1 >\!> B_2 \,]\!] \triangleq \{\, \sigma_1\,(e,\tau,t)\, {}^t[\,\sigma_2\,] \mid \sigma_1\,(e,\delta,t) \in \mathcal{T}_R[\![\, B_1 \,]\!] \ \wedge\ \sigma_2 \in \mathcal{T}_R[\![\, B_2 \,]\!] \,\}$
   $\cup\ \{\, \sigma \in \mathcal{T}_R[\![\, B_1 \,]\!] \mid \sigma \neq \sigma'\,(e,\delta,t) \,\}$
6. $\mathcal{T}_R[\![\, B_1 \,[\!> B_2 \,]\!] \triangleq \{\, \sigma \in \mathcal{T}_R[\![\, B_1 \,]\!] \mid \sigma = \sigma'\,(e,\delta,t) \ \wedge\ \mathsf{res}(\mathsf{mt}(B_2),\sigma) \,\} \cup$
   $\{\, \sigma_1\,\sigma_2 \mid \sigma_1 \in \mathcal{T}_R[\![\, B_1 \,]\!] \ \wedge\ \sigma_2 \in \mathcal{T}_R[\![\, B_2 \,]\!] \ \wedge\ \mathsf{res}(\mathsf{mt}(B_2),\sigma_1) \ \wedge\ \sigma_1 \neq \sigma_1'(e,\delta,t) \ \wedge$
   $(\forall\, e_i \in \overline{\sigma_2} : t_i \geqslant \mathsf{mx}(\sigma_1) \ \wedge\ (\forall\, e \in \mathsf{UE}(B_1), t' < t_i : \sigma_1\,(e,\tau,t') \notin \mathcal{T}_R[\![\, B_1 \,]\!])) \,\}$
7. $\mathcal{T}_R[\![\, B[H] \,]\!] \triangleq \{\, \sigma \mid \exists\, \sigma' \in \mathcal{T}_R[\![\, B \,]\!] : \sigma = \sigma'[H] \,\}$
8. $\mathcal{T}_R[\![\, B \setminus G \,]\!] \triangleq \{\, \sigma \mid \exists\, \sigma' \in \mathcal{T}_R[\![\, B \,]\!] : \sigma = \sigma' \setminus G \,\}$
9. $\mathcal{T}_R[\![\, B_1 \,||_G\, B_2 \,]\!] \triangleq \{\, \sigma \in (\overline{\mathcal{T}_R[\![\, B_1 \,]\!]} \bowtie_G \overline{\mathcal{T}_R[\![\, B_2 \,]\!]})^* \mid \pi_i(\sigma) \in \mathcal{T}_R[\![\, B_i \,]\!] \text{ for } i{=}1,2 \,\}$
10. $\mathcal{T}_R[\![\, B_1 \overset{t}{\rhd}_\xi B_2 \,]\!] \triangleq \{\, (e,a,t')\,\sigma \in \mathcal{T}_R[\![\, B_1 \,]\!] \mid t' \leqslant t \,\}$
    $\cup\ \{\, (\xi,\tau,t)\, {}^t[\,\sigma\,] \mid t \leqslant \mathsf{mt}(B_1) \ \wedge\ \sigma \in \mathcal{T}_R[\![\, B_2 \,]\!] \,\} \cup \{\, \varepsilon \,\}$
11. $\mathcal{T}_R[\![\, B_1 \overset{t}{\blacktriangleright} B_2 \,]\!] \triangleq \{\, \sigma \in \mathcal{T}_R[\![\, B_1 \,]\!] \mid \sigma = \sigma'\,(e,\delta,t') \ \wedge\ \mathsf{res}(t,\sigma) \,\} \cup$
    $\{\, \sigma_1\, {}^t[\,\sigma_2\,] \mid \sigma_1 \in \mathcal{T}_R[\![\, B_1 \,]\!] \ \wedge\ \sigma_2 \in \mathcal{T}_R[\![\, B_2 \,]\!] \ \wedge\ \sigma_1 \neq \sigma_1'\,(e,\delta,t') \ \wedge\ \mathsf{res}(t,\sigma_1)$
    $\wedge\ (\forall\, e \in \mathsf{UE}(B_1), e_i \in \overline{\sigma_2}, t' < t_i : \sigma_1\,(e,\tau,t') \notin \mathcal{T}_R[\![\, B_1 \,]\!])) \,\}.$

□

It can be proven in a similar way as in Chapter 5 that $\mathcal{T}_R[\![B]\!]$ equals the set of timed event traces of B generated by the inference rules for $\longrightarrow\!\!\!\!\!\rightarrow$. Let $\xrightarrow{\sigma}\!\!\!\!\rightarrow$ be the extension of $\longrightarrow\!\!\!\!\!\rightarrow$ for traces in the usual way.

7.36. LEMMA. $\forall B \in \mathsf{PA}_R : \mathcal{T}_R[\![B]\!] = \{ \sigma \mid \exists B' : B \xrightarrow{\sigma}\!\!\!\!\rightarrow B' \}$.

PROOF. Straightforward, but elaborative. □

In order to relate the operationally characterized timed event traces and the traces obtained from the causality-based semantics $\mathcal{E}_R[\![\ ]\!]$ we slightly adapt the definition of $\mathcal{E}_R[\![\ ]\!]$ for $\surd$, $(t)\ a\,;$, and $\triangleright$. In the current definition of $\mathcal{E}_R[\![\ ]\!]$ a unique but arbitrary event is introduced for these constructs modelling the appearance of δ, a, or a timeout, respectively. Here we take the unique event identification for this operators in the definition of $\mathcal{E}_R[\![\ ]\!]$. E.g., for $\surd_\xi$ a new event ξ is introduced (and labelled δ).

The following theorem states that the set of timed event traces of a behaviour expression B of PA_R is identical to the set of timed event traces of the corresponding timed event structure $\mathcal{E}_R[\![B]\!]$.

7.37. THEOREM. $\forall B \in \mathsf{PA}_R : T_R(\mathcal{E}_R[\![B]\!]) = \mathcal{T}_R[\![B]\!]$.

PROOF. The proof is by induction on the structure of B.
Base: For $B = \mathbf{0}$ we simply have $T_R(\mathcal{E}_R[\![\mathbf{0}]\!]) = \{ \varepsilon \} = \mathcal{T}_R[\![\mathbf{0}]\!]$, and for $B = \surd_\xi$ we have $T_R(\mathcal{E}_R[\![\surd_\xi]\!]) = \{ \varepsilon \} \cup \{ (\xi, \delta, t) \mid t \in \mathsf{Time} \} = \mathcal{T}_R[\![\surd_\xi]\!]$.
Induction Step: Assume the theorem holds for B_1 and B_2. We only provide proofs for timed action prefix, choice, disrupt, parallel composition, timeout and watchdog. The proofs for the other operators are conducted in a similar way and are omitted. Let $\Lambda = \mathcal{E}_R[\![B]\!]$ and $\Lambda_i = \mathcal{E}_R[\![B_i]\!] = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$ with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2.

1. $B = (T)\ a_\xi\,; B_1$. For Λ bundles $\{ \{ (\xi, a) \} \} \times E_1$ have been added to $\langle (\{ \xi \}, \varnothing, \varnothing, \{ (\xi, a) \}), \{ (\xi, T) \}, \varnothing, \{ (\xi, \text{false}) \} \rangle$. The non-empty timed event traces of Λ are therefore those interleavings of (ξ, a, t) and ${}^t[\, \sigma \,]$, with $\sigma \in T_R(\Lambda_1)$, that satisfy the following constraints: (i) the first element of ${}^t[\, \sigma \,]$ is preceded by (ξ, a, t), and (ii) $t \in \mathcal{D}(\xi) = T$. Thus we derive:
$$\begin{array}{ll} & T_R(\mathcal{E}_R[\![(T)\ a_\xi\,; B_1]\!]) \\ = & \{ \text{ see above } \} \\ & \{ (\xi, a, t)\ {}^t[\, \sigma \,] \mid t \in T \ \wedge\ \sigma \in T_R(\Lambda_1) \} \cup \{ \varepsilon \} \\ = & \{ \text{ induction hypothesis } \} \\ & \{ (\xi, a, t)\ {}^t[\, \sigma \,] \mid t \in T \ \wedge\ \sigma \in \mathcal{T}_R[\![B_1]\!] \} \cup \{ \varepsilon \} \\ = & \{ \text{ Definition 7.35 } \} \\ & \mathcal{T}_R[\![(T)\ a_\xi\,; B_1]\!] \quad . \end{array}$$

2. $B = B_1 + B_2$. The proof for this construct is analogous to the proof of Theorem 6.34.

3. $B = B_1 [> B_2$. From the untimed case we know that traces of Λ are either (i) traces σ_1 of Λ_1 that end with a δ, or (ii) concatenations of traces $\sigma_1 \in T_R(\Lambda_1)$ and $\sigma_2 \in T_R(\Lambda_2)$ where σ_1 does not contain a δ. Like for the urgent case (cf. Theorem 6.34) we have to take into account that due to the added asymmetric conflicts in Λ initial urgent events of Λ_2 may prevent the

occurrence of events in Λ_1. More specifically, σ_1 is part of a trace of Λ provided that there is no initial urgent event in Λ_2 that can occur earlier than some event in σ_1. We now characterize set (i) and derive for this set:

$$\begin{aligned}
& \{\, \sigma \in T_R(\Lambda_1) \mid \sigma = \sigma'\,(e,\delta,t) \;\wedge\; (\forall\, e_i \in \overline{[\sigma]}, e' \in init(\Lambda_2) : \mathcal{U}_2(e') \;\Rightarrow\; t_i \leqslant \mathcal{D}_2(e'))\,\} \\
= \;& \{ \text{ calculus } \} \\
& \{\, \sigma \in T_R(\Lambda_1) \mid \sigma = \sigma'\,(e,\delta,t) \;\wedge \\
& \qquad\qquad (\forall\, e_i \in \overline{[\sigma]} : t_i \leqslant Min\{\, \mathcal{D}_2(e') \mid e' \in init(\Lambda_2) \;\wedge\; \mathcal{U}_2(e')\,\})\,\} \\
= \;& \{ \text{ Lemma 7.33 } \} \\
& \{\, \sigma \in T_R(\Lambda_1) \mid \sigma = \sigma'\,(e,\delta,t) \;\wedge\; (\forall\, e_i \in \overline{[\sigma]} : t_i \leqslant \mathsf{mt}(B_2))\,\} \\
= \;& \{ \text{ Definition 7.34 } \} \\
& \{\, \sigma \in T_R(\Lambda_1) \mid \sigma = \sigma'\,(e,\delta,t) \;\wedge\; \mathsf{res}(\mathsf{mt}(B_2),\sigma)\,\} \\
= \;& \{ \text{ induction hypothesis } \} \\
& \{\, \sigma \in \mathcal{T}_R[\![\, B_1 \,]\!] \mid \sigma = \sigma'\,(e,\delta,t) \;\wedge\; \mathsf{res}(\mathsf{mt}(B_2),\sigma)\,\} \quad .
\end{aligned}$$

A similar derivation can be carried out for set (ii), taking into account the asymmetric conflicts between E_1 and $init(\Lambda_2)$. By Definition 7.35 the union of the thus obtained sets equals $\mathcal{T}_R[\![\, B_1 \,[\!>\, B_2 \,]\!]$.

4. $B = B_1 \,||_G\, B_2$. Since synchronizations of urgent events cannot appear (cf. Lemma 7.25) no new (asymmetric) conflicts are introduced between urgent events in Λ_1 and events in Λ_2 (or vice versa). This means that $\sigma \in T_R(\Lambda)$ iff $\pi_i(\sigma) \in T_R(\Lambda_i)$, for i=1,2. So, $T_R(\Lambda)$ equals

   $$\{\, \sigma \in (\overline{T_R(\Lambda_1)} \;\bowtie_G\; \overline{T_R(\Lambda_2)})^* \mid \pi_1(\sigma) \in T_R(\Lambda_1) \;\wedge\; \pi_2(\sigma) \in T_R(\Lambda_2)\,\}.$$

   By the induction hypothesis this equals

   $$\{\, \sigma \in (\overline{\mathcal{T}_R[\![\, B_1 \,]\!]} \bowtie_G \overline{\mathcal{T}_R[\![\, B_2 \,]\!]})^* \mid \pi_1(\sigma) \in \mathcal{T}_R[\![\, B_1 \,]\!] \;\wedge\; \pi_2(\sigma) \in \mathcal{T}_R[\![\, B_2 \,]\!]\,\}.$$

   By Definition 7.35 this equals $\mathcal{T}_R[\![\, B_1 \,||_G\, B_2 \,]\!]$.

5. $B = B_1 \overset{t}{\rhd}_\xi B_2$. The plain event structure corresponding to Λ equals $\mathcal{E}'[\![\, B_1 + \tau_\xi \,;\, B_2 \,]\!]$. This means that untimed traces are either traces of $\mathcal{E}_1$ or traces of $\mathcal{E}_2$ preceded by ξ. Since in Λ event ξ is urgent and has delay $\mathcal{D}(\xi) = [t,t]$, it follows that the timed event traces of Λ are either (i) traces of Λ_1 that start before (or at) t—since otherwise ξ will appear and disable all initial events of Λ_1—or (ii) traces of the form $(\xi,\tau,t)\,{}^{t}[\,\sigma\,]$ where σ is a trace of Λ_2, or (iii) empty traces. Since for urgent $e \in init(\Lambda_1)$ we have $e \rightsquigarrow \xi$ it follows (according to the third constraint of Definition 7.5) that ξ can only occur if $t \leqslant \mathcal{D}_1(e)$; otherwise e should precede ξ. Thus,

   $$\begin{aligned}
   & T_R(\mathcal{E}_R[\![\, B_1 \overset{t}{\rhd}_\xi B_2 \,]\!]) \\
   = \;& \{ \text{ see discussion above } \} \\
   & \quad \{\, (e,a,t')\,\sigma \in T_R(\Lambda_1) \mid t' < t \,\} \cup \{\, \varepsilon \,\} \\
   & \cup \{\, (\xi,\tau,t)\,{}^{t}[\,\sigma\,] \mid \sigma \in T_R(\Lambda_2) \;\wedge\; (\forall\, e \in init(\Lambda_1) : \mathcal{U}_1(e) \;\Rightarrow\; t \leqslant \mathcal{D}_1(e))\,\} \\
   = \;& \{ \text{ calculus } \} \\
   & \quad \{\, (e,a,t')\,\sigma \in T_R(\Lambda_1) \mid t' < t \,\} \cup \{\, \varepsilon \,\} \\
   & \cup \{\, (\xi,\tau,t)\,{}^{t}[\,\sigma\,] \mid \sigma \in T_R(\Lambda_2) \;\wedge\; t \leqslant Min\{\, \mathcal{D}_1(e) \mid e \in init(\Lambda_1) \;\wedge\; \mathcal{U}_1(e)\,\}\,\}
   \end{aligned}$$

$$
\begin{aligned}
&= \quad \{\text{ Lemma 7.33 }\}\\
&\quad \{\,(e,a,t')\,\sigma \in T_R(\Lambda_1) \mid t' < t\,\} \cup \{\,(\xi,\tau,t)\,{}^t[\,\sigma\,] \mid \sigma \in T_R(\Lambda_2) \;\wedge\; t \leqslant \mathsf{mt}(B_1)\,\} \cup \{\,\varepsilon\,\}\\
&= \quad \{\text{ induction hypothesis }\}\\
&\quad \{\,(e,a,t')\,\sigma \in \mathcal{T}_R[\![\,B_1\,]\!] \mid t' < t\,\} \cup \{\,(\xi,\tau,t)\,{}^t[\,\sigma\,] \mid \sigma \in \mathcal{T}_R[\![\,B_2\,]\!] \wedge t \leqslant \mathsf{mt}(B_1)\,\} \cup \{\,\varepsilon\,\}\\
&= \quad \{\text{ Definition 7.35 }\}\\
&\quad \mathcal{T}_R[\![\,B_1 \overset{t}{\rhd}_\xi B_2\,]\!] \quad .
\end{aligned}
$$

6. $B = B_1 \overset{t}{\blacktriangleright} B_2$. The untimed event structure of Λ is equal to that of $B_1\,[>\,B_2$. From the untimed case we know that event traces of this expression are either (i) traces of $\mathcal{E}_1$ that end with a δ, or (ii) concatenations of traces σ_1 of $\mathcal{E}_1$ and σ_2 of $\mathcal{E}_2$ such that no δ occurs in σ_1. In the real-time case the delay of all events in E_1 is restricted by $[0,t]$. This means that all events in the traces characterized under (i) should appear at time t at the latest; for the same reason this also holds for all events in σ_1 under (ii). The proof for (i) is similar to the one presented for $[>$. Consider traces characterized by (ii). The delay of all events in E_2 is postponed by t time units. This means that all events in the traces (ii) are of the form $\sigma_1\,{}^t[\,\sigma_2\,]$. Since $E_1 \rightsquigarrow e$ for all $e \in \mathit{init}(\Lambda_2)$, e can only appear in σ_2 iff there is no urgent event enabled in Λ_1 after the execution of σ_1 that can occur earlier (according to the third constraint of Definition 7.5).

$\square$

7.38. Corollary. $\forall\, B, B_1, B_2 \in \mathsf{PA}_R, t, t' \in \mathsf{Time}$:

$$
\Big(B \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B_1 \xrightarrow{(e',a',t')}\!\!\!\!\rightarrow B_2 \;\wedge\; t' < t\Big) \;\Rightarrow\; (\exists\, B' : B \xrightarrow{(e',a',t')}\!\!\!\!\rightarrow B' \xrightarrow{(e,a,t)}\!\!\!\!\rightarrow B_2).
$$

Proof. Directly from Theorems 7.37 and 7.7. $\square$

Let $\mathsf{TS}_R(B)$ be the timed event transition system obtained by $\longrightarrow\!\!\!\!\rightarrow$ and $\mathsf{ETS}_R(\mathcal{E}_R[\![\,B\,]\!])$ the transition system obtained by considering $\mathcal{E}_R[\![\,B\,]\!]$ as initial state and having transitions from Λ to Λ' iff $\Lambda' = \Lambda[\sigma]$ for some $\sigma \in T_R(\Lambda)$ with length 1. Then it follows that:

7.39. Theorem. $\forall\, B \in \mathsf{PA}_R : \mathsf{TS}_R(B) \sim \mathsf{ETS}_R(\mathcal{E}_R[\![\,B\,]\!])$.

Proof. Similar to the proof of Theorem 2.46. $\square$

7.3.6 An alternative approach for PA_R

This section considers an alternative event-based operational semantics for PA_R in the same spirit as in Section 5.4 (and Chapter 6). We only consider timed action-prefix, timeout, and watchdog. For the other operators the inference rules are identical to those for PA_T; the reason that the inference rules of $+$ and $[>$ from Section 5.4 do not have to be changed is due to the fact that we consider a time-consistent setting now.

Timed action-prefix

$(T)\ a_\xi\ ;\ B$ at time t can perform (ξ, a) if $0 \in T$ and behaves subsequently like B (at t). Time can always be passed by $(T)\ a_\xi\ ;\ B$. Let $T \ominus t \triangleq \{\, t'-t \mid t' \in T \ \wedge\ t' \geqslant t \,\}$.

$$\frac{}{\langle (T)\, a_\xi\ ;\ B, t\rangle \rightsquigarrow \langle (T \ominus (t'-t))\, a_\xi\ ;\ B, t'\rangle}\ (t' \geqslant t)$$

$$\frac{}{\langle (T)\, a_\xi\ ;\ B, t\rangle \xrightarrow{(\xi,a)} \langle B, t\rangle}\ (0 \in T)$$

Timeout

If the first component B_1 permits the passage of time with at most t time units while evolving into B_1' then $B_1 \overset{t}{\rhd} B_2$ allows the same, evolving into $B_1' \overset{d}{\rhd} B_2$ where d equals t minus the number of time units that have been passed. $B_1 \overset{0}{\rhd}_\psi B_2$ at time t can perform the timeout event ψ while evolving into B_2 (at t). If B_1 performs an event and evolves into B_1' then $B_1 \overset{t}{\rhd} B_2$ allows the same, also evolving into B_1'.

$$\frac{\langle B_1, t'\rangle \rightsquigarrow \langle B_1', t''\rangle}{\langle B_1 \overset{t}{\rhd}_\psi B_2, t'\rangle \rightsquigarrow \langle B_1' \overset{t-(t''-t')}{\rhd_\psi} B_2, t''\rangle}\ (t''-t' \leqslant t)$$

$$\frac{}{\langle B_1 \overset{0}{\rhd}_\psi B_2, t\rangle \xrightarrow{(\psi,\tau)} \langle B_2, t\rangle} \qquad \frac{\langle B_1, t'\rangle \xrightarrow{(\xi,a)} \langle B_1', t'\rangle}{\langle B_1 \overset{t}{\rhd}_\psi B_2, t'\rangle \xrightarrow{(\xi,a)} \langle B_1', t'\rangle}$$

Watchdog

$B_1 \overset{t}{\blacktriangleright} B_2$ allows the passage of time in the same way as $\overset{t}{\rhd}$, and in addition, if B_2 permits the passage of time then $B_1 \overset{0}{\blacktriangleright} B_2$ can do the same, also evolving into B_2'. If B_1 performs event (ξ, a) and evolves into B_1' then $B_1 \overset{t}{\blacktriangleright} B_2$ can do the same and evolves into either $B_1' \overset{t}{\blacktriangleright} B_2$ if $a \neq \delta$, or B_1' if $a = \delta$. Finally, if $B_1 \overset{0}{\blacktriangleright} B_2$ can perform an event and evolves into B_2' if B_2 can do so.

$$\frac{\langle B_1, t'\rangle \rightsquigarrow \langle B_1', t''\rangle}{\langle B_1 \overset{t}{\blacktriangleright} B_2, t'\rangle \rightsquigarrow \langle B_1' \overset{t-(t''-t')}{\blacktriangleright} B_2, t''\rangle}\ (t''-t' \leqslant t)$$

$$\frac{\langle B_1, t'\rangle \xrightarrow{(\xi,a)} \langle B_1', t'\rangle}{\langle B_1 \overset{t}{\blacktriangleright} B_2, t'\rangle \xrightarrow{(\xi,a)} \langle B_1' \overset{t}{\blacktriangleright} B_2, t'\rangle}\ (a \neq \delta) \qquad \frac{\langle B_1, t'\rangle \xrightarrow{(\xi,\delta)} \langle B_1', t'\rangle}{\langle B_1 \overset{t}{\blacktriangleright} B_2, t'\rangle \xrightarrow{(\xi,\delta)} \langle B_1', t'\rangle}$$

$$\frac{\langle B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle}{\langle B_1 \overset{0}{\blacktriangleright} B_2, t\rangle \rightsquigarrow \langle B_2', t'\rangle}\ (t'-t > 0) \qquad \frac{\langle B_2, t'\rangle \xrightarrow{(\xi,a)} \langle B_2', t'\rangle}{\langle B_1 \overset{t}{\blacktriangleright} B_2\rangle \xrightarrow{(\xi,a)} \langle B_2', t'\rangle}$$

We conclude this section by considering the model properties time determinism, action persistency and time additivity. Since the passage of time is always uniquely determined it follows that time determinism is respected. This can easily be checked by structural induction on B. The alternative event-based operational semantics for PA_R, however, violates action persistency. This entails that the passage of time may suppress the possibility to perform an action. This is not surprising, since in PA_R we have the possibility to specify upper bounds of occurrence of actions, and as soon as time passes beyond this upper bound the possibility to perform this action is lost. For example, transition

$$\langle([0,3])\ a\,;\,\mathbf{0},0\rangle \rightsquigarrow \langle(\varnothing)\ a\,;\,\mathbf{0},\pi\rangle$$

makes it impossible to perform a in the resulting state, whereas a is possible in the starting state.

The alternative event-based operational semantics for PA_R also violates time additivity, as shown by

$$\langle(2)\ a\,;\,\mathbf{0} \overset{7}{\blacktriangleright} (3)\ b\,;\,\mathbf{0},0\rangle \rightsquigarrow \langle(0)\ a\,;\,\mathbf{0} \overset{0}{\blacktriangleright} (3)\ b\,;\,\mathbf{0},7\rangle \rightsquigarrow \langle(0)\ b\,;\,\mathbf{0},23\rangle \quad .$$

There is no single $\rightsquigarrow$ transition that mimics this two-step transition. The reason that the timeout operator does respect time additivity is that at time t an internal (timeout) event is forced to occur, such that time can never pass beyond t without performing this event. Time additivity is obtained if we add the following rule for $\blacktriangleright$:

$$\frac{\langle B_1,t_1\rangle \rightsquigarrow \langle B_1',t_2\rangle \ \wedge\ \langle B_2,t_2\rangle \rightsquigarrow \langle B_2',t_3\rangle}{\langle B_1 \overset{t}{\blacktriangleright} B_2,t'\rangle \rightsquigarrow \langle B_2',t_3\rangle} \quad (t_2{-}t_1 = t\ \wedge\ t_3{-}t_2 > 0)$$

A similar construction is used in ATP_D of Nicollin *et al.* [113] to establish time additivity.

Let $\mathcal{T}_R^*[\![\, B \,]\!]\, t$ denote the set of timed event traces of $\langle B,t\rangle$ under $\rightsquigarrow$ and $\longrightarrow\!\!\!\!\!\rightarrow$. We then have:

7.40. LEMMA. $\forall\, B \in \mathsf{PA}_R, t \in \mathsf{Time} : \mathcal{T}_R^*[\![\, B \,]\!]\, t = \{\, {}^t[\,\sigma\,] \mid \sigma \in \mathcal{T}_R[\![\, B \,]\!] \ \wedge\ tc(\sigma)\,\}$.

PROOF. By induction on the structure of B; similar to Lemma 5.27. □

7.41. COROLLARY. $\forall\, B \in \mathsf{PA}_R : \mathcal{T}_R^*[\![\, B \,]\!]\, t = \{\, {}^t[\,\sigma\,] \mid \sigma \in T_R(\mathcal{E}_R[\![\, B \,]\!]) \ \wedge\ tc(\sigma)\,\}$.

PROOF. Straightforward from the previous lemma and Theorem 7.37. □

7.4 Time in causality-based models

In the literature numerous timed models are proposed based on an interleaving semantics, usually being defined using a kind of timed transition system. Only a few timed models are known (to us) based on a causality-based model. In this section we briefly discuss some existing timed causality-based models.

The only timed model that allows sets of time instants to be associated with events (or causal dependencies) is introduced by Fidge [47]. Fidge proposes a real-time extension of causal trees, a causality-based model introduced in Darondeau & Degano [36], and uses this model to provide a semantics to a timed variant of CCS. Each event e in a causal tree has a set of *backward pointers* to each event on which e causally depends. Time constraints are expressed by associating a set of relative times to events. The relative delays T state that an event can only occur at t time units (for some $t \in T$) after the time at which *all* its causally preceding events occurred (if any). Synchronization can only occur if both participants are willing to engage in the interaction at the same time instant; if not, the synchronization will not take place. Because in the causal tree model different occurrences of the same action cannot be identified as such, Fidge's model must be considered as a timed pomset model whereas our model is a timed lposet model (see Chapter 1 for a discussion about pomsets versus lposets). The real-time semantics of CCS is defined operationally. Due to the adjustments of backward pointers the inference rules are somewhat complicated and the relation with the standard rules for CCS is not so clear.

An extension of Pratt's pomset model [121] with delays is studied in Casley *et al.* [32, 31]. The delays in the model specify the minimal relative delay between two causally dependent actions. Casley *et al.* use a kind of metric space for their model and define several operations on these structures that are generalizations of operations on Pratt's pomset model like concatenation and concurrency. E.g., $P ;^d Q$ specifies that there is a delay of at least d time units between each event in P and each event in Q. They also define some operators that rely on the location where an action occurs. E.g., $P \,\underline{;}^d\, Q$ differs from concatenation in that additional timing constraints are introduced only between colocated actions in P and Q rather than between all of them.

Maggiolo-Schettini & Winkowski [99] consider timed configurations. They distinguish between the time at which an event is enabled (the *enabling time*) and the time at which an event actually happens (its *completion time*). Synchronization structures describe how actions of composed behaviours are combined into actions of the resulting behaviour and which actions are considered to be internal. Two (or more) events can synchronize if they are equally labelled and have identical completion times. Similar to our model of Chapter 4, the enabling time of the resulting event is the maximum of the enabling times of its components. The authors define several operations on their structures (such as sequential and parallel composition, abstraction, choice, and a fixed point operator). An equivalence relation is introduced which is a congruence w.r.t. the operations introduced. The main limitation of this model is that all events are required to happen as soon as possible in some sense. (Recall that a semantics of extended bundle event structures at configuration level is not sufficient due to the presence of asymmetric conflict; see Chapter 2.)

The most extensive treatment of time in a causality-based context is due to Murphy. An interesting timed variant of event structures, called interval event structures, is proposed in [106, 108]. In this model, each event has a duration modelled as the time between the start of an event and its finish. An event with duration d could be modelled in our model by explicitly representing the start and finish of an event by two distinct events, the start causing the finish, and the interval $[d, d]$ associated to this bundle. A fictitious silent event is introduced the start of which causes every event, and all events cause its finish. The model incorporates symmetric conflict, generalizes Winskel's prime event structures, and allows to express Lamport's model of distributed systems [88].

The behaviour of timed systems with both conjunctive and disjunctive causality is studied by Gunawardena in [61, 62]. Like in our model conjunctive causality, corresponding to synchronization, results in a maximum timing constraint. All events are required to happen at exactly the minimal time at which they are enabled. For disjunctive causality an event has to wait for the first event in the set of its enabling events. This boils down to a minimum timing constraint. This implies that in this model an event always is enabled by the first event that occurs in case of disjunctive causality. Gunawardena studies the relationship of his model, timed $\{\,\mathsf{AND}, \mathsf{OR}\,\}$ automata, and the theory of min-max functions. Notions like periodicity can be characterized and cycle times of periodic behaviours can be determined. Since the model does not (yet) include disablings no conflicts between events are incorporated.

Janssen *et al.* [78] introduce a real-time process language consisting of simple sequential processes that are composed by means of layering ($\bullet$) and independent parallelism ($|||$). $P \bullet Q$ executes P and Q in parallel, except when some action in Q is dependent on some action in P; in that case the action in P is guaranteed to happen first. The denotational semantics of a real-time expression is a set of partially ordered runs where a run consists of a set of events (each event having a duration) and a partial order on these events. This order is determined by a causal order and a temporal order, the latter being induced by real-time constraints.

7.5 Conclusions

In this chapter we have presented a real-time extension of extended bundle event structures that allows for the decoration of events and bundles by arbitrary sets of time instants. The model incorporates urgent events and is shown to be sufficiently expressive to support important real-time notions such as timeouts and watchdogs (or timed interrupts). Since urgent events are used in a somewhat restricted way (as opposed to Chapter 6) most of the theory of timed event structures is generalized to the more liberal timed setting in a rather straightforward way. An important consequence of the possibility to prevent an event to occur after a certain time instant (by specifying an upper bound in time or by a conflicting urgent event) is that the model is no longer a conservative extension of the untimed model. That is, the untimed lposets of a real-time event structure are a subset of the lposets of its corresponding untimed (extended bundle) event structure, but equality does not necessarily hold.

An interaction can take place if all participants can engage in it at the same time instant. The interaction cannot appear if such common time instant does not exist. Since in our model we do not have an explicit notion of the passage of time, such an impossible interaction does not result in behaviours which do block the passage of time (so-called *timelocks*) in the entire system—even in causally independent parts!—but simply in the local impossibility to execute the event at hand.

We have considered timeout ($\triangleright$) and watchdog ($\blacktriangleright$) operators in a process algebraic context. $B_1 \overset{t}{\triangleright} B_2$ is modelled by $B_1 + ([t,t])\ \tau\ ;\ B_2$ where τ is required to be urgent and is intended to represent the expiration of a timer. $\blacktriangleright$ could be modelled without the introduction of auxiliary urgent events. Although we used urgent events only for modelling timeout mechanisms, they have an impact on the evolvements of other subprocesses in the context of $+$, $[>$, $\blacktriangleright$, and $\triangleright$. This made the event-based operational semantics of PA_R using timed-actions somewhat more complex. These problems do not appear when separating the passage of time and the occurrence of events: the inference rules for $+$ and $[>$ remain unaffected. We need, however, 9 inference rules to incorporate $\blacktriangleright$ and $\triangleright$. Since upper bounds on the occurrence of actions can be specified action persistency is lost.

Compared to the urgent event structures of Chapter 6 the incorporation of urgent events in real-time event structures is restricted. This resulted in a characterization of timed event traces without being forced to time-consistency (as in Chapter 6). Like for the simple timed model of Chapter 4 we have that for each ill-timed trace there exists a corresponding time-consistent trace with the same timed events.

8 The stochastic timing module

This chapter proposes stochastic variants of extended bundle event structures. As a result causality-based models are obtained that allow the specification of stochastic timing constraints. Events are supposed to happen after a delay that is determined by a stochastic variable with a certain distribution function. First, a simple model is discussed restricting the distribution functions to be exponential. Then the generalization of deterministic times towards more general types of distributions is investigated and a stochastic variant of event structures is proposed with (the more practical) phase-type distributions. This class of distributions includes exponential, Erlang, Coxian and mixtures of exponential distributions. It is shown how both stochastic models can be used to provide a compositional causality-based semantics to a stochastic extension of PA, and for the exponential case a corresponding event-based operational semantics is provided that is proven to coincide with various existing interleaving proposals.

8.1 Introduction

In Chapter 4, 6 and 7 we extended event structures with time and urgency. This facilitates the specification and analysis of deterministic time constraints. In early stages of the design there is often no exact timing information available and in, for instance, multi-media systems phenomena like jitter and response times are not deterministically determined but much more of a stochastic nature. In these cases the use of deterministic timed extensions is not always appropriate. Therefore, it seems to be useful to let the time of occurrence of actions be determined by *stochastic (or random) variables* rather than by constants. In this way a model would be obtained that enables the description of more dynamic stochastic behaviour. See also the discussion in Chapter 1.

This chapter investigates the incorporation of *stochastic timing* into extended bundle event structures. In our timed causality-based model time is associated to causal relations (termed bundles in our model) and to events. Bundle delays specify the relative delay between causally dependent events while event delays enable the specification of timing constraints on events that have no incoming bundle. In this timed model components may synchronize on a common action as soon as all participants are ready to engage, that is, when all individual timing constraints are met. The material presented in this chapter is based on the generalization of deterministic times in our timed model towards *distribution functions* (note that a distribution function uniquely determines a stochastic variable, and vice versa).

We start by investigating a generalization of our timed model of Chapter 4 in which, for simplicity, we restrict to exponential distributions. This results in a simple stochastic event structure model where rates are associated with events only (and not to bundles). The principle that a synchronization takes place as soon as all participants are ready for it means in a stochastic setting that the delay of such action will be distributed as the product of the individual distributions (or, equivalently, as the maximum of the corresponding individual stochastic variables, under the assumption of statistical independence). Since the class of exponential distributions is *not* closed under product, we abandon our synchronization principle of the timed model and take (just for this model) a pragmatic approach by computing the rate of a synchronization simply as a function of the individual rates—similar to several existing stochastic extensions of process algebras. The resulting model is used to provide a compositional causality-based semantics of a simple stochastic process algebra. A corresponding event-based operational semantics is provided (in the same spirit as is done in Chapter 5 for the timed model) which shows that our simple stochastic model closely resembles existing interleaved proposals of stochastic process algebras.

Current stochastic process algebras all use (extensions of) labelled transition systems as an underlying semantical model. This results in a semantics based on the interleaving of causally independent actions. The structure of transition systems closely resembles that of standard Markov chains, which is an advantage when trying to obtain a performance model directly from the formal model. In addition, the elegant—memoryless—properties of exponential distributions enables a smooth incorporation of such distributions into transition systems. The interleaving of causally independent actions, however, complicates the use of more general (nonmemoryless) distributions in transition systems considerably [59].

This aspect is illustrated in Figure 8.1 where the depicted transition system intuitively corresponds to $(F)\ a\,;\mathbf{0} \,|||\, (G)\ b\,;\mathbf{0}$ with F, G distribution functions. In case F and G are memoryless (i.e., exponential distributions) then the time until the occurrence of b (a) after the occurrence of a (b) is still distributed by G (F) irrespective of how much time has elapsed until a (b) occurred. However, in case the memoryless property is not satisfied the residual lifetime of the stochastic variable determined by G since the occurrence of a must be computed in order to correctly deduce the time until b's occurrence. Here, the global state assumption

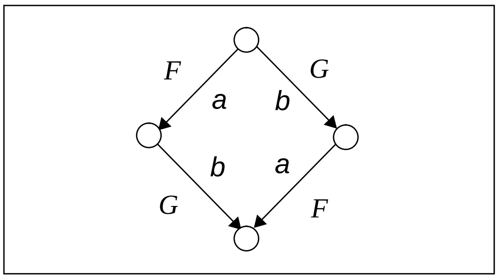


Figure 8.1: Independent actions in a stochastic transition system.

complicates the incorporation of nonmemoryless distributions considerably (despite attempts to circumvent this problem by Götz *et al.* [59]). We hope to show in this chapter that a causality-based model avoids these problems.

When carefully investigating the replacement of deterministic times in our timed model by general distributions it turns out that it is possible to support a class of distributions which is closed under product (corresponding to the maximum of stochastic variables under the assumption of statistical independence), and which contains an identity element for product. These properties will be justified in this chapter. As an interesting class of distribution functions that satisfies these criteria we propose the use of *phase-type* (*PH-*) distributions. PH-distributions can be considered as matrix generalizations of exponential distributions and are well-suited for numerical computation. They are used in many probabilistic models that have matrix-geometric solutions, have a richly developed theory due to Neuts [109, 110], and include frequently used distributions in performance analysis such as hyper- and hypo-exponential, Erlang, and Cox distributions.

This chapter is organized as follows. Section 8.2 reports on the study of exponential distributions in our model, introduces a simple stochastic process algebra including a causality-based semantics, and relates this semantics to existing interleaved proposals. Section 8.3 investigates the use of more general distribution functions in extended bundle event structures and justifies why we are interested in a class of distribution functions which is closed under product and which contains an identity element for product. It introduces PH-distributions and provides some important results that are relevant in the context of this chapter. Finally, Section 8.4 contains conclusions and pointers for future work. Appendix A contains a brief introduction into stochastic notions such as distribution functions and Markov chains.

8.2 Simple stochastic event structures

As a prerequisite we consider exponential distributions. Exponential distributions are defined as follows.

8.1. Definition. A distribution function F, defined by $F(x) = 1 - e^{-\lambda x}$, for $x \geqslant 0$, and $F(x) = 0$, for $x < 0$, is an *exponential distribution* with *rate* λ ($\lambda \in \mathbb{R}^+$). $\Box$

Evidently, a rate uniquely characterizes an exponential distribution. A well-known property of exponential distributions is the memoryless property.

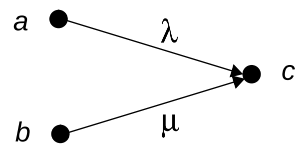
8.2. Lemma. For U an exponentially distributed stochastic variable and $x, y \geqslant 0$ we have $Pr\{\, U \leqslant x + y \mid U > y \,\} = Pr\{\, U \leqslant x \,\}$. This property is known as the *memoryless* (or Markovian) property.

Proof. Standard, see for instance Kobayashi [87]. $\Box$

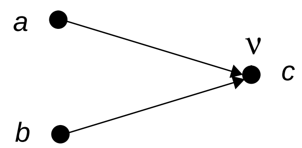
Informally, it states that the probability of U being at most $x+y$ given that it is larger than y is independent of y and equal to the probability of U being at most x.

8.2.1 The model

In this section we develop a simple stochastic variant of extended bundle event structures by associating rates to events. The motivation for only associating rates to events, and not to bundles too, is that when choosing to remain in the domain of exponential distributions it turns out to be sufficient to attach rates to events only. Consider, for example, the following event structure in which rates are associated to bundles:



The interpretation is that a rate associated to bundle X pointing to e determines the time of e's enabling relative to the time of occurrence of its causal predecessor in X. The above structure specifies that the time period between the enabling of e_c and the occurrence of e_a (e_b) is exponentially distributed with rate λ (μ). Given that we want to stay in the domain of exponential distributions this is equivalent to saying that the time between the last occurrence of an event preceding e_c and the enabling of e_c is exponentially distributed with rate ν where ν is determined by λ and μ. Due to the memoryless property this is statistically equivalent to saying that the period between the start of the system and the enabling of e_c is exponentially distributed with rate ν:



Therefore we choose to associate rates to events only. In this way we also keep close to the stochastic transition systems that underly stochastic process algebra based on interleaving (see also Section 8.2.3). Thus,

8.3. Definition. (*Simple stochastic event structure*)

A *simple stochastic* event structure is a tuple $\langle \mathcal{E}, \mathcal{R} \rangle$ with $\mathcal{E}$ an extended bundle event structure $(E, \leadsto, \mapsto, l)$ and $\mathcal{R} : E \longrightarrow \mathbb{R}^+$, the *rate* function. □

As a generalization of the notion of event trace we define the notion of stochastic event trace. We use Σ, possibly subscripted and/or primed, to denote stochastic event structures.

8.4. Definition. (*Stochastic event trace*)

A *stochastic event trace* of stochastic event structure $\Sigma = \langle \mathcal{E}, \mathcal{R} \rangle$ is a sequence σ of rated events $(e_1, \lambda_1) \ldots (e_n, \lambda_n)$ with $e_i \in E$, $\lambda_i \in \mathbb{R}^+$, for $0 < i \leqslant n$ satisfying

1. $e_1 \ldots e_n \in T(\mathcal{E})$
2. $\forall\, i : \lambda_i = \mathcal{R}(e_i)$.

□

The set of stochastic event traces of simple stochastic event structure Σ is denoted $T_S(\Sigma)$. In a similar way as for the deterministic timed case (cf. Chapter 4) lposets can be defined from stochastic configurations. This is not considered further here.

8.2.2 A simple stochastic process algebra

Let the syntax of the language PA_S of simple finite stochastic behaviours be defined as follows:[1]

8.5. DEFINITION. (*Simple stochastic process algebra* PA_S)

$$B ::= \mathbf{0} \;\Big|\; (\lambda)\ a\,;B \;\Big|\; B+B \;\Big|\; B\,||_G\,B \;\Big|\; B[H] \;\Big|\; B\setminus G. \qquad \square$$

Like in the timed process algebra PA_T actions are considered to be atomic and to occur instantaneously. $(\lambda)\ a\,;B$ denotes a behaviour which may engage in a from a time period relative to the beginning of the system with an exponential distributed length (of rate λ) and after the occurrence of a behaves like B. λ specifies the rate of the exponential distribution of a *relative* delay of an action.

In the deterministic timing case a set of behaviours may synchronize on a common action as soon as all participants are ready to engage in this action. For example, in an expression like $(t)\ a\,;\mathbf{0}\,||_a\,(t')\ a\,;\mathbf{0}$ the resulting action a is enabled from time $max(t,t')$. In case the delay of actions (in fact, events) is determined by a stochastic variable, it seems natural—and a straightforward generalization of the deterministic time case—to let the enabling time of a synchronization being determined by the maximum of the stochastic variables that determine the local delay of this action. From basic probability theory [87] we know that the distribution of the maximum of two (or more) independent stochastic variables corresponds to the *product* of their distribution functions.

8.6. THEOREM. Let $U_1,\ldots,U_n$ $(n \geqslant 1)$ be independent stochastic variables where U_i has distribution F_{U_i}, and $W = Max\{\,U_1,\ldots,U_n\,\}$. Then the probability distribution function of W equals

$$F_W(x) = \prod_{i=1}^{n} F_{U_i}(x) \quad ,$$

and its probability density function

$$F'_W(x) = \sum_{i=1}^{n}\left(F'_{U_i}(x)\cdot \prod_{j=1,j\neq i}^{n} F_{U_j}(x)\right) \quad .$$

PROOF. Straightforward by induction on n. We only provide the proof for n=2.

$F_W(x)$

[1] For simplicity we do not consider the syntactical constructs $\surd$, >>, and [> here. Since we mainly introduce this algebra to compare with existing approaches which do not contain these constructs either, this restriction is convenient for our purposes.

$$
\begin{array}{ll}
= & \{ \text{ Definition A.1 } \} \\
 & Pr\{ W \leqslant x \} \\
= & \{ \text{ definition of } W \text{ } \} \\
 & Pr\{ max(U_1, U_2) \leqslant x \} \\
= & \{ \text{ calculus } \} \\
 & Pr\{ U_1 \leqslant x, U_2 \leqslant x \} \\
= & \{ \ U_1 \text{ and } U_2 \text{ are statistically independent } \} \\
 & Pr\{ U_1 \leqslant x \} \cdot Pr\{ U_2 \leqslant x \} \\
= & \{ \text{ Definition A.1 } \} \\
 & F_{U_1}(x) \cdot F_{U_2}(x) \quad .
\end{array}
$$

Obviously, $F'_W(x)$ equals $F'_{U_1}(x) \cdot F_{U_2}(x) + F_{U_1}(x) \cdot F'_{U_2}(x)$. □

Unfortunately, the product of two exponential distributions is *not* an exponential distribution (see also Example 8.21). Therefore, we take in this section a pragmatic approach by combining individual distributions in such a way that the resulting distribution of a synchronization action is again exponential. This is achieved by computing the rate of the resulting action from the individual rates of the components according to $\circledast : \mathbb{R}^+ \times \mathbb{R}^+ \longrightarrow \mathbb{R}^+$. E.g., action a in the composite behaviour $(\lambda)\ a\,;\mathbf{0} \,||_a\, (\mu)\ a\,;\mathbf{0}$ will have rate $\lambda \circledast \mu$. Different choices for $\circledast$ are possible. For an extensive discussion on these possibilities, their (stochastic) interpretation, and desired algebraic properties of $\circledast$ we refer to Götz [57] and Hillston [73].

We now provide a semantics of PA_S by defining a mapping $\mathcal{X}[\![\, B \,]\!]$ which associates a simple stochastic bundle event structure with each expression B of PA_S. $\mathcal{X}$ is an orthogonal extension of the mapping of PA to extended bundle event structures (cf. Chapter 2). Let Φ_S be a function associating to a stochastic behaviour B its corresponding non-stochastic behaviour $\Phi_S(B)$ by simple omitting the rates in B. In the rest of this section let $\mathcal{X}[\![\, B_i \,]\!] = \langle (E_i, \leadsto_i, \mapsto_i, l_i), \mathcal{R}_i \rangle$, for $i = 1, 2$, with $E_1 \cap E_2 = \varnothing$. We assume $\circledast$ to be commutative, associative and have an identity element, denoted $\mathbf{u}$. That is, for all $\lambda \in \mathbb{R}^+$ we have $\lambda \circledast \mathbf{u} = \mathbf{u} \circledast \lambda = \lambda$.

8.7. DEFINITION. (*Causality-based semantics of* PA_S)

$\mathcal{X}[\![\ ]\!]$ is defined recursively as follows:

$$
\begin{array}{rcl}
\mathcal{X}[\![\, \mathbf{0} \,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_S(\mathbf{0}) \,]\!], \varnothing \rangle \\
\mathcal{X}[\![\, (\lambda)\ a\,; B_1 \,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_S((\lambda)\ a\,; B_1) \,]\!], \mathcal{R}_1 \cup \{ (e_a, \lambda) \} \rangle \\
\mathcal{X}[\![\, B_1 + B_2 \,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_S(B_1 + B_2) \,]\!], \mathcal{R}_1 \cup \mathcal{R}_2 \rangle \\
\mathcal{X}[\![\, B_1 \setminus G \,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_S(B_1 \setminus G) \,]\!], \mathcal{R}_1 \rangle \\
\mathcal{X}[\![\, B_1[H] \,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_S(B_1[H]) \,]\!], \mathcal{R}_1 \rangle \\
\mathcal{X}[\![\, B_1 \,||_G\, B_2 \,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_S(B_1 \,||_G\, B_2) \,]\!], \mathcal{R} \rangle \text{ where} \\
\mathcal{R}((e_1, e_2)) & = & \mathcal{R}_1(e_1) \circledast \mathcal{R}_2(e_2) \text{ such that } \mathcal{R}_i(*) = \mathbf{u}.
\end{array}
$$

□

8.8. EXAMPLE. The definition of $\mathcal{X}$ is exemplified by providing the semantics of the following stochastic behaviours (cf. Figure 8.2):

$(a)\ B_1 \ = \ (\lambda_1)\ a\,;\ (\lambda_2)\ b\,;\ \mathbf{0}\,||_b\,(\lambda_3)\ c\,;\ (\lambda_4)\ b\,;\ \mathbf{0}$,
$(b)\ B_2 \ = \ (\mu_1)\ a\,;\ (\mu_2)\ b\,;\ \mathbf{0}\,||_b\,((\mathbf{u})\ b\,;\ \mathbf{0} + (\mu_3)\ d\,;\ \mathbf{0})$, and
$(c)\ B_1\,||_{\{a,b\}}\,B_2$.

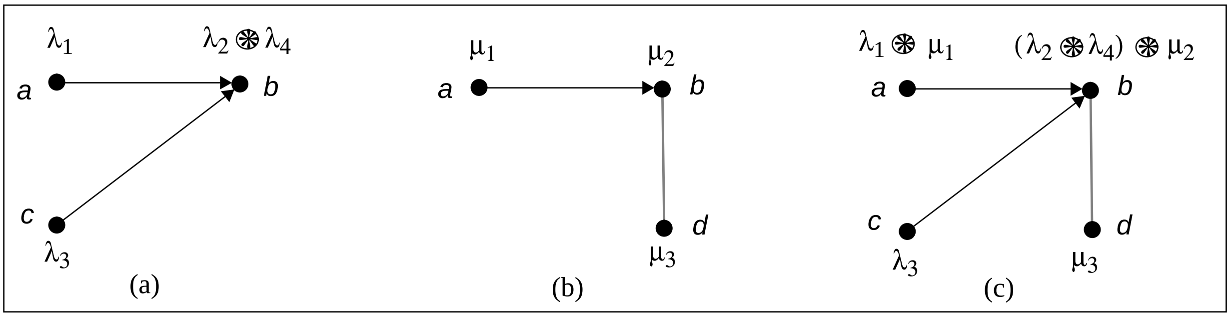


Figure 8.2: Examples of simple stochastic event structure semantics.

□

Actions with rate $\mathbf{u}$, the identity of $\circledast$, do not contribute to the resulting rate of a synchronization. That is, $(\mathbf{u})\ a\,;\ \mathbf{0}\,||_a\,(\lambda)\ a\,;\ \mathbf{0}$ results in action a with rate $\mathbf{u} \circledast \lambda = \lambda$. Such actions are referred to as *passive* and often occur in performance modelling to model service-like activities. For passive actions only one process determines the rate of synchronization while the other participating processes do not impose additional timing constraints.

We conclude this section by discussing *immediate* actions. In performance modelling actions that are irrelevant from a performance evaluation point of view are often considered to take place immediately thus not imposing any additional delay on the system's execution. This has led to the notion of immediate transitions in stochastic Petri nets [4], and similarly to the notion of immediate actions (i.e., actions with rate ∞) in stochastic process algebras (e.g., Bernardo *et al.* [14] and Götz [57]). In our model such actions can easily be incorporated by extending the definition of $\circledast$ such that $\lambda \circledast \infty = \infty \circledast \lambda = \infty$ for all $\lambda \in \mathbb{R}^+ \cup \{\infty\}$. That is, ∞ is a zero element of $\circledast$.

8.2.3 Event-based operational semantics for PA_S

Various stochastic extensions of process algebras are known from the literature [58, 68, 14, 71, 72, 30]. These formalisms have in common that they are based on an interleaving semantics (i.e., a stochastic extension of labelled transition systems) and that distribution functions are restricted to be exponential. The main difference among these stochastic process algebras is the way in which the rate of a synchronized action is computed (see also later on).

In order to compare our simple stochastic event structure model to these existing approaches and to investigate the 'compatibility' of our proposal with the standard semantics of PA

(provided in Chapter 1) we define an operational semantics for PA_S that corresponds to the noninterleaving semantics. The approach we follow is similar to the approach taken for the deterministic timing case (Chapter 5 of this thesis). Thus, we define a transition system in which we keep track of the occurrence of actions in an expression of PA_S. This results in a *stochastic* event transition system.

In order to define an event transition system each occurrence of an action-prefix is subscripted with an arbitrary but unique event occurrence identifier, denoted by a Greek letter. The transition relation $\longrightarrow\!\!\!\!\rightarrow$ is defined as the smallest relation closed under all inference rules defined in Table 8.1. $B \xrightarrow{(e,a,\lambda)}\!\!\!\!\rightarrow B'$ denotes that behaviour B can perform event e, labelled a with rate λ and evolve into B'.

$$\frac{}{(\lambda)\, a_\xi \,;\, B \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B}$$

$$\frac{B_1 \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B_1'}{B_1 + B_2 \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B_1'} \qquad \frac{B_2 \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B_2'}{B_1 + B_2 \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B_2'}$$

$$\frac{B_1 \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B_1'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,*),a,\lambda)}\!\!\!\!\rightarrow B_1' \,||_G\, B_2} \; (a \notin G) \qquad \frac{B_2 \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((*,\xi),a,\lambda)}\!\!\!\!\rightarrow B_1 \,||_G\, B_2'} \; (a \notin G)$$

$$\frac{B_1 \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B_1' \;\wedge\; B_2 \xrightarrow{(\psi,a,\mu)}\!\!\!\!\rightarrow B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,\psi),a,\lambda\circledast\mu)}\!\!\!\!\rightarrow B_1' \,||_G\, B_2'} \; (a \in G)$$

$$\frac{B \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B'}{B \setminus G \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B' \setminus G} \; (a \notin G) \qquad \frac{B \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B'}{B \setminus G \xrightarrow{(\xi,\tau,\lambda)}\!\!\!\!\rightarrow B' \setminus G} \; (a \in G)$$

$$\frac{B \xrightarrow{(\xi,a,\lambda)}\!\!\!\!\rightarrow B'}{B[H] \xrightarrow{(\xi,H(a),\lambda)}\!\!\!\!\rightarrow B'[H]}$$

Table 8.1: Event-based operational semantics for PA_S.

Using the transition relation $\longrightarrow\!\!\!\!\rightarrow$ the notion of (stochastic) event trace can be defined in the usual way. As the transition system induced by $\longrightarrow\!\!\!\!\rightarrow$ is deterministic, the transition system for B can be represented by its set of stochastic event traces $\mathcal{T}_S[\![\, B \,]\!]$. This set can be characterized in a denotational way, and subsequently proven to coincide with the set of stochastic event traces of the corresponding event structure $\mathcal{X}[\![\, B \,]\!]$. This proves the consistency between the operational semantics and denotational semantics in terms of event structures.

8.9. THEOREM. $\forall B \in \mathsf{PA}_S : T_S(\mathcal{X}[\![\, B \,]\!]) = \mathcal{T}_S[\![\, B \,]\!]$.

PROOF. In a similar way as for the deterministic timing case (see Chapter 5). $\square$

8.2.4 Related approaches

From the event transition system defined by $\longrightarrow\!\!\!\!\rightarrow$ we can easily obtain the standard inference rules for PA by omitting the rates and event identifiers. In addition, the transition rules strongly resemble the operational semantics of existing stochastic process algebras, and for various algebras we obtain identical rules when substituting the appropriate operator for $\circledast$. This provides adequacy for our simple stochastic causality-based model.

In one of the first stochastic process algebras, MTIPP (Markovian Timed Processes for Performance Evaluation) by Herzog *et al.* [58, 68], the rate of a synchronized action is simply the product of the rates of the components, thus $\lambda \circledast \mu = \lambda \cdot \mu$. For Bologna's variant (B-MPA) of Bernardo *et al.* [14] the resulting rate is the maximum of the individual rates under the condition that at least one of the participating behaviours must be passive with respect to the interaction, thus, $\lambda \circledast \mu = max(\lambda, \mu)$ given that $\lambda = \mathbf{u}$ or $\mu = \mathbf{u}$. In D-MPA of Buchholz [30] a somewhat different approach is taken—each action label a is assigned a fixed transition rate μ_a, and $(r)\ a\,;\, B$ $(r \in \mathbb{R}^+)$ denotes a behaviour that may engage in a where the time before a is performed is exponentially distributed with rate $r \cdot \mu_a$. When $(r_1)\ a$ and $(r_2)\ a$ synchronize the time before interaction a happens is distributed with rate $r_1 \cdot r_2 \cdot \mu_a$. Using $\circledast$ as product on r_i (rather than on rates) and assuming that μ_a is given, the same scheme can be obtained with the rules of Table 8.1.

Another prominent stochastic process algebra is PEPA (Performance Enhanced Process Algebra) developed by Hillston. In the initial proposal for PEPA [71] the expected delay (i.e., the reciprocal of the rate) of the interaction is assumed to be the sum of the expected duration of the action in each of the participants, i.e., $\lambda \circledast \mu = (\lambda \cdot \mu)/(\lambda + \mu)$. In the final proposal for PEPA [72] the rate of an interaction is computed by taking into account the total capacity of a behaviour to participate in actions with a certain label (the so-called apparent rate). Since apparent rates are based on the entire behaviour of a participant rather than solely on the (local) rate of an event this synchronization policy cannot be modelled using $\circledast$.

As noted before, desired algebraic properties of $\circledast$ are associativity, commutativity and the existence of an identity element. (Algebraically speaking, this means that $\langle \mathbb{R}^+, \circledast \rangle$ is a commutative, or Abelian, monoid.) For modelling immediate actions $\circledast$ should also have a zero element. Besides these properties [57, 73] require $\circledast$ to be distributive over the addition of rates in order to consider $(\lambda)\ a + (\mu)\ a$ and $(\lambda{+}\mu)\ a$ to be equivalent, also in the context of parallel composition (which leads to the distributivity). It is interesting to note that in our model rates are associated to events rather than to actions, and the two a actions in the choice expression above are modelled by distinct events. So, it seems that distributivity of $\circledast$ over $+$ is not a necessary requirement in our model unless distinct events are identified by some congruence relation.

8.3 Generalized stochastic event structures

The main benefit of the model of the previous section is that it is a rather simple extension of bundle event structures which corresponds quite closely to existing stochastic process algebras such as MTIPP [58], a preliminary version of PEPA [71], D-MPA [30], and B-MPA [14] (depending on the choice for $\circledast$). Unfortunately, for keeping the model within the domain of exponential distributions we were unable to let the stochastic variable that determines the delay of an interaction be the maximum of the individual stochastic variables, whilst this seems quite reasonable and would be a straightforward generalization of our deterministic timing model.

In addition, exponential distributions are a bit restrictive in performance modelling and there is a considerable need for more realistic (i.e., nonmemoryless) distributions. Especially in the analysis of high-speed communication systems or multi-media applications where the correlation between successive packet arrivals is no longer negligible and packets tend to have a constant length the usual Poisson arrivals and exponential packet lengths are no longer valid assumptions.

In this section we replace the deterministic times associated to bundles and events in our deterministic timing model (cf. Chapter 4) by stochastic variables having arbitrary distributions, and investigate what the required (algebraic) properties of such distributions are given that the treatment of synchronization is similar to the deterministic case.

8.3.1 The model

Distribution functions are added to bundle event structures in two ways. A distribution function associated with event e determines the time between the start of the system and the enabling of e, while a distribution function associated to bundle $X \mapsto e$ determines the relative time between the enabling of e and its causal predecessor in X.

The interpretation of bundle $\{ e_a \} \mapsto e_b$ decorated with distribution F is that if e_a has happened at a certain time t_a then the time at which e_b is enabled is determined by t_a+U where U is a stochastic variable with distribution F.

If more than one bundle points to an event the following interpretation is chosen. For instance, suppose $\{ e_a \} \mapsto e_c$ and $\{ e_b \} \mapsto e_c$ with distribution F and G, respectively. Now, if e_a (e_b) happens at t_a (t_b) then the time of enabling of e_c is determined by the stochastic variable $max(t_a+U, t_b+V)$, where U (V) has distribution F (G).

As a final example, consider $\{ e_a \} \mapsto e_b$ decorated with distribution F and e_b having distribution G. Using a similar reasoning as above, we infer that the stochastic variable $max(U, t_a+V)$ determines the time of enabling of e_b given that e_a happens at time t_a.

Let DF denote an arbitrary class of distribution functions.

8.10. DEFINITION. (*Stochastic event structure*)

A *stochastic* bundle event structure Σ is a triple $\langle \mathcal{E}, \mathcal{F}, \mathcal{G} \rangle$ with $\mathcal{E}$ an extended bundle event structure $(E, \rightsquigarrow, \mapsto, l)$, and $\mathcal{F} : E \longrightarrow \mathsf{DF}$ and $\mathcal{G} : \mapsto \longrightarrow \mathsf{DF}$, associating a distribution function of class DF to events and bundles, respectively. $\square$

We denote a bundle (X, e) with $\mathcal{G}((X, e)) = F$ by $X \overset{F}{\mapsto} e$. Event traces are considered as sequences of events where each event e_i is associated with a stochastic variable U_i that uniquely determines the minimal enabling time of event e_i. The stochastic variable U_i is determined by the distribution function associated with e_i (i.e., $\mathcal{F}(e_i)$), the distributions linked to all bundles pointing to e_i and the stochastic variables U_j of the causal predecessors of e_i in the trace (as these determine the time of occurrence of e_j).

8.11. DEFINITION. (*Random event trace*)

A *random event trace* of stochastic event structure $\Sigma = \langle \mathcal{E}, \mathcal{F}, \mathcal{G} \rangle$ is a sequence σ of events $(e_1, U_1) \dots (e_n, U_n)$ with $e_i \in E$, and U_i, for all $0 < i \leqslant n$, a stochastic variable with distribution function in class DF iff

1. $e_1 \dots e_n \in T(\mathcal{E})$, and
2. $\forall\, i : U_i = Max(\{\, U_{\mathcal{F}(e_i)} \,\} \cup V_i \cup W_i)$ where
   $V_i = \{\, U_G + U_j \mid \exists\, X : X \overset{G}{\mapsto} e_i \;\wedge\; X \cap \overline{[\sigma_i]} = \{\, e_j \,\}\,\}$ and
   $W_i = \{\, U_j \mid \exists\, e_j \in \overline{[\sigma_i]} : e_j \rightsquigarrow e_i \,\}$.

□

Notice the resemblance of this definition of with the definition of timed event trace in Chapter 4 (Definition 4.5). For distribution function F, U_F denotes the corresponding stochastic variable. In general it is not straightforward to obtain a closed formula for U_i since statistical independence of its constituents cannot always be guaranteed. The stochastic variable $\overline{U} = (U_1, \dots, U_n)$ spans an n-dimensional hyperspace and has joint distribution function

$$F_{\overline{U}}(\overline{x}) = \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} F'_{\overline{U}}(y_1, \dots, y_n)\, dy_n \dots dy_1.$$

8.12. EXAMPLE. Consider the stochastic event structures in Figure 8.3. The event distribution of event e_a is denoted F_a and is omitted in the figure for simplicity. For (a) legal traces are $(e_a, U_a)(e_b, U_b)$ and $(e_b, U_b)(e_a, U_a)$ with $U_a = U_{F_a}$ and $U_b = U_{F_b}$. Note that the stochastic variables are equal for both traces. For (b) $(e_a, U_a)(e_b, U_b)$ is a trace with $U_a = U_{F_a}$ and $U_b = max(U_{F_b}, U_G{+}U_a)$. Finally, for (c) $(e_a, U_a)(e_b, U_b)(e_c, U_c)$ is a trace with $U_a = U_{F_a}$, $U_b = U_{F_b}$ and $U_c = Max\{\, U_{F_c}, U_G{+}U_a, U_H{+}U_b \,\}$. □

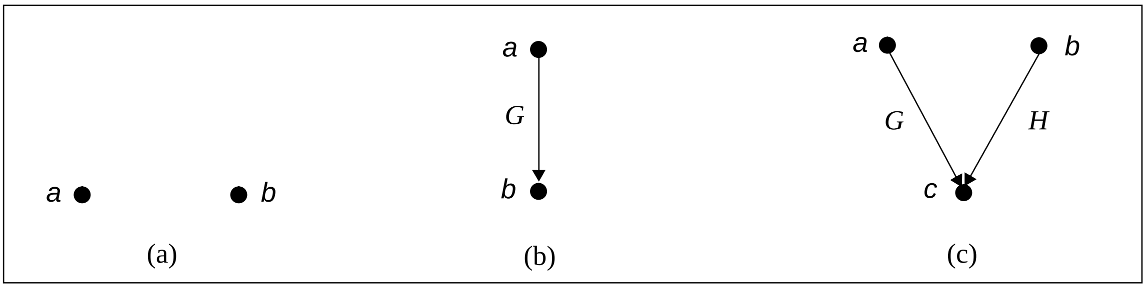


Figure 8.3: Some stochastic bundle event structures.

8.3.2 A generalized stochastic process algebra

In this section we use the model of the previous section as a semantical model for a generalized stochastic process algebra. The aim of this exercise is to investigate what the desired algebraic properties of distribution functions are. Let F be a distribution function in DF. The syntax of behaviours in PA_{GS} is now defined as follows:

8.13. DEFINITION. (*Generalized stochastic process algebra* PA_{GS})

$$B ::= \mathbf{0} \mid \surd \mid (F)\ a\,;B \mid B+B \mid B \gg B \mid B\,[> B \mid B\,||_G\,B \mid B[H] \mid B \setminus G. \qquad \square$$

This syntax is identical to the syntax of PA_T, the timed process algebra of Chapter 4, except that time annotations are replaced by distribution functions from DF.

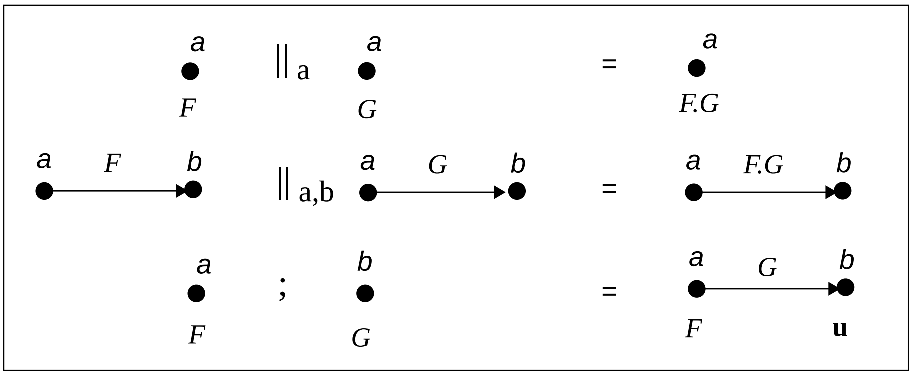


Figure 8.4: Examples of composing stochastic event structures.

In a similar way as for the exponential distribution case we define a mapping $\mathcal{E}_S[\![\, B \,]\!]$ which associates a stochastic bundle event structure to expression B. This provides us a causality-based semantics of PA_{GS}. Let us start by considering some examples (cf. Figure 8.4). In the upper picture we are faced with the question what the resulting distribution of a in $(F)\ a\,;\mathbf{0}\,||_a\,(G)\ a\,;\mathbf{0}$ will be. When we adopt the synchronization paradigm of the deterministic timed model $max(U_F, U_G)$ would determine the timing of a. This results in distribution $F \cdot G$. A similar reasoning applies to the next picture (where, for simplicity, irrelevant distributions are omitted). Finally, in the lower picture the main issue is what the resulting distribution, H say, of b will be. In the deterministic timed case b would be associated time 0, the unit element of max. Hence, in the stochastic case $H = \mathbf{u}$, the unit element of $\cdot$. This motivates that we require the class DF of distribution functions to be closed under product $(\cdot)$ and to have an identity element $\mathbf{u}$ for this operation. Recall that the product of distributions corresponds to the maximum of their stochastic variables under the assumption of statistical independence.

In the following definition let $\mathcal{E}_S[\![\, B_i \,]\!] = \Sigma_i = \langle (E_i, \rightsquigarrow_i, \mapsto_i, l_i), \mathcal{F}_i, \mathcal{G}_i \rangle$, for $i = 1, 2$, with $E_1 \cap E_2 = \varnothing$. We assume that the stochastic variables corresponding to the bundle and event distributions in Σ_1 and Σ_2 are statistically independent. The positive events of Σ are those events that have a distribution function different from $\mathbf{u}$, i.e., $pos(\Sigma) = \{\, e \in E \mid \mathcal{F}(e) \neq \mathbf{u} \,\}$. Let $pin(\Sigma) = pos(\Sigma) \cup init(\Sigma)$. Let E_U denote the universe of events.

8.14. Definition. (*Semantics of* $\mathbf{0}$, $\surd$, *and* $(F)\ a\,;$)

$$\begin{array}{rcl} \mathcal{E}_S[\![\,\mathbf{0}\,]\!] & \triangleq & \langle \mathcal{E}'[\![\,\Phi_S(\mathbf{0})\,]\!], \varnothing, \varnothing\rangle \\ \mathcal{E}_S[\![\,\surd\,]\!] & \triangleq & \langle \mathcal{E}'[\![\,\Phi_S(\surd)\,]\!], \{\,(e_\delta, \mathbf{u})\,\}, \varnothing\rangle \\ \mathcal{E}_S[\![\,(F)\ a\,;\,B_1\,]\!] & \triangleq & \langle (E, \rightsquigarrow_1, \mapsto, l_1 \cup \{\,(e_a, a)\,\}), \mathcal{F}, \mathcal{G}\rangle \text{ where} \\ E & = & E_1 \cup \{\,e_a\,\} \text{ for some } e_a \in E_U \setminus E_1 \\ \mapsto & = & \mapsto_1\ \cup (\{\,\{\,e_a\,\}\,\} \times \mathit{pin}(\Sigma_1)) \\ \mathcal{F} & = & \{\,(e_a, F)\,\} \cup (E_1 \times \{\,\mathbf{u}\,\}) \\ \mathcal{G} & = & \mathcal{G}_1 \cup \{\,((\{\,e_a\,\}, e), \mathcal{F}_1(e)) \mid e \in \mathit{pin}(\Sigma_1)\,\}. \end{array}$$

□

The semantics of $\mathbf{0}$ and $\surd$ is self-explanatory. In $\mathcal{E}_S[\![\,(F)\ a\,;\,B_1\,]\!]$ a bundle is introduced from a new event e_a (labelled a) to all initial events of Σ_1 and, in addition, to all events in Σ_1 that have a distribution function different from $\mathbf{u}$. The distribution of these events is now relative to e_a, so each bundle $\{\,e_a\,\} \mapsto e$ is associated with a distribution $\mathcal{F}_1(e)$, and the distribution $\mathcal{F}(e)$ is made $\mathbf{u}$. The distribution $\mathcal{F}(e_a)$ becomes F. In the untimed and exponential case (cf. Chapter 2 and Definition 8.7) it suffices to only introduce bundles from e_a to the initial events of Σ_1. Introducing bundles from e_a to all events in $\mathit{pin}(\Sigma_1)$ is, however, semantically equivalent (as shown in Chapter 2) and is used here only to make distributions of events relative to e_a. To exemplify this, Figure 8.5 depicts (a) $\mathcal{E}_S[\![\,B_1\,]\!]$, and (b) $\mathcal{E}_S[\![\,(F)\ a\,;\,B_1\,]\!]$.

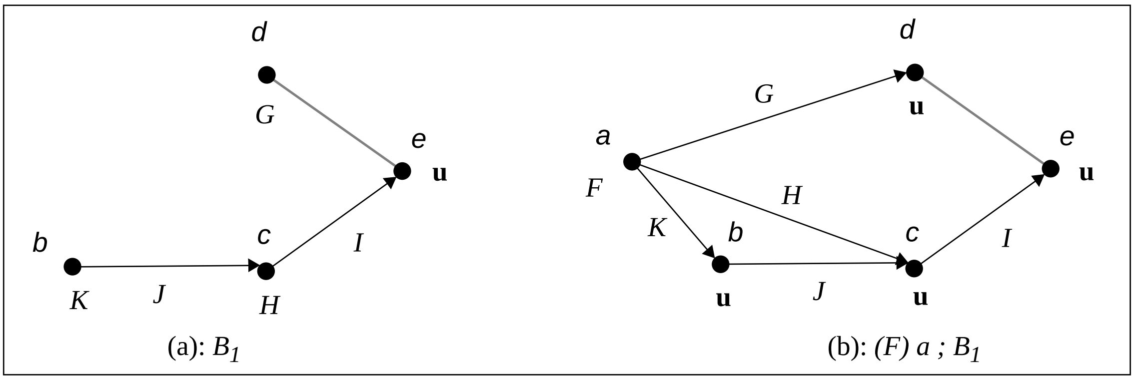


Figure 8.5: Example of stochastic action prefix.

8.15. Definition. (*Semantics of* $\backslash$, $[\,]$, $+$, $>>$ *and* $[>$)

$$\begin{array}{rcl} \mathcal{E}_S[\![\,B_1\ \mathsf{op}\ B_2\,]\!] & \triangleq & \langle \mathcal{E}'[\![\,\Phi_S(B_1\ \mathsf{op}\ B_2)\,]\!], \mathcal{F}_1 \cup \mathcal{F}_2, \mathcal{G}_1 \cup \mathcal{G}_2\rangle,\ \mathsf{op}\ \in \{\,+, [>\,\} \\ \mathcal{E}_S[\![\,\mathsf{op}\ B_1\,]\!] & \triangleq & \langle \mathcal{E}'[\![\,\Phi_S(\mathsf{op}\ B_1)\,]\!], \mathcal{F}_1, \mathcal{G}_1\rangle \text{ for } \mathsf{op}\ \in \{\,\backslash, [\,]\,\} \\ \mathcal{E}_S[\![\,B_1 >> B_2\,]\!] & \triangleq & \langle (E_1 \cup E_2, \rightsquigarrow, \mapsto, l), \mathcal{F}, \mathcal{G}\rangle \text{ where} \\ \rightsquigarrow & = & \rightsquigarrow_1\ \cup\ \rightsquigarrow_2\ \cup \{\,(e, e') \mid e, e' \in \mathit{exit}(\Sigma_1)\ \wedge\ e \neq e'\,\} \\ \mapsto & = & \mapsto_1\ \cup\ \mapsto_2\ \cup (\{\,\mathit{exit}(\Sigma_1)\,\} \times \mathit{pin}(\Sigma_2)) \\ l & = & ((l_1 \cup l_2) \setminus (\mathit{exit}(\Sigma_1) \times \{\,\delta\,\})) \cup (\mathit{exit}(\Sigma_1) \times \{\,\tau\,\}) \\ \mathcal{F} & = & \mathcal{F}_1 \cup (E_2 \times \{\,\mathbf{u}\,\}) \\ \mathcal{G} & = & \mathcal{G}_1 \cup \mathcal{G}_2 \cup \{\,((\mathit{exit}(\Sigma_1), e), \mathcal{F}_2(e)) \mid e \in \mathit{pin}(\Sigma_2)\,\}. \end{array}$$

□

Finally, we explain the semantics of the parallel composition operator. Events of $\mathcal{E}_S[\![\, B_1 \,||_G\, B_2 \,]\!]$ are constructed in the same way as in Definition 8.7. The distribution associated with a bundle is equal to the product of the distribution functions associated with the bundles we get by projecting on the i-th components (i=1, 2) of the events in the bundle, if this projection yields a bundle in $\mathcal{E}_S[\![\, B_i \,]\!]$. The distribution of an event is the product of the distributions of its components that are different from $*$.

8.16. Definition. (*Semantics of* $||_G$)

$$
\begin{array}{rcl}
\mathcal{E}_S[\![\, B_1 \,||_G\, B_2 \,]\!] & \triangleq & \langle \mathcal{E}'[\![\, \Phi_S(B_1 \,||_G\, B_2) \,]\!], \mathcal{F}, \mathcal{G}\rangle \text{ where} \\
\mathcal{F}((e_1, e_2)) & = & \mathcal{F}_1(e_1) \cdot \mathcal{F}_2(e_2) \text{ with } \mathcal{F}_i(*) = \mathbf{u}. \\
\mathcal{G}((X, (e_1, e_2))) & = & \mathcal{G}_1((pr_1(X), e_1)) \cdot \mathcal{G}_2((pr_2(X), e_2)) \\
 & & \quad \text{with } \mathcal{G}_i((\varnothing, e_i)) = \mathbf{u}, \text{ for } i{=}1,2.
\end{array}
$$

□

8.3.3 PH-distributions

We conclude that the desired properties of the class of distribution functions that is of interest to us are that it should be closed under product and have an identity element for product. An interesting class of distribution functions that satisfy these constraints are the *phase-type (PH-)* distributions. PH-distributions can be considered as matrix generalizations of exponential distributions and are well-suited for numerical computation. They are used in many probabilistic models that have matrix-geometric solutions, have a richly developed theory due to Neuts [109, 110], and include frequently used distributions such as hyper- and hypo-exponential, Erlang, and Cox distributions.

Intuitively, a PH-distribution is characterized by the time until absorption in a finite-state continuous-time Markov process with a single absorbing state[2]. Consider a continuous-time Markov chain (cf. Figure 8.6) with transient states $\{\,1, \ldots, m\,\}$ and absorbing state m+1, initial probability vector $[\underline{\alpha}, \alpha_{m+1}]$ with $\underline{\alpha}\underline{1} + \alpha_{m+1} = 1$, and (infinitesimal) generator matrix

$$
\mathbf{Q} = \left[\begin{array}{cc} \mathbf{T} & \underline{T^0} \\ 0 & 0 \end{array} \right] \quad ,
$$

where $\mathbf{T}$ is a square matrix of order m such that $\mathbf{T}(i,i) < 0$ and $\mathbf{T}(i,j) \geqslant 0\,(i \neq j)$. The row sums of $\mathbf{Q}$ equal zero, i.e., $\mathbf{T}\underline{1} + \underline{T^0} = \underline{0}$.

$\mathbf{T}(i,j)$ $(i \neq j)$ can be interpreted as the rate at which the current state changes from transient state i to transient state j. Stated otherwise, starting from state i it takes an exponentially distributed time with mean $1/\mathbf{T}(i,j)$ to reach state j. $\underline{T^0}(i)$ is the rate at which the system can move from transient state i to the absorbing state, state m+1. $-\mathbf{T}(i,i)$ is the total rate of departure from state i, or, equivalently, the residence time in state i is exponentially distributed with rate $-1/\mathbf{T}(i,i)$. In general, the transition rates may depend on the time at

[2]Requiring a single absorbing state is not a severe restriction as Markov processes with more than one such state can easily be converted into a Markov process with a single absorbing state.

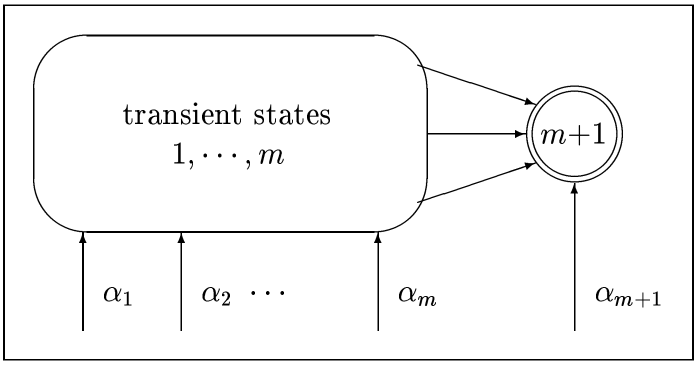


Figure 8.6: Schematic view of a PH-distribution.

which a system is considered. In this dissertation we confine ourselves to Markov chains whose behaviour is invariant to time-shifts. That is, at any time the rate to go from one state to another is the same. Such processes are often referred to as *time-homogeneous* Markov chains.

The probability distribution $F(x)$ of the time until absorption in state $m+1$ is now given by [3]

$$F(x) = 1 - \underline{\alpha} \cdot e^{\mathbf{T}x} \cdot \underline{1} \quad ,$$

for $x \geqslant 0$, and $F(x) = 0$, for $x < 0$. The pair $(\underline{\alpha}, \mathbf{T})$ is called a *representation* of F. The corresponding probability density function equals

$$F'(x) = \underline{\alpha} \cdot e^{\mathbf{T}x} \cdot \underline{T^0} \quad ,$$

for $x \geqslant 0$, and $F'(x) = 0$, for $x < 0$. The moments μ_i of $F(x)$ are finite and given by

$$\mu_i = (-1)^i \cdot i! \cdot (\underline{\alpha} \cdot \mathbf{T}^{-i} \cdot \underline{1}) \text{ for } i = 1, 2, \ldots \quad .$$

The first moment of a stochastic variable corresponds to its expectation, and the difference between the second moment and the square of the first moment corresponds to its variance.

Note the resemblance of the expressions for $F(x)$, $F'(x)$ and μ_i to the corresponding expressions for exponential distributions. In fact, for m=1 we obtain the results for regular exponential distribution. PH-distributions can thus be considered as *matrix generalizations* of the exponential distributions, which makes them suitable for numeric computations.

8.17. DEFINITION. (*Phase-type distribution*)

A continuous distribution function F on $[0, \infty)$ is called of *phase-type* (*PH*-distribution) iff it is the distribution of time to absorption in a continuous-time Markov chain as defined above. □

8.18. EXAMPLE. Example PH-distributions are the exponential, Erlang, hyper- and hypo-exponential, and Coxian distributions. Important to note is that these well-known (PH-type)

[3]For square matrix $\mathbf{T}$ of order m, $e^{\mathbf{T}x}$ is defined by $e^{\mathbf{T}x} = \mathbf{I}_m + \mathbf{T}x + \mathbf{T}^2\frac{x^2}{2!} + \mathbf{T}^3\frac{x^3}{3!} + \ldots$, where $\mathbf{I}_m$ denotes the identity matrix of order m and $\mathbf{T}^k\frac{x^k}{k!}$ is matrix $\mathbf{T}^k$ with each element multiplied by $\frac{x^k}{k!}$.

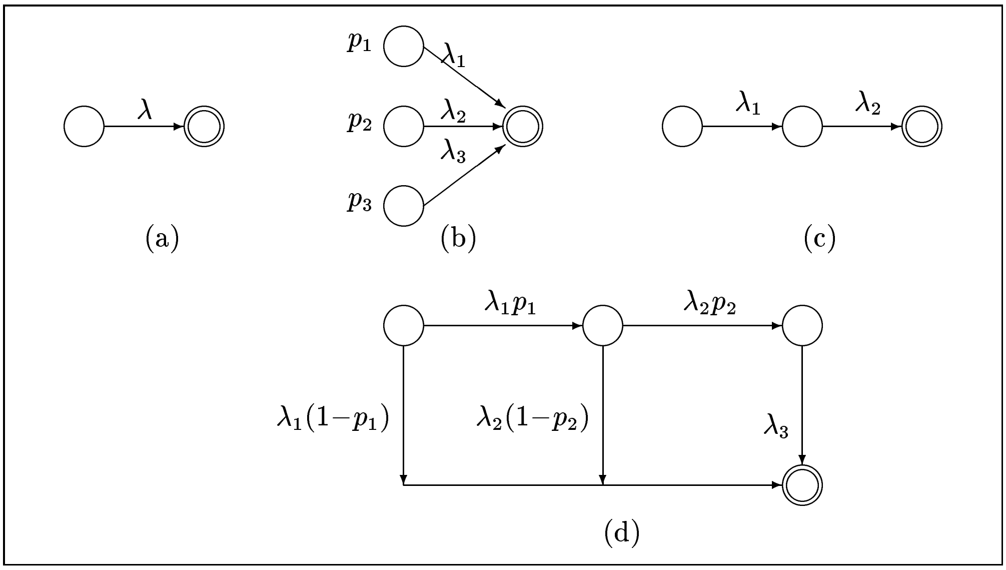


Figure 8.7: Some example PH-distributions.

distributions are acyclic while the definition of PH-type distributions also allows for cyclic Markov chains. Figure 8.7 illustrates an (a) exponential distribution with rate λ, (b) a 3-stage hyper-exponential distribution with rates λ_i, for i=1, 2, 3 (c) a 2-stage hypo-exponential distribution with rates λ_i, for i=1, 2, and (d) a 3-phase Coxian distribution. Representations of (b) and (d) are $\underline{\alpha}_{(b)} = [p_1, p_2, p_3]$ with $p_1+p_2+p_3 = 1$, $\underline{\alpha}_{(d)} = [1, 0, 0]$, and

$$\mathbf{T}_{(b)} = \begin{bmatrix} -\lambda_1 & 0 & 0 \\ 0 & -\lambda_2 & 0 \\ 0 & 0 & -\lambda_3 \end{bmatrix}, \qquad \mathbf{T}_{(d)} = \begin{bmatrix} -\lambda_1 & \lambda_1 \cdot p_1 & 0 \\ 0 & -\lambda_2 & \lambda_2 \cdot p_2 \\ 0 & 0 & -\lambda_3 \end{bmatrix}.$$

□

If U and V are statistically independent stochastic variables with PH-distributions G and H respectively, then the distribution F of $W = max(U, V)$ is equal to the product of G and H and is again a PH-distribution. The product of two PH-distributions is calculated as follows.

8.19. Theorem. Let PH-distributions G, H have representations $(\underline{\alpha}, \mathbf{T})$ and $(\underline{\beta}, \mathbf{S})$ of orders m and n, respectively. Then $F(x) = G(x) \cdot H(x)$ is a PH-distribution with representation $(\underline{\gamma}, \mathbf{L})$ of order $m \cdot n + m + n$ given by

$$\begin{aligned} \underline{\gamma} &= [\underline{\alpha} \otimes \underline{\beta}, \beta_{n+1}\underline{\alpha}, \alpha_{m+1}\underline{\beta}] \text{ and} \\ \mathbf{L} &= \begin{bmatrix} \mathbf{T} \otimes \mathbf{I}_n + \mathbf{I}_m \otimes \mathbf{S} & \mathbf{I}_m \otimes \underline{S^0} & \underline{T^0} \otimes \mathbf{I}_n \\ 0 & \mathbf{T} & 0 \\ 0 & 0 & \mathbf{S} \end{bmatrix}. \end{aligned}$$

Proof. See Neuts [109, Chapter 2]. □

$\otimes$ denotes the *tensor* (or Kronecker) *product* and is defined below. Note that $\mathbf{T} \otimes \mathbf{I}_n + \mathbf{I}_m \otimes \mathbf{S}$ is sometimes also referred to as the *tensor sum* of $\mathbf{T}$ and $\mathbf{S}$, denoted $\mathbf{T} \oplus \mathbf{S}$. $\mathbf{T} \oplus \mathbf{S}$ represents the generator matrix of a Markov process which is the Cartesian product of the Markov processes represented by $\mathbf{T}$ and $\mathbf{S}$. Tensor algebra is extensively discussed in Davio [38]. The PH-distribution consisting only of the absorbing state is the identity under product.

8.20. DEFINITION. (*Tensor product*)

The tensor (or Kronecker) product of two matrices $\mathbf{A}$ and $\mathbf{B}$ of orders $r_1 \times c_1$ and $r_2 \times c_2$, respectively, is defined as $\mathbf{C} = \mathbf{A} \otimes \mathbf{B}$ with $\mathbf{C}$ of order $r_1 r_2 \times c_1 c_2$ and

$$\mathbf{C}((i_1-1)r_2 + i_2, (j_1-1)c_2 + j_2) = \mathbf{A}(i_1, j_1) \cdot \mathbf{B}(i_2, j_2) \quad ,$$

where $0 < i_k \leqslant r_k$, $0 < j_k \leqslant c_k$ for $k{=}1,2$. □

The resulting matrix $\mathbf{C}$ can be considered to consist of $r_1 c_1$ blocks each having dimension $r_2 \times c_2$, that is, the dimension of $\mathbf{B}$:

$$\mathbf{C} = \begin{bmatrix} \mathbf{A}(1,1) \cdot \mathbf{B} & \mathbf{A}(1,2) \cdot \mathbf{B} & \dots & \mathbf{A}(1,c_1) \cdot \mathbf{B} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \mathbf{A}(r_1,1) \cdot \mathbf{B} & \mathbf{A}(r_1,2) \cdot \mathbf{B} & \dots & \mathbf{A}(r_1,c_1) \cdot \mathbf{B} \end{bmatrix} \quad .$$

The maximum of two PH-distributions is exemplified in the following example.

8.21. EXAMPLE. Exponential distributions G and H with rates λ and μ have representations $([1],[-\lambda])$ and $([1],[-\mu])$, respectively. The maximum F of these distributions has representation $(\underline{\gamma}, \mathbf{L})$ with $\underline{\gamma} = [1,0,0]$ and

$$\mathbf{L} = \begin{bmatrix} -(\lambda+\mu) & \mu & \lambda \\ 0 & -\lambda & 0 \\ 0 & 0 & -\mu \end{bmatrix} \quad .$$

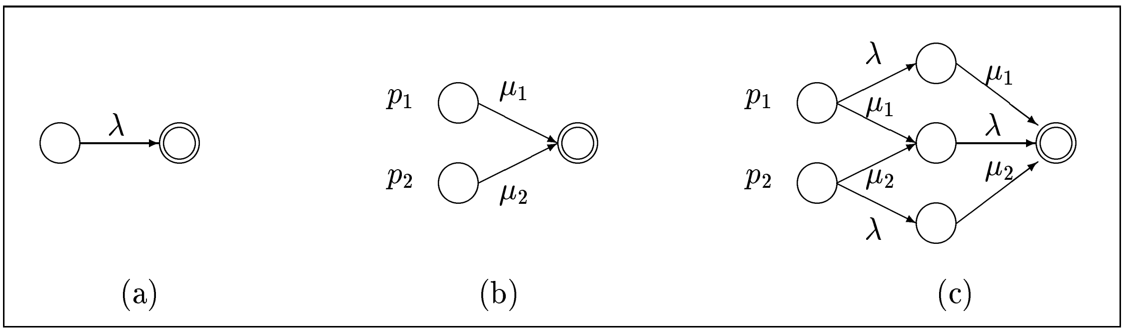


Figure 8.8: Maximum of a 1- and 2-stage hyper-exponential distribution.

As a second example let G be an exponential distribution with rate λ and H a 2-stage hyper-exponential distribution with rates μ_1 and μ_2, and initial probabilities p_1, p_2 with $p_1{+}p_2 = 1$

(cf. Figure 8.8(a) and (b)). The maximum F has representation $(\underline{\gamma}, \mathbf{L})$ with $\underline{\gamma} = [p_1, p_2, 0, 0, 0]$ and

$$\mathbf{L} = \begin{bmatrix} -(\lambda+\mu_1) & 0 & \mu_1 & \lambda & 0 \\ 0 & -(\lambda+\mu_2) & \mu_2 & 0 & \lambda \\ 0 & 0 & -\lambda & 0 & 0 \\ 0 & 0 & 0 & -\mu_1 & 0 \\ 0 & 0 & 0 & 0 & -\mu_2 \end{bmatrix}.$$

The corresponding Markov process is depicted in Figure 8.8(c). □

We conclude the exposition on PH-distributions by an observation. When considering Markov chains as ordinary finite state automata where transitions are labeled with rates, computing the product of two PH-distributions boils down to computing the product automaton of the constituent automata (cf. Figure 8.8). From the work of Plateau & Fourneau [119] it is known that the product chain of two continuous-time Markov chains with generator matrices $\mathbf{Q}$ and $\mathbf{R}$ has generator matrix $\mathbf{Q} \oplus \mathbf{R}$. This means that the product of two PH-distributions G and H with generator matrices $\mathbf{Q}$ and $\mathbf{R}$, respectively, is equal to F with generator matrix $\mathbf{Q} \oplus \mathbf{R}$. This is a much simpler characterization than given in Theorem 8.19.

8.4 Concluding remarks

In this chapter we have made an investigation of stochastic extensions of a process algebra in a causality-based setting. We presented a simple event structure model restricted to exponential distributions and a more general one involving PH-distributions. The simple semantic model is shown to be compatible with the standard operational semantics of (ordinary) process algebras like LOTOS and CSP and to closely resemble existing stochastic extensions of interleaved models like MTIPP, B-MPA, D-MPA and a preliminary version of PEPA.

The model involving PH-distributions evolved from a rather straightforward generalization of the deterministic timed model of Chapter 4. This results in associating distributions to events and bundles. Similar to the timed case it can be proven that a model with bundle distributions only suffices in case all initial actions of a specification have distribution $\mathbf{u}$, and all occurring parallel compositions satisfy the constraint that argument behaviours are able to participate in initial synchronization actions.

Another interesting class of distribution functions that satisfies our constraints is introduced by Sahner & Trivedi [131]. Here, the product of distribution functions of 'exponential polynomial form'

$$F(x) = \sum_i a_i \cdot x^{k_i} \cdot e^{b_i x} \text{ for } x \geqslant 0.$$

for k_i a natural and a_i, b_i real or complex numbers, is used to model the concurrent execution of groups of tasks. Cox, exponential, Erlang, and mixtures of exponential distributions also fall into this class of distributions. The applicability of such distributions in the context of our work is for further study.

To our knowledge only a few process algebras exist supporting a wider class of distribution functions than exponential ones. Ajmone Marsan *et al.* [3] define a stochastic extension of LOTOS in which random variables with arbitrary distribution functions specify the time lapse between actions. Once an action becomes enabled an experiment is carried out, the outcome of which represents the actual delay of the action. The main limitation of this proposal is that all stochastic timing constraints must be specified at 'top level', thus reducing compositionality and avoiding the issue of how to combine local distribution functions in case of synchronization. Götz *et al.* [59] discuss a generalization of MTIPP which supports arbitrary distribution functions. In order to associate the appropriate distribution function to actions in the interleaved semantic model, they introduce the notion of 'start references'. Such references are used to keep track of residual lifetimes of stochastic variables. In our model a similar notion is not needed, and general distributions could be incorporated in a more natural way. In the thesis of Rettelbach [128] a variant of MTIPP is discussed that allows for Erlang distributions. Here a special invisible action is used in the operational semantics to let the Erlang distribution move from one phase to another.

Though this chapter provides the first basic ingredients to study the (semi-) automated development of performance models out of system specifications in a causality-based setting, there are a number of issues to be settled. To mention a few, we did not yet address the issue of how to obtain a performance model from an event structure representation while exploiting the explicit parallelism present in the semantics. Some examples of how this could be done starting from an event structure with deterministic times and probabilistic choices can be found in Chapter 9. It has to be investigated how this approach carries over to the stochastic case.

A comparison with Petri nets is also considered to be useful. The relationship of bundle event structures with Petri nets has been studied by Boudol & Castellani [25] and it would be interesting to extend this study to (nonexponential) stochastic Petri nets. A problem here is that there is currently a lot of research going on in the field of nonexponential stochastic Petri nets and there is no consensus yet on the incorporation of general distributions into nets (see, for instance, Trivedi *et al.* [143]).

9 The probability module

This chapter presents a probabilistic variant of extended bundle event structures, in which internal events (i.e., events labeled τ) can be assigned a fixed probability. In this way, a causality-based model is obtained that allows for the specification of (internal) probabilistic behaviour. For probabilistic event structures the notion of cluster, a set of mutually conflicting internal events such that the sum of the probabilities associated to these events is 1, is defined. A cluster corresponds to an independent stochastic experiment. A probabilistic process algebra PA_P is introduced and assigned a causality-based and corresponding event-based operational semantics. The integration of the probabilistic model with the deterministic timed model (of Chapters 4 and 7) is briefly discussed. By means of example it is shown how to obtain a performance model (i.e., a discrete-time semi-Markov chain) from a timed probabilistic event structure.

9.1 Introduction

It is widely recognized that the behaviour of systems cannot be modelled adequately by only providing a means for describing the possible orderings of the execution of actions; issues like time and probability play an important rôle as well. In this chapter we equip extended bundle event structures with a notion of *probability*. In this way we facilitate the specification of reliability issues; quantification of concerns like the possibility that an unreliable communication medium loses or garbles a message, or the possibility that a system component exhibits some faulty behaviour now becomes possible.

The aim of this chapter is to investigate how probabilities can be introduced in a causality-based framework in a simple though practically useful way. The basic idea is to use probabilities to model (discrete) *stochastic experiments* that are (statistically) independent from the context in which they are considered. In order to facilitate this, some events are equipped with probabilities—we will call such events *probabilistic events*—and these events are required to be internal, i.e., labelled τ. A probabilistic event models an outcome of a stochastic experiment. Since a realization of an experiment usually has a single outcome, we require all probabilistic events that constitute the range of outcomes to be mutually in conflict. Such a group of events will be called a *cluster*.

Since all probabilistic events are internal, their probability of appearance can be determined without the need for conditioning probabilities on the possible behaviour of the environment. This is a simplifying assumption. We believe that still an interesting model remains, because there are lots of applications for which the description of *internal probabilistic behaviour* suffices. Typically the environment has no control over probabilistic phenomena one often encounters in practice: for instance, the fact that a system component spontaneously fails (like garbling a message) is usually due to some internal misbehaviour completely out of the environment's control [122]. There exist various probabilistic variants of formal models that do allow the resolution of experiments to be determined by the environment. This leads to more complicated models since probabilities must be adjusted depending on the environment in which they are considered. In addition, it seems not clear (yet) how the environment will influence the probabilistic behaviour of systems; different perspectives can be taken which result in different probabilistic models. An overview and classification of such models is provided by Van Glabbeek *et al.* [53].

The process algebra PA of Chapter 1 is enriched with a *probabilistic choice* operator, denoted $+_p$, where $B_1 +_p B_2$ denotes a behaviour that nondeterministically behaves like B_1 (with probability p), or like B_2 (with probability $1-p$), under the condition that this choice can be made autonomously, i.e., without interference of the environment. The fact that this choice can be made without participation of the environment is met by imposing some appropriate syntactical constraints. We investigate the use of the probabilistic causality-based model for providing a denotational semantics for this process algebra, called PA_P. Like for the timed, urgent, and (simple) stochastic case a consistent event-based operational semantics for PA_P is presented.

To our knowledge this constitutes the first attempt towards enhancing a partial-order model with probabilistic information. Current probabilistic (asynchronous) process algebras all use probabilistic extensions of labelled transition systems as an underlying semantical model. It is quite common to distinguish between probabilistic and nonprobabilistic transitions in these models. The main problem with this approach is the intertwining of these types of transitions. That is to say, it is not clear what the intended meaning is of a probability attached to a transition in the presence of a competitive nonprobabilistic transition. Typical behaviours that cause such situations are combinations of parallel composition and probabilistic choice, as in

$$(\tau\,;\,B_1 +_p \tau\,;\,B_2)\,|||\,a\,;\,B_3 \quad .$$

The fact that there is one global state in which either a or one of the two probabilistic alternatives can happen makes it difficult to interpret p as the probability that B_1 will be chosen. There have been several solutions proposed for this problem, some of which we will discuss later on in this chapter, but most of them loose the property of backwards compatibility with the nonprobabilistic semantics. We hope to show in this chapter that a causality-based model, which has no direct notion of global state, does not has these problems.

This chapter is further organized as follows. Section 9.2 introduces the notion of cluster and probabilistic event structure and carries notions like event trace, remainder and configuration over to a probabilistic setting. Section 9.3 presents the probabilistic process algebra PA_P; the syntactical constraints of the formalism are introduced and justified, and a causality-based denotational semantics and event-based operational semantics for this formalism are presented. Section 9.4 discusses a possible way in which the probabilistic and (simple) timed, urgent models of Chapters 4 and 6 can be integrated. This integration is used in Section 9.5 to show by means of example how performance models, in particular discrete-time semi-Markov chains, can be obtained from timed probabilistic event structures. Section 9.6 puts our work and results in the context of several other proposals for probabilistic process algebras and addresses options for further work. Finally, Section 9.7 summarizes the technical results of this chapter.

9.2 Probabilistic event structures

This section deals with probabilistic event structures. Section 9.2.1 introduces the basic ideas and the notion of probabilistic event structure. The status of such event structure after the execution of a sequence of events is presented in Section 9.2.2. Section 9.2.3 shows how probabilities can be calculated for sets of executions of probabilistic event structures.

9.2.1 What are probabilistic event structures?

The basic idea is to incorporate fixed probabilities in event structures by associating probabilities with events. Suppose we have an event e and we decorate this event with probability p, $p \in (0,1)$, that is $0 < p < 1$. The intuitive interpretation is that e happens with likelihood p provided that it is enabled. Thus, p is a *conditional* probability.

A group of events, each event having a fixed probability, intends to model an *independent* stochastic experiment, that is, the probability assigned to an event is independent from its context. An experiment consists of a set of possible outcomes. Each outcome has associated a real number which represents the probability of its occurrence when the experiment is carried out. Each realization of the experiment has precisely one outcome.

In order to model stochastic experiments, events are grouped into *clusters* of mutually conflicting events.

9.1. DEFINITION. (*Cluster*)

For event structure $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$, set $Q \subseteq E$ is a *cluster* of $\mathcal{E}$, iff

1. $\mid Q \mid > 1$
2. $\forall\, e \in Q : l(e) = \tau$
3. $\forall\, e, e' \in Q : e \neq e' \;\Rightarrow\; e \,\#\, e'$
4. $\forall\, e \in Q, e' \in E : e \rightsquigarrow e' \;\Rightarrow\; e' \in Q$
5. $\forall\, e, e' \in Q, X \subseteq E : X \mapsto e \;\Rightarrow\; X \mapsto e'$.

□

The first constraint requires a cluster to consist of at least two events; this is convenient for technical reasons and poses no real practical constraint. In order to guarantee that stochastic experiments represented by clusters are indeed independent from their context we require all events in a cluster to be internal (i.e., labelled τ). In this way we are sure that such events are not subject of interaction anymore, which would make their probability dependent on the context in which they will be embedded. According to the third constraint events in a cluster mutually exclude each other such that only one event (i.e., the outcome of the experiment) can happen. In addition we require that events in a cluster are not in conflict with events outside the cluster; allowing such conflicts would destroy the interpretation that an event probability represents the likelihood that this event happens (once enabled). This is stated by the fourth constraint. Finally, all events in a cluster must be pointed to by the same set of bundles. Together with the fourth constraint this guarantees that if an event in a cluster is enabled all events in this cluster are enabled.

A probabilistic event structure is an event structure in which some events are assigned a probability. We assume a (partial) mapping π that decorates an event with a probability in $(0,1)$. The interpretation is that an event e with $\pi(e) = p$ happens with probability p once it is enabled.

9.2. Definition. (*Probabilistic event structure*)

A *probabilistic event structure* is a tuple $\langle \mathcal{E}, \pi \rangle$ with

- $\mathcal{E}$, an extended bundle event structure $(E, \leadsto, \mapsto, l)$
- $\pi : E \longrightarrow_p (0,1)$, the *probability* function

such that for all $e \in \mathit{dom}(\pi)$

$$\exists\, Q \subseteq \mathit{dom}(\pi) : e \in Q \;\wedge\; Q \text{ is a cluster } \wedge \sum_{e' \in Q} \pi(e') = 1.$$

□

$\longrightarrow_p$ indicates a partial function. The constraint requires the domain of π to consist completely of clusters such that the sum of the probabilities assigned to all events in a cluster equals one. In this way, cluster Q in $\langle \mathcal{E}, \pi \rangle$ can be considered to represent a *stochastic experiment* for which the probability of outcome $e \in Q$ equals $\pi(e)$.

For depicting probabilistic event structures we use the following conventions. The probability of an event is depicted near to the event. For convenience, we often omit the event label for $e \in \mathit{dom}(\pi)$ and indicate the mutual conflicts between events in a cluster by a grey shaded surface. We use Π, possibly subscripted and/or primed, to denote a probabilistic event structure and EBES_P to denote the class of probabilistic event structures. $\mathsf{cl}(\Pi)$ denotes the set of clusters of Π that are assigned a probability. (Note that it is not required for each cluster of Π to be contained in the domain of π.)

9.3. Example. Some example probabilistic event structures are depicted in Figure 9.1. Figure 9.1(b) contains a single cluster of 4 events with $\pi(e_1) = \frac{1}{4}$, $\pi(e_2) = \frac{1}{12}$, $\pi(e_3) = \frac{1}{6}$ and $\pi(e_4) = \frac{1}{2}$. Figure 9.1(c), referred to as Π, contains two clusters. That is, $\mathsf{cl}(\Pi) = \{\{e_1, e_2\}, \{e_3, e_4, e_5\}\}$.

The structures in Figure 9.2 are *not* probabilistic event structures. Figure 9.2(a) violates the requirement that the domain of π consists of clusters only—$e_1, e_3 \in \mathit{dom}(\pi)$, but $\neg\,(e_1 \mathbin{\#} e_3)$.

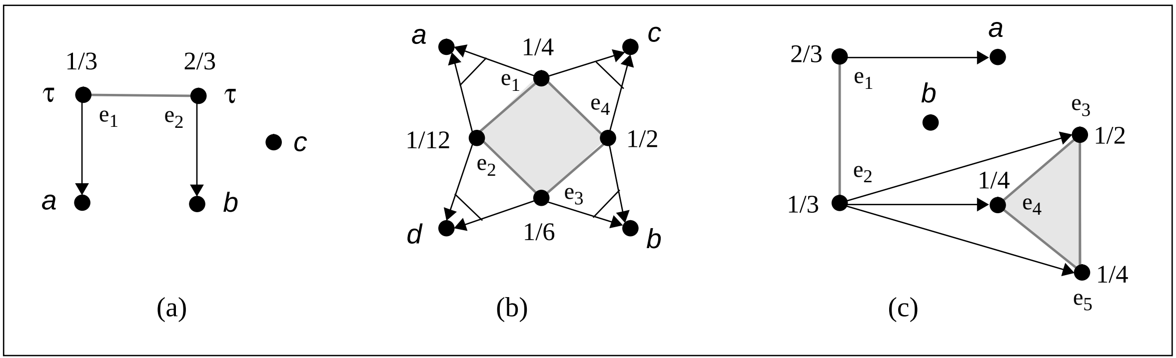


Figure 9.1: Some example probabilistic event structures.

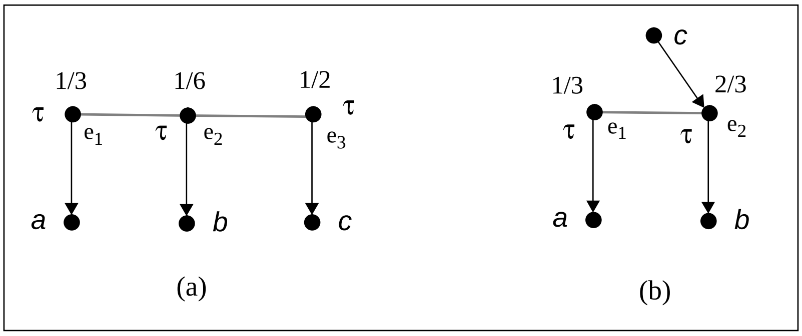


Figure 9.2: Some example event structures that are not probabilistic.

Since $e_1, e_2 \in dom(\pi)$ but have different enablings, Figure 9.2(b) violates the constraints of being a probabilistic event structure. $\square$

The set of event traces of Π is simply the set of event traces of $\mathcal{E}$; the probabilities do not affect the possibility of events to happen, they only quantify the probability of happening. This also means that the set of configurations of Π, $C_P(\Pi)$, is simply equal to $C(\mathcal{E})$, and lposets of Π can be generated according to the recipe for plain event structures (cf. Chapter 2).

9.2.2 Probabilistic remainder

The definitions and results in this section are all relative to $\Pi = \langle (E, \rightsquigarrow, \mapsto, l), \pi \rangle$. The status of a probabilistic event structure after the execution of a sequence of events is defined as follows:

9.4. Definition. (*Probabilistic remainder*)

The *probabilistic remainder* $\Pi[\sigma] = \langle \mathcal{E}', \pi' \rangle$ of $\Pi = \langle \mathcal{E}, \pi \rangle$ after event trace σ is

- $\mathcal{E}' = \mathcal{E}[\sigma] = (E', \rightsquigarrow', \mapsto, l')$, and
- $\pi' = \pi \upharpoonright (E' \setminus \{\, e' \in E' \mid \exists\, e \in \overline{\sigma} : e \# e' \,\})$.

$\square$

The first component is equal to the remainder of $\mathcal{E}$, see Definition 2.28. All events in σ are removed from the domain of π (i.e., $\pi \upharpoonright E'$, where $E' = E \setminus \overline{\sigma}$). In addition, the probabilities of events in conflict with some event e in σ are removed, because the stochastic experiment (= cluster) of which e is part of has happened. Notice that the remaining events of this experiment cannot happen anymore as they were in mutual conflict with e. This is established by introducing an empty bundle pointing to those events in the remainder; see Definition 2.28.

9.5. EXAMPLE. The notion of probabilistic remainder is exemplified in Figure 9.3. After the execution of e_a the (only) cluster is enabled, and after the execution of e_3 (labelled τ) the cluster is 'broken' and events e_1 and e_2 are removed from $dom(\pi)$. □

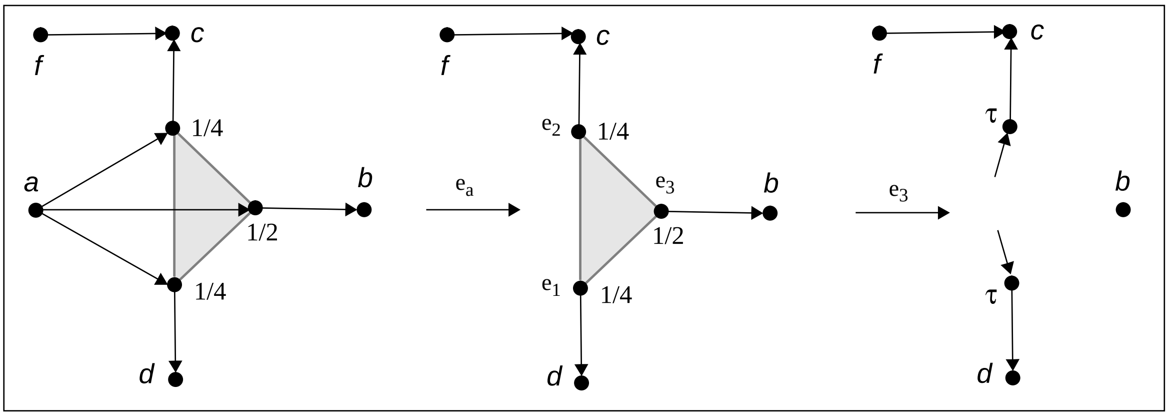


Figure 9.3: Example remainder of a probabilistic event structure.

As a next step we prove that the probabilistic remainder of a probabilistic event structure is again a probabilistic event structure. We first need some results concerning clusters in remainders. The first lemma states that a cluster is unaffected if no event in it is executed.

9.6. LEMMA. $\forall \sigma \in T_P(\Pi), Q \in \mathsf{cl}(\Pi) : Q \cap \overline{\sigma} = \varnothing \Rightarrow Q \in \mathsf{cl}(\Pi[\sigma])$.

PROOF. Let $\sigma \in T_P(\Pi)$ and assume Q is a cluster in Π such that $Q \cap \overline{\sigma} = \varnothing$. Let $\Pi[\sigma] = \langle (E', \rightsquigarrow', \mapsto', l'), \pi' \rangle$. We systematically check all requirements for Q being a cluster in $\Pi[\sigma]$.

1. Given that $Q \cap \overline{\sigma} = \varnothing$ we have $Q \subseteq E \Leftrightarrow Q \subseteq E \setminus \overline{\sigma} \Leftrightarrow Q \subseteq E'$.
2. $|\,Q\,| > 1$ follows immediately from $Q \cap \overline{\sigma} = \varnothing$ and $Q \in \mathsf{cl}(\Pi)$.
3. For all $e \in Q : l'(e) = (l \restriction E')(e) = l(e) = \tau$.
4. For all $e, e' \in Q : e \neq e' \Rightarrow e \#' e'$ follows immediately from the fact that $\rightsquigarrow' = \rightsquigarrow \cap (E' \times E')$ and Q is a cluster of Π.
5. For all $e \in Q, e' \in E' : e \rightsquigarrow' e' \Rightarrow e' \in Q$. Follows immediately from the fact that $\rightsquigarrow' = \rightsquigarrow \cap (E' \times E')$ and Q is a cluster of Π.
6. For all $e, e' \in Q, X \subseteq E' : X \mapsto' e \Rightarrow X \mapsto' e'$. From Definition 2.28 we know that the interesting cases are when either (a) an existing bundle $X \mapsto e$ is removed or (b) a new one $\varnothing \mapsto' e$ is introduced.
   (a) If a bundle X pointing to some event in Q is removed (since $X \cap \overline{\sigma} = \{ e_j \}$), then all bundles originating from e_j are removed in $\Pi[\sigma]$. Since all events in Q have the same bundles pointing to them in Π this means that all bundles $X \mapsto e$ with $e \in Q$ are removed.

(b) If a new bundle $X = \varnothing$ pointing to some event $e \in Q$ is added, this can only be because there exists e' such that $e \rightsquigarrow e'$. Since $Q \in \mathsf{cl}(\Pi)$, $e'' \rightsquigarrow e'$ for all $e'' \in Q$, so bundle $\varnothing \mapsto' e''$ is present in $\Pi[\sigma]$ for all $e'' \in Q$.

This proves that all events in Q have the same bundles pointing to them in $\Pi[\sigma]$.

7. Sum of the probabilities in Q equals 1. Since $Q \cap \overline{\sigma} = \varnothing$ we have $Q \cap dom(\pi') = Q \cap dom(\pi) = Q$, and $\pi'(e) = \pi(e)$ for all $e \in Q$.

□

The following lemma says that once an event in a cluster is executed the entire cluster is 'broken'. Let $\Pi[\sigma] = \langle \mathcal{E}', \pi' \rangle$.

9.7. LEMMA. $\forall \sigma \in T_P(\Pi), Q \in \mathsf{cl}(\Pi) : Q \cap \overline{\sigma} \neq \varnothing \ \Rightarrow \ Q \cap dom(\pi') = \varnothing$.

PROOF. Let $\sigma \in T_P(\Pi)$, $Q \in \mathsf{cl}(\Pi)$ such that $Q \cap \overline{\sigma} = \{ e_1, \ldots, e_k \}$, for $k > 1$. Since Q is a cluster of Π we have for all $e \in Q$ that $e \# e_j$ for $0 < j \leqslant k$. From Definition 9.4 it follows that all these events are removed from $dom(\pi)$, and so $Q \cap dom(\pi') = \varnothing$. □

9.8. THEOREM. $\forall \Pi \in \mathsf{EBES}_P$ and $\sigma \in T_P(\Pi) : \Pi[\sigma] \in \mathsf{EBES}_P$.

PROOF. Let $\Pi' = \Pi[\sigma] = \langle \mathcal{E}', \pi' \rangle$. It is quite evident that $\mathcal{E}' = \mathcal{E}[\sigma]$ satisfies the requirements for being an extended bundle event structure. Besides, π' satisfies the constraints of Definition 9.2 since $\mathsf{cl}(\Pi) \subseteq \mathsf{cl}(\Pi')$—if some event in a cluster Q in Π appears in $\overline{\sigma}$, then all events in Q are removed from the domain of π (cf. Lemma 9.7), and if no event in Q appears in $\overline{\sigma}$, then Q is unaffected (cf. Lemma 9.6). So, $dom(\pi')$ consists only of clusters Q with $\sum_{e \in Q} \pi'(e) = 1$. □

9.2.3 Probability measure on configurations

In this section we provide a means to calculate probabilities for the dynamic representations of an event structure, namely configurations.

As a first observation we remark that in general, π being a partial function, the set $C_P(\Pi)$ of all configurations of Π does not generate a random space—there are configurations for which it does not make sense to speak about probabilities. For instance, what is the probability of configuration $\{ e_c \}$ of Figure 9.1(a)? There are also *sets* of configurations that are indistinguishable from the probabilistic point of view. For instance, again with reference to Figure 9.1(a), the following configurations are probabilistically indistinguishable:

$$c_1 = \{ e_1 \}, c_2 = \{ e_1, e_c \}, c_3 = \{ e_1, e_a \}, c_4 = \{ e_1, e_a, e_c \} \ .$$

In other words, whenever it is known that some configuration in $V = \{ c_1, c_2, c_3, c_4 \}$ has happened (i.e., its events have happened) it does not make sense to reason about the probability that a particular element of V has happened. All configurations in V share a common feature, viz. the fact that e_1 has happened; moreover, the probability of appearance of e_1 is $\frac{1}{3}$. So, the only question which makes sense in this example is 'What is the probability of having any configuration that contains e_1?'. Below we associate probabilities to sets of configurations.

We first capture the notion of being probabilistic indistinguishable. For $C \in C_P(\Pi)$ let $C \cap dom(\pi)$, the *stochastic choice* of C, denoted by $\mathsf{sc}(C)$.

Equivalence class $[C]_{\rightleftharpoons}$	$\mathsf{sc}(C)$	$Pr\{\,[C]_{\rightleftharpoons}\,\}$
$\varnothing, \{\,e_b\,\}$	$\varnothing$	undefined
$\{\,e_1\,\}, \{\,e_1, e_b\,\}, \{\,e_1, e_a\,\}, \{\,e_1, e_a, e_b\,\}$	$\{\,e_1\,\}$	2/3
$\{\,e_2\,\}, \{\,e_2, e_b\,\}$	$\{\,e_2\,\}$	1/3
$\{\,e_2, e_3\,\}, \{\,e_2, e_3, e_b\,\}$	$\{\,e_2, e_3\,\}$	1/6
$\{\,e_2, e_4\,\}, \{\,e_2, e_4, e_b\,\}$	$\{\,e_2, e_4\,\}$	1/12
$\{\,e_2, e_5\,\}, \{\,e_2, e_5, e_b\,\}$	$\{\,e_2, e_5\,\}$	1/12

Table 9.1: Equivalence classes, stochastic choices and probabilities for Figure 9.1(c).

9.9. Definition. For $C_1, C_2 \in C_P(\Pi)$ let $\rightleftharpoons$ be defined as $C_1 \rightleftharpoons C_2 \Leftrightarrow \mathsf{sc}(C_1) = \mathsf{sc}(C_2)$. □

It is easy to verify that $\rightleftharpoons$ is an equivalence relation. Let $[C]_{\rightleftharpoons}$ denote the equivalence class of C under $\rightleftharpoons$. That is, $[C]_{\rightleftharpoons} = \{\,C' \in C_P(\Pi) \mid C \rightleftharpoons C'\,\}$.

The probability of a set of configurations is defined for equivalence classes of configurations (under $\rightleftharpoons$) that contain a nonempty set of probabilistic events $\mathsf{sc}(C) = \{\,e_1, \ldots, e_k\,\}$. The probability of such set of configurations is then equal to $\pi(e_1) \cdot \ldots \cdot \pi(e_k)$.

9.10. Definition. (*Probability measure on sets of configurations*)

For $C \in C_P(\Pi)$ such that $\mathsf{sc}(C) \neq \varnothing$, let $Pr\{\,[C]_{\rightleftharpoons}\,\} \triangleq \prod_{e \in \mathsf{sc}(C)} \pi(e)$. □

9.11. Example. Consider the probabilistic event structure of Figure 9.1(c). The equivalence classes under $\rightleftharpoons$, stochastic choices, and probabilities $Pr\{\,[C]_{\rightleftharpoons}\,\}$ of this structure are summarized in Table 9.1. □

9.3 A probabilistic process algebra

This section introduces a probabilistic process algebra PA_P and provides a causality-based semantics using probabilistic event structures. Section 9.3.1 introduces the syntax of PA_P including the syntactical constraints for probabilistic processes. Section 9.3.2 presents the causality-based semantics. Some properties of this semantics are proven in Section 9.3.3. Finally, Section 9.3.4 presents an event-based operational semantics for PA_P and investigates the relationship of this semantics with the causality-based interpretation.

9.3.1 Syntax

In order to express probabilities PA is extended with a *probabilistic choice* operator, denoted $+_p$, for $p \in (0,1)$. Under the assumption that the choice between B_1 and B_2 cannot be influenced by the environment, behaviour $B_1 +_p B_2$ nondeterministically behaves like B_1 (with probability p) or like B_2 (with probability $1-p$).

9.12. DEFINITION. (*Probabilistic formalism $\mathcal{L}$*)

$$B ::= \mathbf{0} \mid \surd \mid a\,;B \mid B+B \mid B +_p B \mid B \,||_G\, B \mid B[H] \mid B \setminus G \mid B >> B \mid B \,[> B. \quad \square$$

$+_p$ and $+$ bind equally strong. Throughout this chapter p, q and r denote elements in $(0,1)$.

In PA_P we distinguish between a standard and a probabilistic choice. We believe that this distinction is important—from a design perspective it is necessary to express choices for which the probability of an alternative is left unspecified. Such quantitative knowledge may either be absent at the current stage of design or it may be deliberately left unspecified. Therefore, one should not be forced to associate such quantity with an alternative. When going from an abstract specification to a more concrete specification it seems useful to consider the refinement of $+$ by $+_p$. (This is not to say that in the final stage of the design trajectory all standard choices are replaced by probabilistic ones.) For these reasons we have decided to *extend* PA with a probabilistic choice rather than to *replace* the standard choice by a probabilistic one.

The assumption that the probabilistic choice between B_1 and B_2 cannot be influenced by the environment is forced by syntactical constraints on B_1 and B_2. These constraints guarantee that $B_1 +_p B_2$ induces an *independent* stochastic experiment. Below we define the syntactical constraints. Besides the syntactical constraints for $+_p$ we must be careful with the mixture of $+$ (or $[>$) and $+_p$. For instance, constructs like

$$a\,;\mathbf{0} + (\tau\,;b\,;\mathbf{0} +_{0.4} \tau\,;c\,;\mathbf{0})$$

are abandoned, since the probability of the appearance of, for example, $\tau\,;b\,;\mathbf{0}$ cannot be determined. Also

$$a\,;\surd\,[> (\tau\,;b\,;\mathbf{0} +_{0.99} \tau\,;c\,;\mathbf{0})$$

is not an allowed expression, since the probability of $\tau\,;b$ depends on whether $a\,;\surd$ terminates successfully or not.

Before characterizing the expressions belonging to PA_P we introduce two subsidiary predicates pc and ppc. $\mathsf{ppc}(B)$ is true iff B is a (pure) probabilistic choice at 'top' level.

9.13. DEFINITION. Let $\mathsf{ppc} : \mathcal{L} \longrightarrow \mathsf{Bool}$ be defined as follows:

$$\begin{aligned}
\mathsf{ppc}(B_1 +_p B_2) &\triangleq (\mathsf{ppc}(B_1) \vee B_1 = \tau\,;B_1') \wedge (\mathsf{ppc}(B_2) \vee B_2 = \tau\,;B_2') \\
\mathsf{ppc}(B_1 >> B_2) &\triangleq \mathsf{ppc}(B_1) \\
\mathsf{ppc}(\mathsf{op}\,B) &\triangleq \mathsf{ppc}(B) \text{ for } \mathsf{op} \in \{\setminus, [\,]\}.
\end{aligned}$$

ppc is false for all other syntactical constructs. $\square$

$\mathsf{pc}(B)$ is true iff B has a probabilistic choice at the 'component' level.

9.14. Definition. Let $\mathsf{pc} : \mathcal{L} \longrightarrow \mathsf{Bool}$ be defined as follows:

$$\begin{array}{rcl} \mathsf{pc}(B_1 +_p B_2) & \triangleq & \text{true} \\ \mathsf{pc}(B_1 >> B_2) & \triangleq & \mathsf{pc}(B_1) \\ \mathsf{pc}(B_1 \,||_G\, B_2) & \triangleq & \mathsf{pc}(B_1) \,\vee\, \mathsf{pc}(B_2) \\ \mathsf{pc}(\mathsf{op}\, B) & \triangleq & \mathsf{pc}(B) \text{ for } \mathsf{op} \in \{\, \backslash, [\,]\,\}. \end{array}$$

pc is false for all other syntactical constructs. □

9.15. Definition. (*Probabilistic process algebra* PA_P)

$\mathsf{PA}_P \triangleq \{\, B \in \mathcal{L} \mid \mathsf{ppa}(B)\,\}$ where $\mathsf{ppa} : \mathcal{L} \longrightarrow \mathsf{Bool}$ is defined as:

$$\begin{array}{rcl} \mathsf{ppa}(\mathbf{0}) & \triangleq & \text{true} \\ \mathsf{ppa}(\surd) & \triangleq & \text{true} \\ \mathsf{ppa}(\mathsf{op}\, B) & \triangleq & \mathsf{ppa}(B) \text{ for } \mathsf{op} \in \{\, a\,;, \backslash, [\,]\,\} \\ \mathsf{ppa}(B_1\, \mathsf{op}\, B_2) & \triangleq & \mathsf{ppa}(B_1) \,\wedge\, \mathsf{ppa}(B_2) \text{ for } \mathsf{op} \in \{\, ||_G\,, >>\,\} \\ \mathsf{ppa}(B_1 + B_2) & \triangleq & \neg\,\mathsf{pc}(B_1) \,\wedge\, \neg\,\mathsf{pc}(B_2) \,\wedge\, \mathsf{ppa}(B_1) \,\wedge\, \mathsf{ppa}(B_2) \\ \mathsf{ppa}(B_1 +_p B_2) & \triangleq & \mathsf{ppc}(B_1 +_p B_2) \,\wedge\, \mathsf{ppa}(B_1) \,\wedge\, \mathsf{ppa}(B_2) \\ \mathsf{ppa}(B_1\, [> B_2) & \triangleq & \neg\,\mathsf{pc}(B_1) \,\wedge\, \neg\,\mathsf{pc}(B_2) \,\wedge\, \mathsf{ppa}(B_1) \,\wedge\, \mathsf{ppa}(B_2). \end{array}$$

□

B is a legitimate expression of PA_P if its components are legitimate expressions. The components of a probabilistic choice should start with an internal action τ, or should be probabilistic choices. In a standard choice or disrupt both argument behaviours may not contain a probabilistic choice at the 'component' level.

Examples of expressions that belong to PA_P are

$$\begin{array}{r} (\tau\,;\, a\,;\, \mathbf{0} \,+_{0.3}\, \tau\,;\, b\,;\, \mathbf{0}) \,||_b\, c\,;\, b\,;\, \mathbf{0} \\ a\,;\, \mathbf{0} + b\,;\, (\tau\,;\, a\,;\, \mathbf{0} \,+_{0.99}\, \tau\,;\, c\,;\, \mathbf{0}) \\ \tau\,;\, a\,;\, \mathbf{0} \,+_{0.3}\, (\tau\,;\, b\,;\, \mathbf{0} \,+_{0.4}\, \tau\,;\, c\,;\, \mathbf{0}) \quad . \end{array}$$

Notice that probabilistic choices can be used in the context of parallel compositions.

Probabilistic choices are restricted to be performed between behaviours the first actions of which are required to be unobservable actions. For instance, $a\,;\, B_1 \,+_p\, a\,;\, B_2$ and $a\,;\, B_1 \,+_p\, \tau\,;\, B_2$ are not taken into consideration here, although their nonprobabilistic counterparts express instances of nondeterminism. The reason for this choice is to keep our model as simple as possible. On the other hand, we also have the following equations, where $\approx_{te}$ denotes *testing equivalence* ([111], see also Chapter 1),

$$\begin{array}{rcl} a\,;\, B_1 + a\,;\, B_2 & \approx_{te} & \tau\,;\, a\,;\, B_1 + \tau\,;\, a\,;\, B_2 \\ a\,;\, B_1 + \tau\,;\, B_2 & \approx_{te} & \tau\,;\, ((a\,;\, B_1) + B_2) + \tau\,;\, B_2 \quad . \end{array}$$

Thus all forms of nondeterminism can be rewritten in the required format of our formalism, while preserving the notion of testing equivalence. As a consequence the proposed model is expressive enough as long as reasoning modulo testing equivalence is acceptable.

9.3.2 Causality-based semantics

In this section we give a causality-based semantics to PA_P. We do so by defining a mapping $\mathcal{E}_P[\![\ ]\!] : \mathsf{PA}_P \longrightarrow \mathsf{EBES}_P$.

9.16. DEFINITION. Let $\Phi_P : \mathsf{PA}_P \longrightarrow \mathsf{PA}$ be defined as follows

$$
\begin{array}{rcl}
\Phi_P(\mathbf{0}) & \triangleq & \mathbf{0} \\
\Phi_P(\surd) & \triangleq & \surd \\
\Phi_P(\mathsf{op}\, B) & \triangleq & \mathsf{op}\, \Phi_P(B) \text{ for } \mathsf{op} \in \{\, a\,;\, , \backslash, [\,]\,\} \\
\Phi_P(B_1 +_p B_2) & \triangleq & \Phi_P(B_1) + \Phi_P(B_2) \\
\Phi_P(B_1\, \mathsf{op}\, B_2) & \triangleq & \Phi_P(B_1)\, \mathsf{op}\, \Phi_P(B_2) \text{ for } \mathsf{op} \in \{\, +, ||_G\, , >>, [>\,\}.
\end{array}
$$

□

So, Φ_P associates to a probabilistic behaviour B in PA_P its corresponding nonprobabilistic behaviour $\Phi_P(B)$ in PA by simply transforming all occurrence of $+_p$ in B into $+$.

In the following definition let $\mathcal{E}_P[\![\, B_i\,]\!] = \Pi_i = \langle \mathcal{E}_i, \pi_i \rangle$, for i=1, 2. The definition of $\mathcal{E}[\![\ ]\!]$ is provided in Chapter 2. The function *init* which is defined for event structures in Chapter 2 is used for probabilistic event structures in the same way.

9.17. DEFINITION. (*Causality-based semantics of* PA_P)

Let $\mathcal{E}_P[\![\ ]\!] : \mathsf{PA}_P \rightarrow \mathsf{EBES}_P$ be defined as follows:

$$
\begin{array}{rcl}
\mathcal{E}_P[\![\, \mathbf{0}\,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_P(\mathbf{0})\,]\!], \varnothing \rangle \\
\mathcal{E}_P[\![\, \surd\,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_P(\surd)\,]\!], \varnothing \rangle \\
\mathcal{E}_P[\![\, \mathsf{op}\, B_1\,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_P(\mathsf{op}\, B_1)\,]\!], \pi_1 \rangle \text{ for } \mathsf{op} \in \{\, a\,;\, , \backslash, [\,]\,\} \\
\mathcal{E}_P[\![\, B_1\, \mathsf{op}\, B_2\,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_P(B_1\, \mathsf{op}\, B_2)\,]\!], \pi_1 \cup \pi_2 \rangle \text{ for } \mathsf{op} \in \{\, +, >>, [>\,\} \\
\mathcal{E}_P[\![\, B_1 +_p B_2\,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_P(B_1 +_p B_2)\,]\!], \pi \rangle \text{ where} \\
\pi & = & \pi_1 \restriction (E_1 \setminus \mathit{init}(\Pi_1)) \cup \pi_2 \restriction (E_2 \setminus \mathit{init}(\Pi_2)) \\
 & & \cup \, \{\, (e, p) \mid e \in \mathit{init}(\Pi_1) \setminus \mathit{dom}(\pi_1)\,\} \\
 & & \cup \, \{\, (e, p \cdot \pi_1(e)) \mid e \in \mathit{init}(\Pi_1) \cap \mathit{dom}(\pi_1)\,\} \\
 & & \cup \, \{\, (e, 1{-}p) \mid e \in \mathit{init}(\Pi_2) \setminus \mathit{dom}(\pi_2)\,\} \\
 & & \cup \, \{\, (e, (1{-}p) \cdot \pi_2(e)) \mid e \in \mathit{init}(\Pi_2) \cap \mathit{dom}(\pi_2)\,\} \\
\mathcal{E}_P[\![\, B_1\, ||_G\, B_2\,]\!] & \triangleq & \langle \mathcal{E}[\![\, \Phi_P(B_1\, ||_G\, B_2)\,]\!], \pi \rangle \text{ with} \\
\pi & = & \{\, ((e, *), p) \mid (e, p) \in \pi_1 \ \wedge \ (e, *) \in E\,\} \, \cup \\
 & & \{\, ((*, e), p) \mid (e, p) \in \pi_2 \ \wedge \ (*, e) \in E\,\}.
\end{array}
$$

□

Apart from the probability part π the semantics of the probabilistic expression B=$B_1 +_p B_2$ is equivalent to the semantics of the nondeterministic choice. For noninitial events of B, π is defined as the union of π_1 and π_2. For initial events the situation is slightly more complicated.

All probabilities of initial events of B_1 must be multiplied with p and those of B_2 with $1-p$. In order to do so we have to distinguish between events that are already assigned a probability in B_1 or B_2 and those that are not.

In $\mathcal{E}_P[\![B_1 \,||_G\, B_2]\!]$ events are assigned a probability when one of their components is equal to $*$ and the other component is assigned a probability in $\mathcal{E}_P[\![B_i]\!]$, for i=1, 2.

9.18. Example. Figure 9.4 shows the probabilistic event structures corresponding to (a) $B_1 = \tau\,;\, b\,;\, \mathbf{0} +_{1/3} \tau\,;\, \mathbf{0}$, (b) $B_2 = \tau\,;\, \mathbf{0} +_{1/2} (\tau\,;\, b\,;\, (\tau\,;\, \mathbf{0} +_{2/5} \tau\,;\, \mathbf{0}) +_{1/2} \tau\,;\, \mathbf{0})$, and (c) $B_1 +_{1/6} B_2$. The reader should be able to find corresponding expressions of the event structures of Figure 9.1 without great difficulty. □

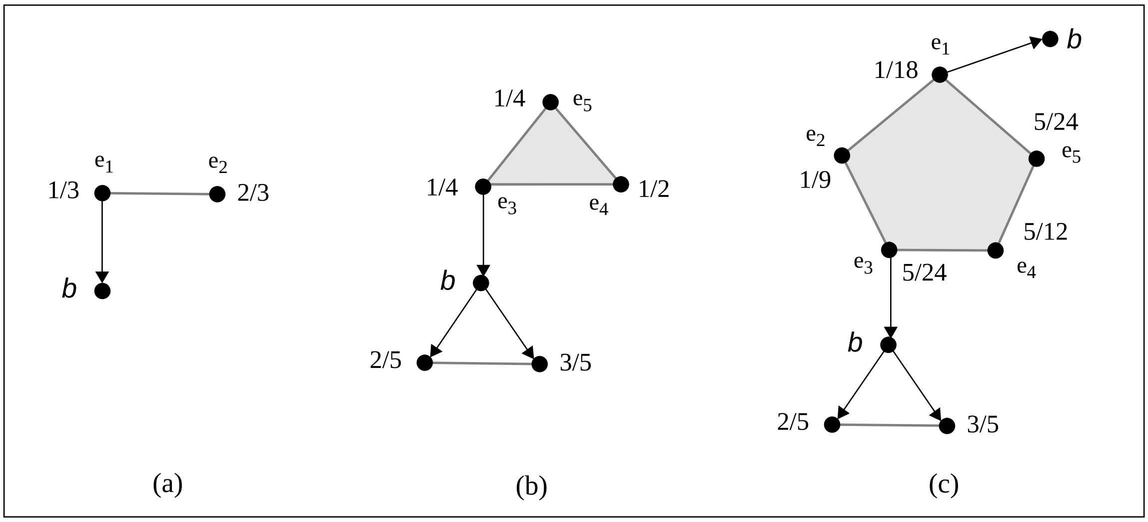


Figure 9.4: Example of semantics for probabilistic choice.

The probabilistic extension is backwards compatible with the plain case, in the sense that the semantics $\mathcal{E}[\![\ ]\!]$ of a behaviour in PA is fully preserved in the definition of $\mathcal{E}_P[\![\ ]\!]$.

9.19. Theorem. *Compatibility theorem*

$$\forall\, B \in \mathsf{PA}_P : L(\mathcal{E}_P[\![B]\!]) = L(\mathcal{E}[\![\Phi_P(B)]\!]).$$

Proof. Let $\mathcal{E}_P[\![B]\!] = \langle \mathcal{E}, \pi \rangle$. By definition $L(\mathcal{E}_P[\![B]\!]) = L(\mathcal{E})$. From Definition 9.17 it immediately follows that $\mathcal{E} = \mathcal{E}[\![\Phi_P(B)]\!]$. □

9.3.3 Properties

As a next property we would like to prove that for all $B \in \mathsf{PA}_P$ its causality-based semantics $\mathcal{E}_P[\![B]\!]$ is a probabilistic event structure. This means that $\mathcal{E}_P[\![B]\!]$ must satisfy the constraints of Definition 9.2. We first prove that for expressions B that do not satisfy the pc predicate do not contain any initial probabilistic event in $\mathcal{E}_P[\![B]\!]$.

9.20. Lemma. For $B \in \mathsf{PA}_P$ let $\mathcal{E}_P[\![B]\!] = \Pi = \langle \mathcal{E}, \pi \rangle$. Then we have:

$$\neg\, \mathsf{pc}(B) \;\Rightarrow\; \mathit{init}(\Pi) \cap \mathit{dom}(\pi) = \varnothing.$$

PROOF. By induction on the structure of B.
Base: For $B = \mathbf{0}$ and $\surd$ the lemma trivially holds since $dom(\pi) = \varnothing$ for these cases. For $B = a_\xi\,;B_1$ we have $init(\Pi) \cap dom(\pi) = \{\,\xi\,\} \cap dom(\pi_1) = \varnothing$.
Induction Step: Assume the lemma holds for B_1 and B_2 and suppose $\neg\,\mathsf{pc}(B)$. Let $\mathcal{E}_P[\![\,B_i\,]\!] = \Pi_i = \langle \mathcal{E}_i, \pi_i \rangle$, for i=1, 2. We only consider $+$, $+_p$, $>>$, and $||_G$; the proofs for the other constructs are similar and omitted.

1. Choice: $B = B_1 + B_2$. For this case we infer:

$$
\begin{array}{ll}
 & init(\mathcal{E}_P[\![\,B_1 + B_2\,]\!]) \cap dom(\pi) \\
= & \{\ \text{Definition 9.17}\ \} \\
 & (init(\Pi_1) \cup init(\Pi_2)) \cap (dom(\pi_1) \cup dom(\pi_2)) \\
= & \{\ E_1 \cap E_2 = \varnothing\ \} \\
 & (init(\Pi_1) \cap dom(\pi_1)) \cup (init(\Pi_2) \cap dom(\pi_2)) \\
= & \{\ B_1 + B_2 \in \mathsf{PA}_P \ \Rightarrow\ \neg\,\mathsf{pc}(B_1)\ \wedge\ \neg\,\mathsf{pc}(B_2);\ \text{induction hypothesis}\ \} \\
 & \varnothing\ .
\end{array}
$$

2. Probabilistic choice: trivial, since the premise does not hold.

3. Enabling: $B = B_1 >> B_2$. For this case we infer:

$$
\begin{array}{ll}
 & init(\mathcal{E}_P[\![\,B_1 >> B_2\,]\!]) \cap dom(\pi) \\
= & \{\ \text{Definition 9.17}\ \} \\
 & init(\Pi_1) \cap (dom(\pi_1) \cup dom(\pi_2)) \\
= & \{\ E_1 \cap E_2 = \varnothing\ \} \\
 & init(\Pi_1) \cap dom(\pi_1) \\
= & \{\ \neg\,\mathsf{pc}(B_1 >> B_2) \Leftrightarrow \neg\,\mathsf{pc}(B_1);\ \text{induction hypothesis}\ \} \\
 & \varnothing\ .
\end{array}
$$

4. Parallel composition: $B = B_1\,||_G\,B_2$. Then:

$$
\begin{array}{ll}
 & init(\mathcal{E}_P[\![\,B_1\,||_G\,B_2\,]\!]) \cap dom(\pi) = \varnothing \\
\Leftrightarrow & \{\ \text{Definition 9.17}\ \} \\
 & init(\mathcal{E}_P[\![\,B_1\,||_G\,B_2\,]\!]) \cap ((dom(\pi_1) \times \{\,*\,\}) \cup (\{\,*\,\} \times dom(\pi_2))) = \varnothing \\
\Leftarrow & \{\ \{\,e_1 \mid (e_1, *) \in init(\mathcal{E}_P[\![\,B_1\,||_G\,B_2\,]\!])\,\} \subseteq init(\Pi_1);\ \text{similar for } \Pi_2\ \} \\
 & (init(\Pi_1) \cup init(\Pi_2)) \cap (dom(\pi_1) \cup dom(\pi_2)) = \varnothing \\
\Leftrightarrow & \{\ \neg\,\mathsf{pc}(B_1\,||_G\,B_2) \Leftrightarrow \neg\,\mathsf{pc}(B_1)\ \wedge\ \neg\,\mathsf{pc}(B_2);\ \text{induction hypothesis}\ \} \\
 & \text{true}\ .
\end{array}
$$

□

The following lemma says that the initial events of expression B for which $\mathsf{ppc}(B)$ holds constitute a cluster.

9.21. LEMMA. $\forall\, B \in \mathsf{PA}_P : \mathsf{ppc}(B) \;\Rightarrow\; \mathit{init}(\mathcal{E}_P[\![\, B \,]\!]) \in \mathsf{cl}(\mathcal{E}_P[\![\, B \,]\!])$.

PROOF. By induction on the structure of B.
Base: For $B = \mathbf{0}$ and $B = \surd$ the premise does not hold, so the lemma holds.
Induction Step: Assume the lemma holds for B_1 and B_2. From the definition of ppc it is clear that we only have to consider probabilistic choice, enabling, hiding and relabelling. For all other constructs the predicate does not hold and the lemma is trivially true. Let $\Pi = \mathcal{E}_P[\![\, B \,]\!] = \langle \mathcal{E}, \pi \rangle$ and $\Pi_i = \mathcal{E}_P[\![\, B_i \,]\!] = \langle \mathcal{E}_i, \pi_i \rangle$, for i=1, 2.

1. $B = B_1 >> B_2$. For this case we have $\mathit{init}(\Pi) = \mathit{init}(\Pi_1)$ and $\mathsf{cl}(\Pi_1) \subseteq \mathsf{cl}(\Pi)$. From the induction hypothesis we know $\mathit{init}(\Pi_1) \subseteq \mathsf{cl}(\Pi_1)$, and so $\mathit{init}(\Pi) \subseteq \mathsf{cl}(\Pi)$. The proofs for hiding and relabelling are similar and omitted.

2. $B = B_1 +_p B_2$. According to the definition of ppc there are four cases to be distinguished:

   (a) $B_1 = \tau_\xi\, ;\, B_1'$ and $B_2 = \tau_\psi\, ;\, B_2'$. From Definition 9.17 it follows that $\mathit{init}(\Pi) = \{\, \xi, \psi \,\}$, $\psi \,\#\, \xi$, and that ξ, ψ are not in conflict with any other event. In addition, no bundles point to ξ and ψ, $\pi(\xi) = p$ and $\pi(\psi) = 1-p$, so the sum of probabilities in $\mathit{init}(\Pi)$ equals 1. This proves that $\mathit{init}(\Pi) \in \mathsf{cl}(\Pi)$.

   (b) B_1 is of the form $B_1' +_q B_1''$ and $B_2 = \tau_\psi\, ;\, B_2'$. According to the induction hypothesis $\mathit{init}(\Pi_1) \in \mathsf{cl}(\Pi_1)$. From Definition 9.17 it follows that $\mathit{init}(\Pi) = \mathit{init}(\Pi_1) \cup \mathit{init}(\Pi_2)$ and that all events in $\mathit{init}(\Pi_1)$ are put in conflict with all events in $\mathit{init}(\Pi_2)$. Besides, no other conflicts or bundles are added. It directly follows that $\mathit{init}(\Pi)$ satisfies the constraints of Definition 9.1. It remains to check that $\sum_{e \in \mathit{init}(\Pi)} \pi(e)$ equals 1:

$$
\begin{aligned}
& \sum_{e \in \mathit{init}(\Pi)} \pi(e) \\
= \quad & \{\ \mathit{init}(\Pi) = \mathit{init}(\Pi_1) \cup \mathit{init}(\Pi_2)\ \} \\
& \sum_{e \in \mathit{init}(\Pi_1) \,\cup\, \mathit{init}(\Pi_2)} \pi(e) \\
= \quad & \{\ \text{Definition 9.17}\ \} \\
& \sum_{e \in \mathit{init}(\Pi_1) \setminus \mathit{dom}(\pi_1)} p + \sum_{e \in \mathit{init}(\Pi_1) \cap \mathit{dom}(\pi_1)} p \cdot \pi_1(e) \\
& + \sum_{e \in \mathit{init}(\Pi_2) \setminus \mathit{dom}(\pi_2)} (1-p) + \sum_{e \in \mathit{init}(\Pi_2) \cap \mathit{dom}(\pi_2)} (1-p) \cdot \pi_2(e) \\
= \quad & \{\ \mathit{init}(\Pi_2) = \{\, \psi \,\};\ \psi \notin \mathit{dom}(\pi_2)\ \} \\
& (1-p) + \sum_{e \in \mathit{init}(\Pi_1) \setminus \mathit{dom}(\pi_1)} p + \sum_{e \in \mathit{init}(\Pi_1) \cap \mathit{dom}(\pi_1)} p \cdot \pi_1(e) \\
= \quad & \{\ \mathit{init}(\Pi_1) \in \mathsf{cl}(\Pi_1) \Leftrightarrow \mathit{init}(\Pi_1) \cap \mathit{dom}(\pi_1) = \mathit{init}(\Pi_1)\ \} \\
& (1-p) + p \cdot \sum_{e \in \mathit{init}(\Pi_1)} \pi_1(e) \\
= \quad & \{\ \text{induction hypothesis}\ \} \\
& 1\ .
\end{aligned}
$$

(c) B_2 is of the form $B_2' +_r B_2''$ and $B_1 = \tau_\psi \,;\, B_1'$. Similar to the previous case.

(d) B_1 is of the form $B_1' +_q B_1''$ and B_2 is of the form $B_2' +_r B_2''$. According to the induction hypothesis $init(\Pi_1) \in \mathsf{cl}(\Pi_1)$ and $init(\Pi_2) \in \mathsf{cl}(\Pi_2)$. Analogously to case 2. it follows in a straightforward way that $init(\Pi)$ satisfies the constraints of Definition 9.1. It remains to check that $\sum_{e \in init(\Pi)} \pi(e) = 1$:

$$
\begin{array}{ll}
& \displaystyle\sum_{e \in init(\Pi)} \pi(e) \\
= & \{ \text{ see derivation above } \} \\
& \displaystyle\sum_{e \in init(\Pi_1) \setminus dom(\pi_1)} p + \sum_{e \in init(\Pi_1) \cap dom(\pi_1)} p \cdot \pi_1(e) \\
+ & \displaystyle\sum_{e \in init(\Pi_2) \setminus dom(\pi_2)} (1-p) + \sum_{e \in init(\Pi_2) \cap dom(\pi_2)} (1-p) \cdot \pi_2(e) \\
= & \{ \; init(\Pi_i) \in \mathsf{cl}(\Pi_i) \Leftrightarrow init(\Pi_i) \cap dom(\pi_i) = init(\Pi_i), \text{ for } i{=}1,2 \; \} \\
& \displaystyle p \cdot \sum_{e \in init(\Pi_1)} \pi_1(e) + (1-p) \cdot \sum_{e \in init(\Pi_2)} \pi_2(e) \\
= & \{ \text{ induction hypothesis } \} \\
& 1 \quad .
\end{array}
$$

□

The previous two lemmas provide the ingredients to prove that for all $B \in \mathsf{PA}_P$ we have that $\mathcal{E}_P[\![B]\!]$ is a probabilistic event structure.

9.22. Theorem. $\forall B \in \mathsf{PA}_P : \mathcal{E}_P[\![B]\!] \in \mathsf{EBES}_P$.

Proof. By induction on the structure of B. For all $B \in \mathsf{PA}_P$ with $\mathcal{E}_P[\![B]\!] = \Pi = \langle \mathcal{E}, \pi \rangle$ it follows from Theorem 9.19 that $\mathcal{E}$ is an extended bundle event structure. It suffices to consider the constraints on π. According to Definition 9.2 this boils down to prove that $dom(\pi)$ consists of clusters Q, for which $\sum_{e \in Q} \pi(e) = 1$.
Base: For $B = \mathbf{0}$ and $B = \surd$ the theorem follows directly since $\pi = \varnothing$ for these cases.
Induction Step: Assume the theorem holds for B_1 and B_2. Let $\mathcal{E}_P[\![B_i]\!] = \Pi_i = \langle \mathcal{E}_i, \pi_i \rangle$, for $i{=}1,2$. We prove the theorem for ;, $+$, $+_p$ and $||_G$. The proofs for the other operators are similar and omitted.

1. $B = a \,;\, B_1$: trivial as $\mathsf{cl}(\Pi) = \mathsf{cl}(\Pi_1)$, $\pi = \pi_1$ and the theorem holds for B_1.

2. $B = B_1 + B_2$: simple, since $\mathsf{cl}(\Pi) = \mathsf{cl}(\Pi_1) \cup \mathsf{cl}(\Pi_2)$, $\pi = \pi_1 \cup \pi_2$ and the theorem holds for B_1 and B_2.

3. $B = B_1 +_p B_2$. It follows from Definition 9.17 and Lemma 9.21 that $\mathsf{cl}(\Pi)$ equals

$$\{ Q \in \mathsf{cl}(\Pi_1) \mid Q \cap init(\Pi_1) = \varnothing \} \cup \{ Q \in \mathsf{cl}(\Pi_2) \mid Q \cap init(\Pi_2) = \varnothing \} \cup init(\Pi).$$

From the induction hypothesis we know that for clusters in $\mathsf{cl}(\Pi_1)$ and $\mathsf{cl}(\Pi_2)$ the sum of the probabilities is 1, and that for these clusters the probability function π is unaffected. From Lemma 9.21 it follows that $init(\Pi) \in \mathsf{cl}(\Pi)$.

4. $B = B_1 \,||_G\, B_2$. According to Definition 9.17 $\mathsf{cl}(\Pi)$ equals

$$\{\, Q \times \{\,*\,\} \mid Q \in \mathsf{cl}(\Pi_1)\,\} \cup \{\, \{\,*\,\} \times Q \mid Q \in \mathsf{cl}(\Pi_2)\,\}.$$

   In addition, $dom(\pi) = (dom(\pi_1) \times \{\,*\,\}) \cup (\{\,*\,\} \times dom(\pi_2))$. From these two characterizations it follows from the induction hypothesis that $dom(\pi)$ solely consists of clusters. Since probabilities in these clusters are unaffected it directly follows that the sum of the probabilities of events in clusters equals one.

□

9.3.4 Event-based operational semantics for PA_P

In this section we present an event-based operational semantics for PA_P. This operational semantics is derived in the same way as in Chapter 5 of this thesis for the timed case. Again each occurrence of an action-prefix and successful termination is subscripted with a unique event occurrence identifier, denoted by a Greek letter.

The operational semantics defines a *probabilistic* event transition system. We use two transition relations: $\longrightarrow$ and $\Longrightarrow$ for normal and probabilistic transitions, respectively. $B \xrightarrow{(e,a)} B'$ denotes that B may perform event e labelled a and evolve into B'. This transition involves no probabilistic event. $B \xRightarrow{(e,\tau,p)} B'$ denotes that B may perform probabilistic event e labelled τ with probability p and subsequently will evolve into B'.

$\longrightarrow$ and $\Longrightarrow$ are the smallest relations closed under the inference rules of Tables 9.2 and 9.3. These inference rules are inspired by a proposal of Langerak & Latella [91] to provide an interleaving semantics to (a subset of) PA_P.

The inference rules of Table 9.2 determine $\longrightarrow$. These rules are almost identical to those of the (nonprobabilistic) event transition system for PA of Chapter 2, except for the two nonsynchronization rules for parallel composition. We require that one component of $B_1 \,||_G\, B_2$ can only autonomously perform a (nonprobabilistic) action a if the other component cannot perform a probabilistic event. In this way, probabilistic transitions have *priority* over other transitions. This avoids that probabilistic and nonprobabilistic transitions are mixed; see Theorem 9.25.

The probabilistic transition rules for PA_P are listed in Table 9.3. There are no inference rules for successful termination, action-prefix, choice and disrupt, since these syntactical constructs cannot perform any probabilistic transition. For $\surd$ and $a\,;B$ this is quite obvious: the first can only perform δ whereas the second can only perform a. $B_1 + B_2$ cannot perform a probabilistic transition since it has no probabilistic choice at 'component' level, i.e., $\neg\,\mathsf{pc}(B_1)$ and $\neg\,\mathsf{pc}(B_2)$ hold. The same applies to $B_1 \,[\!>\, B_2$. The first two rules for $+_p$ are the only rules where ordinary transitions of component behaviours result in probabilistic transitions of the composite behaviour. The second pair of rules for $+_p$ take care of adjusting probabilities. If B_1 may perform event e with probability q, then $B_1 +_p B_2$ may do so with probability $p \cdot q$. The rules for enabling, hiding and relabelling are rather straightforward extensions of the rules for the nonprobabilistic case. For parallel composition the components are required to jointly perform probabilistic transitions, and while doing so their probabilities are multiplied. This

$$\frac{}{\surd_\xi \xrightarrow{(\xi,\delta)} \mathbf{0}} \qquad \frac{}{a_\xi\,;\, B \xrightarrow{(\xi,a)} B}$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1'}{B_1 + B_2 \xrightarrow{(\xi,a)} B_1'} \qquad \frac{B_2 \xrightarrow{(\xi,a)} B_2'}{B_1 + B_2 \xrightarrow{(\xi,a)} B_2'}$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1'}{B_1 >> B_2 \xrightarrow{(\xi,a)} B_1' >> B_2} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{(\xi,\delta)} B_1'}{B_1 >> B_2 \xrightarrow{(\xi,\tau)} B_2}$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1'}{B_1 \,[> B_2 \xrightarrow{(\xi,a)} B_1' \,[> B_2} \quad (a \neq \delta) \qquad \frac{B_1 \xrightarrow{(\xi,\delta)} B_1'}{B_1 \,[> B_2 \xrightarrow{(\xi,\delta)} B_1'}$$

$$\frac{B_2 \xrightarrow{(\xi,a)} B_2'}{B_1 \,[> B_2 \xrightarrow{(\xi,a)} B_2'} \qquad \frac{B \xrightarrow{(\xi,a)} B'}{B[H] \xrightarrow{(\xi,H(a))} B'[H]}$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,*),a)} B_1' \,||_G\, B_2} \quad (a \notin G^\delta \;\wedge\; \neg\, \mathsf{pc}(B_2))$$

$$\frac{B_2 \xrightarrow{(\xi,a)} B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((*,\xi),a)} B_1 \,||_G\, B_2'} \quad (a \notin G^\delta \;\wedge\; \neg\, \mathsf{pc}(B_1))$$

$$\frac{B_1 \xrightarrow{(\xi,a)} B_1' \;\wedge\; B_2 \xrightarrow{(\psi,a)} B_2'}{B_1 \,||_G\, B_2 \xrightarrow{((\xi,\psi),a)} B_1' \,||_G\, B_2'} \quad (a \in G^\delta)$$

$$\frac{B \xrightarrow{(\xi,a)} B'}{B \setminus G \xrightarrow{(\xi,a)} B' \setminus G} \quad (a \notin G) \qquad \frac{B \xrightarrow{(\xi,a)} B'}{B \setminus G \xrightarrow{(\xi,\tau)} B' \setminus G} \quad (a \in G)$$

Table 9.2: Nonprobabilistic transition rules for PA_P.

$$\frac{B_1 \xrightarrow{(\xi,\tau)} B_1'}{B_1 \ +_p \ B_2 \stackrel{(\xi,\tau,p)}{\Longrightarrow} B_1'} \qquad \frac{B_2 \xrightarrow{(\xi,\tau)} B_2'}{B_1 \ +_p \ B_2 \stackrel{(\xi,\tau,1-p)}{\Longrightarrow} B_2'}$$

$$\frac{B_1 \stackrel{(\xi,\tau,q)}{\Longrightarrow} B_1'}{B_1 \ +_p \ B_2 \stackrel{(\xi,\tau,p\cdot q)}{\Longrightarrow} B_1'} \qquad \frac{B_2 \stackrel{(\xi,\tau,q)}{\Longrightarrow} B_2'}{B_1 \ +_p \ B_2 \stackrel{(\xi,\tau,(1-p)\cdot q)}{\Longrightarrow} B_2'}$$

$$\frac{B_1 \stackrel{(\xi,\tau,p)}{\Longrightarrow} B_1'}{B_1 >> B_2 \stackrel{(\xi,\tau,p)}{\Longrightarrow} B_1' >> B_2}$$

$$\frac{B_1 \stackrel{(\xi,\tau,p)}{\Longrightarrow} B_1'}{B_1 \,||_G\, B_2 \stackrel{((\xi,*),\tau,p)}{\Longrightarrow} B_1' \,||_G\, B_2} \ (\neg\, \mathsf{pc}(B_2)) \qquad \frac{B_2 \stackrel{(\xi,\tau,p)}{\Longrightarrow} B_2'}{B_1 \,||_G\, B_2 \stackrel{((*,\xi),\tau,p)}{\Longrightarrow} B_1 \,||_G\, B_2'} \ (\neg\, \mathsf{pc}(B_1))$$

$$\frac{B_1 \stackrel{(\xi,\tau,p)}{\Longrightarrow} B_1' \ \wedge \ B_2 \stackrel{(\psi,\tau,q)}{\Longrightarrow} B_2'}{B_1 \,||_G\, B_2 \stackrel{((\xi,\psi),\tau,p\cdot q)}{\Longrightarrow} B_1' \,||_G\, B_2'}$$

$$\frac{B \stackrel{(\xi,\tau,p)}{\Longrightarrow} B'}{B \setminus G \stackrel{(\xi,\tau,p)}{\Longrightarrow} B' \setminus G} \qquad \frac{B \stackrel{(\xi,\tau,p)}{\Longrightarrow} B'}{B[H] \stackrel{(\xi,\tau,p)}{\Longrightarrow} B'[H]}$$

Table 9.3: Probabilistic transition rules for PA_P.

ensures that the sum of the probabilities of all outgoing transitions of a state equals 1; see Theorem 9.26.

9.23. EXAMPLE. Consider $B = (\tau_\xi \,;\, \mathbf{0} \ +_{0.3} \ \tau_\psi \,;\, \mathbf{0}) \,|||\, a_\chi \,;\, \mathbf{0}$. Since probabilistic transitions have priority over other transitions there is no possibility to initially perform (χ, a). We do have the following derivation:

$$\begin{array}{l}
(\tau_\xi \,;\, \mathbf{0} \ +_{0.3} \ \tau_\psi \,;\, \mathbf{0}) \,|||\, a_\chi \,;\, \mathbf{0} \\
\stackrel{(\xi,\tau,0.3)}{\Longrightarrow} \ \{ \text{ (probabilistic choice), (parallel composition) } \} \\
\mathbf{0} \,|||\, a_\chi \,;\, \mathbf{0} \\
\xrightarrow{((*,\chi),a)} \ \{ \text{ (action-prefix), (parallel composition) } \} \\
\mathbf{0} \,|||\, \mathbf{0} \quad .
\end{array}$$

□

9.24. EXAMPLE. Let $B = (\tau_\xi \,;\, a \,;\, \mathbf{0} \ +_{0.2} \ \tau_\psi \,;\, b \,;\, \mathbf{0}) \,|||\, (\tau_\chi \,;\, \mathbf{0} \ +_{0.6} \ \tau_\varphi \,;\, \mathbf{0})$. The initial state of the transition system corresponding to B has four outgoing probabilistic branches labelled: (a) $((\xi,\chi),\tau,0.48)$, (b) $((\xi,\varphi),\tau,0.32)$, (c) $((\psi,\chi),\tau,0.12)$, and (d) $((\psi,\varphi),\tau,0.08)$. □

In the resulting transition system states can be partitioned into two groups: states that only have outgoing probabilistic transitions and states that only have outgoing nonprobabilistic transitions. There are no states that have both.

9.25. THEOREM. $\forall\, B \in \mathsf{PA}_P : B \not\Longrightarrow \;\vee\; B \not\longrightarrow$.

PROOF. Straightforward by induction on the structure of B. □

The following lemma states that the sum of the probabilities of all outgoing probabilistic transitions of a state equals one.

9.26. THEOREM. $\forall\, B \in \mathsf{PA}_P : (\exists\, e : B \xRightarrow{(e,\tau,p)}\,) \;\Rightarrow\; \sum_{B \xRightarrow{(e,\tau,q)}} q = 1.$

PROOF. Straightforward by induction on the structure of B. □

Let $\mathsf{TS}_P(B)$ be the probabilistic event transition system of B obtained by applying the inference rules to B. For $\mathcal{E}[\![\, B \,]\!]$ a probabilistic transition system ETS_P is constructed in the following way. States of the transition system ETS_P are reachable probabilistic event structures (or, derivates) of $\mathcal{E}[\![\, B \,]\!]$ with $\mathcal{E}[\![\, B \,]\!]$ being the initial state. There is a transition from Π to Π' if $\Pi' = \Pi[\sigma]$ for event trace σ with $\mid \sigma \mid = 1$. We then have the following consistency result between the causality-based semantics and the event-based operational semantics:

9.27. THEOREM. $\forall\, B \in \mathsf{PA}_P : \Phi_P(\mathsf{TS}_P(B)) \approx_{te} \Phi_P(\mathsf{ETS}_P(\mathcal{E}_P[\![\, B \,]\!]))$.

PROOF.

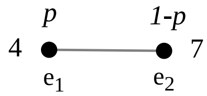
$$
\begin{array}{cl}
 & \Phi_P(\mathsf{ETS}_P(\mathcal{E}_P[\![\, B \,]\!])) \\
=_{\text{iso}} & \{ \text{ Definition 9.17 } \} \\
 & \mathsf{ETS}(\mathcal{E}[\![\, B \,]\!]) \\
\sim & \{ \text{ Theorem 2.46 } \} \\
 & \mathsf{TS}(B) \\
\approx_{te} & \{ \text{ [91, Proposition 4.4] } \} \\
 & \Phi_P(\mathsf{TS}_P(B)) \quad .
\end{array}
$$
□

Stated in words, take the probabilistic transition system for B obtained from the operational semantics and construct a probabilistic transition system for the denotational semantics of B, $\mathcal{E}_P[\![\, B \,]\!]$, by considering event traces of length 1. If the probabilities in the transition labels are omitted (by Φ_P) then the two resulting (plain) transition systems are *testing equivalent.* Remark that this is not such a strong result; for the timed, real-time and urgent case we obtained strong bisimulation equivalence! The reason for this is that in the operational semantics of PA_P probabilistic transitions have priority over other transitions. In this way, the possibility to perform an observable action may be postponed since probabilistic choices have to be resolved first. This phenomenon is not present in the noninterleaving semantics.

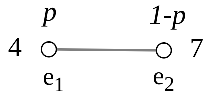
9.4 Time and probability

In this section we briefly discuss the *integration* of our probabilistic model EBES_P, the deterministic (simple) timed model EBES_T, and its urgent variant EBES_U. The resulting integrated model is used in the next section to illustrate how a performance model can be obtained from an event structure model.

In order for clusters to model stochastic experiments we pose the restriction that all events in a cluster are enabled at the same time. Under this constraint situations like



cannot appear. Here it would be difficult to interpret this cluster as a stochastic experiment, since before time 7 only event e_1 can happen and not e_2. An alternative interpretation would be to take the individual timing constraints into account only after having made the probabilistic choice between the events. The main problem with this interpretation is that it is not a plausible interpretation when also considering urgent events. Consider, for instance, the cluster



where event e_2 will never happen since it is excluded by urgent e_1 since $\mathcal{D}(e_1) < \mathcal{D}(e_2)$. Making first a choice among the events without taking the timing constraints into consideration would make no sense here. Here, however, it seems quite reasonable to require e_1 and e_2 to have identical timings; what would otherwise be the rôle of the event probabilities? For simplicity we therefore require all events in a cluster to be enabled at the same time. At a syntactical level it suffices to require all initial (internal) actions in a probabilistic choice to have the same time delay. From an application point of view this is not a severe restriction as typically no time constraints are put on *internal* probabilistic behaviour.

A timed, urgent, probabilistic event structure is an (extended bundle) event structure equipped with the deterministic time, urgency and probability modules, $\mathcal{D}$, $\mathcal{T}$, $\mathcal{U}$ and π, respectively. The causality-based semantics of an extension of PA including $(t)\ a\,;B$, $+_p$ and $\mathcal{U}_U()$ can now easily be provided by combining $\mathcal{E}_U[\![\ ]\!]$ and $\mathcal{E}_P[\![\ ]\!]$ in the most obvious way. It is now straightforward to prove by induction on the structure of behaviour expressions that for all clusters in the event structure corresponding to timed, urgent, probabilistic behaviour all events in these clusters are enabled at the same time.

9.5 Performance analysis—two examples

This section presents two simple examples that illustrate how unreliable time-dependent systems can be specified using our formalism, and, more importantly, that exemplify how a performance model can be generated from a causality-based model. The examples are kept rather intuitive in the sense that no formal mapping between the event structures and the performance model, that is, discrete-time semi-Markov chains, is given.

9.5.1 Discrete-time semi-Markov chains

As we do not expect the reader to be fully acquainted with the notion of *discrete-time semi-Markov chains* (DTSMCs) we give a brief explanation of such processes and explain how limiting distributions can be obtained for such models. It is assumed that the reader is familiar with the notion of discrete-time Markov chains and the notion of limiting distribution (see also Appendix A). A more thorough treatment of semi-Markov processes can be found in Ross [130] and Heyman & Sobel [70].

In a discrete-time Markov chain (DTMC) the state *residence time* (or sojourn time), that is, the probability distribution of staying in a state for a certain time, is restricted to be geometrically distributed. A discrete-time *semi-Markov* chain (DTSMC) allows residence times to have an *arbitrary* distribution. This means that a DTSMC does not need to satisfy the memoryless property (see Lemma 8.2), because the probability of going from one state to another depends not only on the current state (as for memoryless distributions) but also on the amount of time already spent in this state.

Apart from the fact that a DTSMC allows more general residence time distributions, it behaves similar to a DTMC. In fact, when one abstracts from the residence time distributions in a DTSMC one obtains a corresponding DTMC, referred to as the *embedded* DTMC. From Appendix A we recall that the limiting distribution π of a DTMC with transition probability matrix $\mathbf{P}$ can be computed by solving the following system of linear equations

$$\pi \cdot \mathbf{P} = \pi, \quad \sum_i \pi_i = 1 \quad .$$

π_i is the limiting distribution of state i, that is, π_i is the probability of being in state i of the DTMC 'on the long run'. Note that the limiting distribution of a DTMC only exists if the chain is regular (see Appendix A).

The limiting distribution of a DTSMC is calculated by first determining the limiting distribution of its embedded DTMC in the aforementioned way, and subsequently interpreting these results for the DTSMC by taking into account the average residence times. Let U_{ij} be a (discrete) stochastic variable that determines the number of time units spent in state i if the next state is j ($i \neq j$) and let R_i be a (discrete) stochastic variable that determines the residence time of state i (i.e., the number of time units spent in state i). Then

$$Pr\{\, R_i = k \,\} \triangleq \sum_j \mathbf{P}(i,j) \cdot Pr\{\, U_{ij} = k \,\} \quad .$$

Let r_i denote the *average residence time* of state i. That is,

$$r_i \triangleq \sum_k k \cdot Pr\{\, R_i = k \,\} \quad .$$

Let T_i denote the average number of time units between successive transitions to i. The limiting distribution ϕ of a DTSMC is now defined as:

9.28. DEFINITION. (*Limiting distribution of a DTSMC*)

The limiting distribution ϕ_i of state i of a DTSMC equals r_i/T_i. $\square$

The limiting distribution of a DTSMC exists iff a limiting distribution exists for its embedded DTMC. Let π_i be the limiting distribution of state i of the embedded DTMC. An alternative interpretation is that π_i denotes the limiting distribution of the DTSMC at hand being in i at some transition instant, that is, at a moment of transition. Stated otherwise, π_i can be considered as the fraction of *(transition) instants* at which the DTSMC is in state i, considering an infinite amount of transition instants. In order to obtain the fraction of *time* the system is in state i (i.e., ϕ_i), the average residence times must be taken into account. This gives rise to the following relationship between π_i and ϕ_i:

9.29. DEFINITION. (*Alternative characterization of limiting distribution of a DTSMC*)

For i a state of a DTSMC with limiting distribution π_i in the embedded DTMC:

$$\phi_i \triangleq \frac{\pi_i \cdot r_i}{\sum_j \pi_j \cdot r_j} \quad .$$

$\square$

In the following examples we will use these definitions in the following way. Given some DTSMC we first calculate the limiting distributions π_i of its embedded DTMC and determine the average residence times r_i. Using Definition 9.29 we subsequently determine the limiting distributions ϕ_i of the DTSMC. Finally, we calculate T_i by using Definition 9.28.

9.5.2 An unreliable coffee machine

As an example of deducing a performance model from a causality-based model we consider an *unreliable coffee machine*. Although we have not dealt with recursive specifications up to now, this example uses a simple form of recursion—tail recursion—to describe the iterative behaviour of processes. (A formal treatment of recursion is provided in Chapter 10.)

The example consists of a coffee machine C and a user U. U represents an impatient user—after inserting a coin he wants to have coffee at his disposal within n time units, $n \in \mathsf{Time}$. If coffee is not supplied within this time period a new coin is inserted, assuming that the coffee machine suffers from some failure, and the process is repeated. For simplicity it is assumed that consuming coffee takes no time.

$$U := \mathcal{U}_{to}(coin\,;\,(coffee\,;\,U + (n)\,to\,;\,U)) \quad .$$

The coffee machine is quite realistic in the sense that it sometimes refuses to offer any coffee even after a coin has been inserted. Let p be the probability the machine behaves in this unreliable way. Furthermore, producing coffee is assumed to take k time units ($k \in \mathsf{Time}$).

$$C := coin\,;\,(\tau\,;\,C \,+_p\, \tau\,;\,(k)\,coffee\,;\,C) \quad .$$

The overall system is specified by

$$S := U \,||_{\{\, coin, coffee \,\}}\, C \quad .$$

In order to make synchronizations on *coffee* possible we assume in the sequel that $n > k$. The

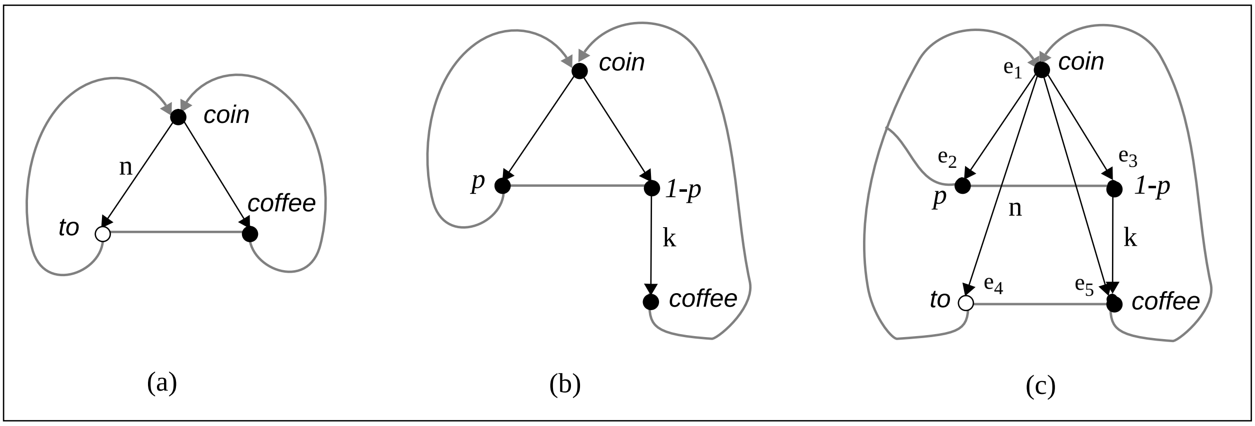


Figure 9.5: Timed probabilistic event structures of (a) U, (b) C, and (c) S.

corresponding timed probabilistic event structures of U, C, and S are depicted in Figure 9.5. These figures only explicitly depict the *finite* part of the event structure corresponding to the "body" of the processes. Recursive calls should be considered as appropriate unfoldings of the finite representations. To illustrate this principle Figure 9.6 illustrates for process U how such unfolding should be performed. Each successive unfolding is obtained by instantiating the original (finite) structure. The sequence of event structures obtained by unfolding in this way is equivalent to the approximations of the denotational semantics of recursive processes as defined in Chapter 10.

The way in which we obtain *finite representations* of infinite event structures is not formalized here and is a subject for further study. Finite representations can be obtained in those cases where the infinite event structure possesses a certain *regular pattern*, such as in Figure 9.6. Unfortunately, it is not so clear to determine this regularity principle such that, for instance, all processes for which a finite labelled transition system exist are captured. An initial attempt to formally characterize this regularity can be found in Latella [93].

Assume now that we want to calculate the average number of cups of coffee, N_c, offered per unit of time. In order to determine this quantity the following grouping of events is introduced $s_1 = \{\, e_1, e_2, e_4 \,\}$ and $s_2 = \{\, e_3, e_5 \,\}$ (see Figure 9.7(a)). s_1 represents the case in which no coffee is offered, s_2 represents the case in which actually coffee is offered, i.e., the successful case. The grouping of events imposes a particular *view* on the system. In this view one abstracts from system characteristics that are irrelevant for the kind of performance analysis one performs. For instance, for our purpose, it is not necessary to keep events e_2 and e_4 separated as they both lead to the same situation, i.e., not offering any coffee. The groups of events and probabilistic transitions between them can be considered as a DTSMC, see Figure 9.7(b).

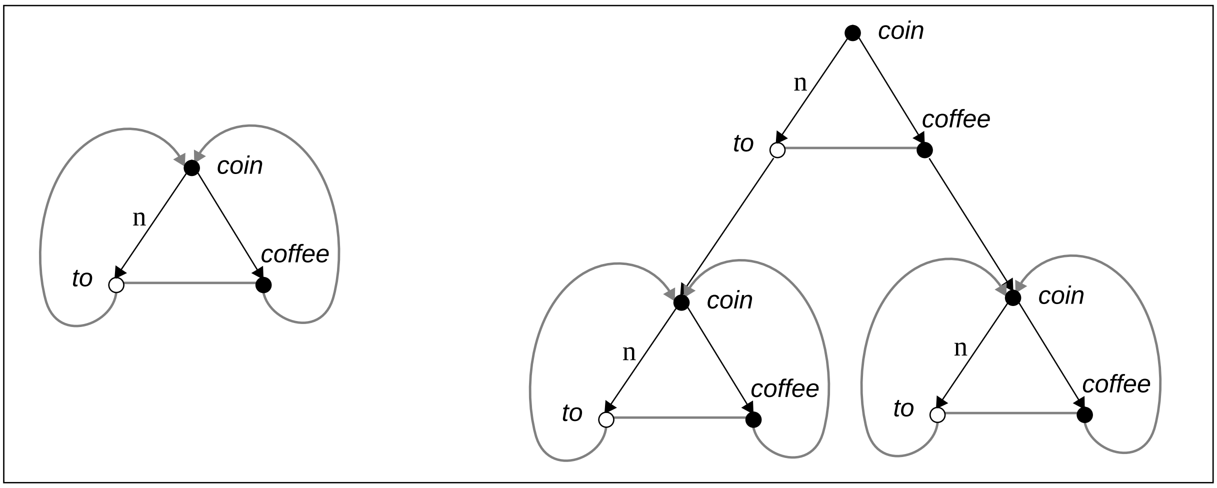


Figure 9.6: Unfoldings of the timed probabilistic event structure of U.

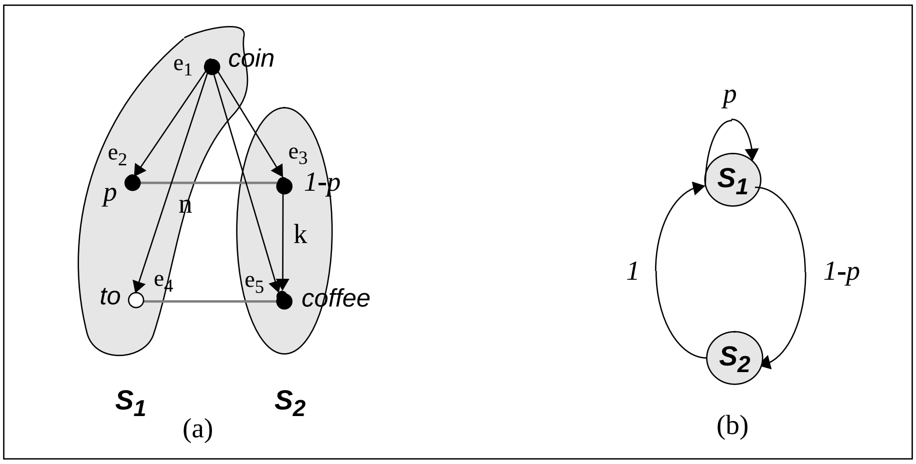


Figure 9.7: (a) Grouping of events and (b) a corresponding DTSMC.

Under the assumption that an event takes place as soon as it is enabled (maximal progress), we determine the average residence times as follows. From Figure 9.7(a) we deduce that k time units are spent in state s_2, so $r_2 = k$. For state s_1 there are two possibilities: if a transition is taken from s_1 to s_2 no time is spent in s_1, and if the system remains in state s_1 n time units are spent in s_1. The average residence time of s_1 thus becomes $r_1 = (1-p) \cdot 0 + p \cdot n$.

Using standard means we obtain for the limiting distribution π of the embedded DTMC[1]:

$$\pi_1 = \frac{1}{2-p} \quad , \pi_2 = \frac{1-p}{2-p} \quad .$$

Using Definition 9.29 and the average residence times determined just above we obtain for the

[1] Since all states are aperiodic it follows that the embedded DTMC of Figure 9.7(b) is regular (cf. Appendix A).

limiting distribution of the DTSMC:

$$\phi_1 = \frac{n \cdot p}{n \cdot p + k \cdot (1-p)} \quad , \phi_2 = \frac{k \cdot (1-p)}{n \cdot p + k \cdot (1-p)} \quad .$$

(Note that for $k{=}n$ one obtains $\phi_1 = p$ and $\phi_2 = 1{-}p$.) According to Definition 9.28 the average number T_i of time units between successive transitions to s_i equals r_i/ϕ_i. Since s_2 represents the successful case we obtain:

$$N_c = \frac{1}{T_2} = \frac{1-p}{n \cdot p + k \cdot (1-p)} \quad .$$

For $p \rightarrow 0$ the average number of time units between two coffee events approximates k, the time to produce coffee.

9.5.3 Illustrating locality

One of the main advantages of using a partial-order model for performance analysis was—as claimed in Chapter 1—the locality aspect, i.e., if one is interested in analyzing only part of a system it is relatively easy to do so without considering other (irrelevant) parts. To illustrate this we consider the following example:

$$\begin{aligned} &Q := ((1)\, c\, ; \surd \,|||\, (2)\, d\, ; \surd) \quad , \\ &R := (\tau\, ; (d_b)\ b\, ; \surd +_p \tau\, ; (d_a)\ a\, ; \surd) \quad , \text{and} \\ &P := (1)\ s\, ; (R \,|||\, Q) >> P \quad . \end{aligned}$$

Here, Q and R are independent processes that only synchronize their start and finish in each 'invocation' of P. R can autonomously choose whether to perform a b (with probability p) or to perform an a (with probability $1{-}p$). For the purpose of this example we assume that R is 'slower' than Q, i.e., $max(d_a, d_b) \geqslant 2$, and suppose we are interested in the average delay between two events labelled a (or b). Similar to the previous example we consider the timed probabilistic event structure corresponding to P (cf. Figure 9.8(a)) and group events appropriately—$s_1 = \{ e_1, e_2, e_6, e_8 \}$ and $s_2 = \{ e_3, e_7, e_9 \}$; note that events e_4 and e_5 do not belong to any group. The limiting distributions of the embedded DTMC (cf. Figure 9.8(b)) are:

$$\pi_1 = \frac{1}{2-p} \quad , \pi_2 = \frac{1-p}{2-p} \quad .$$

Using Definition 9.29 and the fact that $r_1 = p \cdot (1{+}d_b) + (1{-}p) \cdot 1$ and $r_2 = d_a$ we obtain for the limiting distribution ϕ of the DTSMC:

$$\phi_1 = \frac{1 + d_b \cdot p}{1 + d_a + (d_b - d_a) \cdot p} \quad , \phi_2 = \frac{d_a \cdot (1-p)}{1 + d_a + (d_b - d_a) \cdot p} \quad .$$

The average delay between two a events equals T_2, the average time between successive transitions to s_2. Using Definition 9.28 we get:

$$T_2 = \frac{r_2}{\phi_2} = 1 + d_a + \frac{(d_b + 1) \cdot p}{1-p} \quad .$$

For $p \to 0$ the average delay reaches $1+d_a$, which is optimal; for $p \to 1$ the average delay approximates ∞ and a's are never generated.

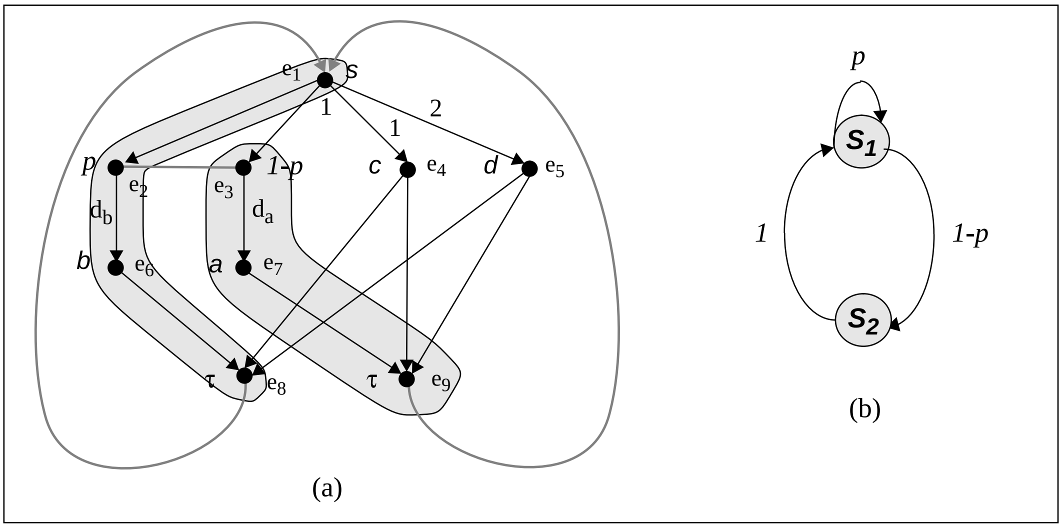


Figure 9.8: (a) Timed probabilistic event structure of P and (b) a corresponding DTSMC.

Observe that the average delay between two subsequent a's is analyzed without considering the—for this purpose—irrelevant process Q (more precisely, events e_4 and e_5). This seems reasonable as only R is involved in generating a events. Here we claim that this 'locality' aspect is a direct consequence of the distinction between parallel composition and nondeterminism in the probabilistic model. (The corresponding labelled transition system consists of 54 states, and includes 9 transitions labelled a.)

9.6 Related and further work

Probabilistic process algebras have been studied quite extensively in the literature. Probabilistic extensions of different process algebras have been proposed, such as ACP (by Baeten *et al.* [8]), CCS (by, amongst others, Christoff [34] and Hansson & Jonsson [65]), CSP (by Lowe [96, 97] and Seidel [135]), LOTOS (by, amongst others, Miguel *et al.* [102], Rico & von Bochmann [129], Sisto *et al.* [138], and recently Núñez & de Frutos [115]), and synchronous CCS (by Giacalone *et al.* [48], Van Glabbeek *et al.* [53] and Tofts [141]). For overviews of probabilistic process algebras we refer to the theses of Christoff [35] and Hansson [64]. The models underlying most of these process algebras are labelled transition systems in which probabilities are associated with transitions. To our knowledge PA_P is the first probabilistic process algebra with a noninterleaving semantics. In this section we discuss and compare several characteristics of our work with that in the literature.

9.6.1 Nondeterminism, probabilistic choice and parallel composition

In order to be able to specify 'real' nondeterminism and probabilistic nondeterminism we have chosen to equip PA_P with both a standard and probabilistic choice (see also the discussion in Section 9.3.1). Several probabilistic process algebras replace the standard choice by a probabilistic one, usually $+_{1/2}$. Since in an interleaving setting for finite processes parallel composition can be reduced to choice using the expansion law, parallel composition implicitly becomes probabilistic! For instance,

$$a \,|||\, b = a\,;\, b \;+_{1/2}\; b\,;\, a \quad .$$

In probabilistic ACP of Baeten *et al.* [8] parallel composition becomes even explicitly probabilistic. There, $P \,||_G^{p,q}\, Q$ denotes a process in which an interaction between P and Q happens with probability $1-q$, and an autonomous action of either P or Q with probability q. Given that an autonomous action occurs, P will perform such action with probability p and Q with probability $1-p$. A form of probabilistic parallel composition operator, where only the latter probability (p) is indicated, is proposed for LOTOS by Sisto *et al.* [138], and independently by Núñez & de Frutos [115]. We believe that probabilistic information is typically associated with alternatives in a specification, one excluding the other. Imposing a probability on causally independent events—like those resulting from parallel composition—seems not desirable from a design point of view, since it disturbs their independence.

9.6.2 Related approaches

Other models that do incorporate both a standard and probabilistic choice operator, and besides require probabilistic choices to be independent from the environment—like we do—can be found in [65, 96, 102, 45].

Hansson & Jonsson [65, 64] distinguish in their timed probabilistic variant of CCS, called TPCCS, between probabilistic (P) and action (A) states such that these two types of states strictly *alternate*. In action states outgoing transitions possibly involve the participation of the environment, but in probabilistic states they do not. This implies that probabilistic moves are always performed autonomously. In our operational semantics we also distinguish between A- and P-states, but do not require them to strict alternate.

Lowe [96] distinguishes between three types of states: action states (A), from which the process may evolve by performing observable actions; probabilistic states (P), from which the process may evolve probabilistically; and nondeterministic states (N) from which the process may evolve nondeterministically. Lowe uses the resulting NPA transition systems (or graphs) as a semantical model for a probabilistic variant of CSP. He allows only internal probabilistic choices because 'we do not believe that a probabilistic external choice is particularly useful in its own right'. Lowe showed that none of the standard semantical models for CSP (like Hoare traces and failures) can be extended to cover both $+_p$ and $+$, and concluded that 'it seems very hard to combine the two phenomena' [97].

LOTOS-P, the probabilistic version of LOTOS proposed by Miguel *et al.* [102], models stochastic experiments as internal actions. **random** x **in** B denotes a behaviour B possibly containing free occurrences of variable x, where x is the outcome of a realization of an experiment. For instance, an unreliable channel that may lose messages can be specified as

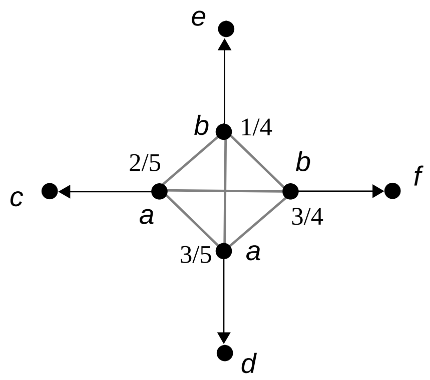
$$Chan := in\,;\ \mathbf{random}\ x\ \mathbf{in}\,([x] \rightarrow out\,;\ Chan + [\neg\, x] \rightarrow Chan).$$

Here it is assumed that x models the outcome of an experiment with two possible outcomes: true or false. Each possible outcome is represented by a transition labelled τ. In this way experiments are obtained that are independent from the environment.

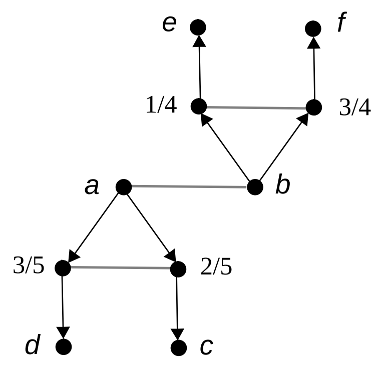
Fang *et al.* [45] present a probabilistic process algebra, called $PPARTY^i$, where probabilities are associated with internal activities of a process. Probabilities are linked to time by forcing that a probabilistic transition takes one unit of time. They do, however, incorporate a (binary) parallel composition operator $|$, where $B_1 \mid B_2$ terminates as soon as either B_1 or B_2 terminates. As a result, for instance, $a \mid (\tau +_p \tau)$ will never resolve the probabilistic choice, since a is first forced to occur (normal transitions have priority over probabilistic ones) which results in the termination of the entire process.

9.6.3 Reactive, generative, and stratified models

Several models allow a probabilistic choice to depend on the environment, in the sense that the probability of choosing one alternative or the other may depend on interactions with the environment. There are different ways in which to resolve such probabilistic interactions. Van Glabbeek *et al.* [53] consider three approaches: *reactive*, *generative* and *stratified*; in decreasing order of abstractness. In the generative case the entire set of alternatives in a state is equipped with a single probability distribution. The probabilities are conditioned on the set of actions accepted by the environment. Choices involving possibly different actions are resolved probabilistically. In the reactive model a separate probability distribution is associated with each action, and choices between different actions are resolved by the environment. (We do not discuss the stratified model here.) In a similar way as pointed out by Hansson [64] our model can be considered to fit within the realm of the reactive models. For example consider the following probabilistic variants of event structures:



(a)



(b)

(a) represents a reactive probabilistic process which initially can either perform an event labelled a or b.[2] (b) represents the corresponding event structure in EBES_P. If a is performed both event structures will with probability $\frac{2}{5}$ be able to perform an event labelled c and with probability $\frac{3}{5}$ an event labelled d. A similar reasoning applies to the case when b is performed.

9.6.4 Compatibility with nonprobabilistic semantics

Given an expression $B \in \mathsf{PA}_P$ and its nonprobabilistic image $\Phi_P(B)$ we have the nice result that omitting the probability information in $\mathcal{E}_P[\![B]\!]$, the probabilistic event structure corresponding to B, results in exactly the 'plain' event structure semantics of $\Phi_P(B)$. Thus, the semantics of PA_P is a complete *conservative extension* of the semantics of PA. A similar result has been reported for LOTOS-P [102], the probabilistic variant of LOTOS in [138], and probabilistic ACP [8]. It is interesting to note that for the interleaving semantics for a subset of PA_P (using identical syntactical constraints as we have) in [91] such result is not obtained—Langerak & Latella could only prove the transition system of $\Phi_P(B)$ and the transition system obtained by removing the probabilities from the probabilistic transition system of B to be testing equivalent.

9.6.5 Further work

Probabilistic event structures can be seen as a causality-based denotational model for system behaviour involving probabilities. An issue for further study is to see how to obtain from the causality-based semantics of PA_P more abstract semantics in the form of equivalences (congruences) and pre-orders (pre-congruences) that would reflect natural notions of transformation and implementation for probabilistic systems well.

Another direction to extend this work would be a further enhancement of expressive power. Interesting topics from an application point of view would be to allow for the assignment of probabilities to noninternal events (for instance, in the reactive sense), to work with intervals of probabilities, as can be found in Wang [150], or to incorporate an operator like $[>_p$ that

[2] Evidently, this is not a probabilistic event structure; for the sake of this example we allow probabilities to be assigned to noninternal events and are not restricted by the cluster concept.

allows for the quantification of the probability a behaviour is disrupted by another one, as can be found in Sisto *et al.* [138]. We believe that for $[>_p$ a probabilistic extension of $\rightsquigarrow$ would be appropriate; the interpretation of $e \overset{p}{\rightsquigarrow} e'$ being that e will be disabled by e' with probability p once both e and e' are enabled.

We have illustrated the use of our semantic model to obtain a performance model in the form of a discrete-time semi-Markov chain in two simple examples. There, the explicit presence of parallelism in the semantics helps in obtaining the performance model. It should be noted, however, that this connection is most readily exploited in the form of graphs (as used in the example), whereas the semantics of infinite behaviours is in reality given by infinite event structures (see Chapter 10). Under a regularity assumption, which applies in the case of tail recursion as used in the examples, such infinite structures can be finitely represented by graphs, which are subsequently transformed into performance models. It would be most interesting and useful, however, to represent infinite behaviour directly in terms of such a graph-based semantics. A first attempt in this direction can be found in Latella [93]. Although the structure of a performance model ultimately depends on the performance metrics one is interested in, such graph models could be a basis to study generic transformations to obtain Markov-like performance models from them in a systematic way, and guidelines and heuristics for applying them. Certainly, application of our method should first be attempted on larger, more realistic examples (e.g. broadband networks, multi-media), to develop a better feeling for what is really required.

We have addressed the use of probabilities in our deterministic timed model and concluded that under a simple additional constraint on the timing of cluster-events, clusters remain to correspond to stochastic experiments. We believe that an analogous constraint would also do in the stochastic setting of the previous chapter. It has recently been argued by Brinksma [27] that in the realm of stochastic process algebras different choices exist: the 'structural' choice ($+$), and the 'capacitive' choice (denoted here as $\oplus$) which reflects the more usual interpretation of choice constructs in performance models like CTMCs (see, for instance, Hillston [72]). $\oplus$ can be characterized as

$$(F)\ a\,;\, B_1 \oplus (G)\ a\,;\, B_2 = (F \cdot G)\ a\,;\,(B_1\ +_p\ B_2)$$

where $p = Pr\{\, U_F < U_G \,\}$. (Note that $+_p$ is an internal choice here.) Incorporating $+_p$ in PA_{GS}, the stochastic process algebra of Chapter 8, would enable to express both $\oplus$ and $+$ in a causality-based framework.

9.7 Conclusions

In this chapter we have developed a way of specifying probabilistic behaviour in (extended bundle) event structures. We have defined the notion of cluster, a set of internal, mutually conflicting events that have identical enablings and disablings. An event structure which only assigns probabilities to events in a cluster in such a way that the sum of these probabilities for each cluster equals 1 is referred to as a probabilistic event structure. By assigning probabilities in this way clusters represent stochastic experiments, the outcome of which can be determined independently from the environment. We considered the status of a probabilistic event structure after the execution of a set of events and defined a probability measure for sets of configurations. The mixture of deterministic time and probabilities has been investigated. PA has been equipped with a probabilistic (internal) choice operator $+_p$, $p \in (0,1)$, such that $B_1 +_p B_2$ nondeterministically behaves like B_1 with probability p or like B_2 with probability $1-p$. The resulting formalism, PA_P is assigned a causality-based semantics which is proven to be a conservative extension of the semantics of PA. A corresponding event-based operational semantics is presented which is shown to be testing equivalent to an 'interleaving' view of the noninterleaving semantics. Finally, we have exemplified how a performance model could be obtained from a (timed) probabilistic event structure.

10 Recursion

In order to specify real-life systems, recursion is a vital ingredient of any specification formalism. This chapter provides an event structure semantics for recursively defined processes. We consider the timed, real-time, urgent, and the probabilistic variant, and show that the stochastic case can be taken into account by a straightforward generalization of the deterministic timed case. Recursion is dealt with using the well-known standard domain theory. A complete partial order is defined on each type of event structure and all operators on these structures (which correspond to operators in the related process algebra) are shown to be continuous w.r.t. this partial order. The semantics of $P := B$ is then defined as the limit of a series of better and better approximations. Finally, for PA_T, PA_R, PA_U and PA_P we give an event-based operational semantics for recursively defined processes and prove the consistency of this operational semantics and the denotational causality-based semantics.

10.1 Introduction

In order to specify practically meaningful systems, recursion is indispensable. Until so far, the different models introduced in this thesis do not incorporate a mechanism to cope with recursion. The—quite standard—way to incorporate recursion is to extend the syntax of the process algebra at hand with the construct $B ::= P$, where P is a *process identifier*, and to assume a behaviour to appear in a context of a finite set of *process definitions* of the form $P := B$, where B (the body) is a behaviour that possibly contains occurrences of P (or other process identifiers). Occurrences of process identifiers in body B are referred to as *process instantiations*.

A simple recursive specification is $P := a\,;\,P$ which specifies a behaviour that infinitely many times can perform action a. In this chapter we consider the event structure semantics of recursive process definitions. That is, for $P := B$ we are looking for event structures that satisfy equations of the form $\mathcal{E} = \mathcal{F}_B(\mathcal{E})$. For the example above, it is clear that an event structure consisting of infinitely many events e_n with $e_n \mapsto e_{n+1}$ for all $n \geqslant 1$, all labelled a, is a solution. To obtain an event structure for arbitrary recursive process definitions, is, however, not so evident.

Fortunately, there is a well-established piece of theory, referred to as *domain theory*, that deals with the problem of constructing a denotational semantics for recursive definitions (see e.g. the treatments of Manna *et al.* [100], Tennent [139], Gunther & Scott [63] and Schmidt [132]).

The basic notions and results from domain theory as used in this chapter are summarized in Appendix B. Domain theory can be applied to our setting as follows.

As stated above we are looking for an event structure $\mathcal{E}$ that solves $\mathcal{E} = \mathcal{F}_B(\mathcal{E})$. That is, $\mathcal{E}$ is a *fixed point* of $\mathcal{F}_B$. Here $\mathcal{F}_B$ is a function that substitutes an event structure for each occurrence of P in B, interpreting all operators in B as operators on event structures. For example, for $P := a\,;P$ the result is $\mathcal{F}_B(\mathcal{E}) = \overline{a\,;}\,\mathcal{E}$, where $\overline{a\,;}$ is an operator that 'prefixes' an event structure with an event labelled a.

From domain theory it is known that fixed points can be determined once it is known that $\mathcal{F}_B$ is *continuous* w.r.t. a pointed complete partial order (denoted $\trianglelefteq$) on event structures. Let us first consider the order and then deal with continuity. A *pointed complete partial order* (pointed c.p.o.) is a partial order with a least element, usually denoted $\perp$, such that each totally ordered set (i.e., chain) of event structures has a least upper bound (l.u.b.). For chain $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \ldots$ the l.u.b. is denoted $\bigsqcup_i \mathcal{E}_i$. $\mathcal{F}_B$ is continuous w.r.t. $\trianglelefteq$ if and only if it preserves l.u.b.'s:

$$\mathcal{F}_B(\bigsqcup_i \mathcal{E}_i) = \bigsqcup_i \mathcal{F}_B(\mathcal{E}_i) \quad .$$

Preservation of l.u.b.'s means that applying $\mathcal{F}_B$ on the l.u.b. of a chain $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \ldots$ is identical to determining the l.u.b. of the chain $\mathcal{F}_B(\mathcal{E}_1) \trianglelefteq \mathcal{F}_B(\mathcal{E}_2) \trianglelefteq \ldots$. In general, preservation of l.u.b.'s is not straightforward to prove. However, under the condition that two ordered event structures with identical sets of events are identical it suffices, by a nice result of Winskel [155], to prove *continuity on events* (which is easier) rather than continuity in the above sense. For the models in this dissertation this condition applies (as proven in this chapter) and we can adopt Winskel's approach. $\mathcal{F}_B$ is continuous on events if and only if it is *monotonic*, that is,

$$\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \;\Rightarrow\; \mathcal{F}_B(\mathcal{E}_1) \trianglelefteq \mathcal{F}_B(\mathcal{E}_2)$$

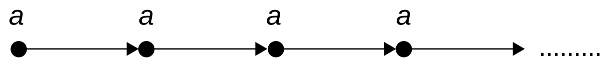
and, for each chain $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \ldots$

$$E\left(\mathcal{F}_B(\bigsqcup_i \mathcal{E}_i)\right) \subseteq E\left(\bigsqcup_i \mathcal{F}_B(\mathcal{E}_i)\right) \quad .$$

Here $E(\mathcal{E})$ denotes the set of events of $\mathcal{E}$. For example, $\overline{a\,;}$ is continuous on events (and so, continuous w.r.t. $\trianglelefteq$) iff (i) it is monotonic—'prefixing' an event to $\mathcal{E}_1$ which is smaller than $\mathcal{E}_2$ should result in a smaller event structure than 'prefixing' the event to $\mathcal{E}_2$—and (ii) the set of events of e_a prefixed to l.u.b. $\bigsqcup_i \mathcal{E}_i$ is a subset of the set of events of the l.u.b. of the chain obtained by prefixing each $\mathcal{E}_i$ with e_a.

Given a pointed c.p.o. and a function that is continuous it is known from domain theory that the set $\{\, \mathcal{E} \mid \mathcal{F}_B(\mathcal{E}) = \mathcal{E} \,\}$ has a *least* element, referred to as the *least fixed point*, which is unique and equals $\bigsqcup_i \mathcal{F}_B^i(\perp)$, for $i \geqslant 0$. So, the equation $\mathcal{E} = \mathcal{F}_B(\mathcal{E})$ can be solved by means of *approximation*. That is, $\mathcal{E}$ is approximated, starting with the 'worst' approximation $\perp$, then $\mathcal{F}_B(\perp)$, which—by monotonicity—approximates $\mathcal{F}_B(\mathcal{F}_B(\perp))$, and so on. $\perp$, $\mathcal{F}_B(\perp)$, $\mathcal{F}_B(\mathcal{F}_B(\perp)), \ldots$ is a sequence of better and better approximations which, by continuity of $\mathcal{F}_B$, converges to a limit $\bigsqcup_i \mathcal{F}_B^i(\perp)$.

For $\mathcal{F}_B(\mathcal{E}) = \overline{a\,;}\,\mathcal{E}$ we start the approximation with the empty event structure. In each successive approximation we now extend the previously obtained event structure with a new event labelled a pointing to the initial event(s) of this structure, and as a result, the l.u.b. of this sequence will be an event structure consisting of an infinite chain of equally labelled events (with label a):



In this chapter the above procedure is applied to timed, real-time, urgent, stochastic and probabilistic event structures. In this way, we obtain a noninterleaving semantics for PA_T, PA_R, PA_U, PA_{GS} and PA_P that includes recursion. The event-based operational semantics of $\mathsf{PA}_T, \mathsf{PA}_R, \mathsf{PA}_U$ and PA_P is extended with recursion and consistency between this operational semantics and the denotational causality-based semantics is proven.

From the above description it is clear that the semantics of $P := B$ may result in an event structure of infinite size, i.e., with an infinite number of events. As a result bundles of infinite size and an infinite number of conflicts can appear. Until so far, our event structure models have been *finite*, but there are no severe difficulties in extending this to infinite event structures; only in case of timed event structures we need to adapt the definition of time appropriately. In this chapter it is assumed that infinite event structures can appear.

This chapter is further organized as follows. In Section 10.2 we start by recapitulating the most important definitions and results of Langerak [89] concerning a pointed c.p.o. on extended bundle event structures and the denotational semantics of $P := B$ where $B \in \mathsf{PA}$. Section 10.3 considers recursive process definitions in PA_T. Sections 10.4, 10.5 and 10.6 do the same for PA_U, PA_R, and PA_{GS}, respectively. Section 10.7 considers recursion in the probabilistic setting. Sections 10.4, 10.3 and 10.7 also consider the extension of the event-based operational semantics of PA_T and PA_R, PA_U, and PA_P with recursion. Section 10.8 presents the conclusions of this chapter.

10.2 Extended bundle event structures

This section introduces a pointed c.p.o. on extended bundle event structures, explains the approach of [89, Chapter 8], and summarizes the main results. Section 10.2.1 introduces the pointed c.p.o. $\trianglelefteq$, provides a characterization of the l.u.b. of a chain of event structures ordered under $\trianglelefteq$, and presents some properties of this ordering and its limits. Section 10.2.2 considers the function $\mathcal{F}_B$ (see Section 10.1), proves continuity w.r.t. $\trianglelefteq$ for all operators on event structures and defines the denotational semantics of $P := B$ for $B \in \mathsf{PA}$.

10.2.1 A pointed complete partial order

10.1. DEFINITION. (*Partial order on extended bundle event structures*)

Let $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2. Then $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ iff

1. $E_1 \subseteq E_2$
2. $\rightsquigarrow_1 = \rightsquigarrow_2 \cap (E_1 \times E_1)$
3. $\mapsto_1 = \{ ((X \cap E_1), e) \mid e \in E_1 \ \wedge \ X \mapsto_2 e \}$
4. $l_1 = l_2 \upharpoonright E_1$.

□

where $\upharpoonright$ denotes restriction. It is straightforward to verify that $\trianglelefteq$ is a partial order. The constraint $E_1 \subseteq E_2$ is self-explanatory. For conflicts we require that no new conflicts appear in $\mathcal{E}_2$ between events that are already in $\mathcal{E}_1$. Similarly, the third constraint forbids the introduction of bundles in $\mathcal{E}_2$ pointing to events in $\mathcal{E}_1$ for which there exists no projected bundle in $\mathcal{E}_1$. Note that this constraint allows for bundles to grow in such a way that the old bundle is contained in the new one.

10.2. LEMMA. $\langle \mathsf{EBES}, \trianglelefteq \rangle$ is a pointed c.p.o..

PROOF. Routine and omitted. □

It is easy to show that $\perp = (\varnothing, \varnothing, \varnothing, \varnothing)$, the empty bundle event structure, is the least element under $\trianglelefteq$.

10.3. EXAMPLE. Consider the event structures of Figure 10.1, referred to as (a) $\mathcal{E}_1$, (b) $\mathcal{E}_2$, (c) $\mathcal{E}_3$ and (d) $\mathcal{E}_4$, and assume equally labelled events to be identical. We have $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ since $E_1 \subseteq E_2$, $\rightsquigarrow_1 = \rightsquigarrow_2 \cap (E_1 \times E_1)$, and $(\{ e_a, e_c \} \cap E_1) \mapsto_1 e_b$. It is also easy to check that $\mathcal{E}_2 \trianglelefteq \mathcal{E}_3$ (and, since $\trianglelefteq$ is a partial order, $\mathcal{E}_1 \trianglelefteq \mathcal{E}_3$). Since $\{ e_a, e_d \} \mapsto_4 \{ e_b \}$, but $(\{ e_a, e_d \} \cap E_2) \not\mapsto_2 \{ e_b \}$ we have $\mathcal{E}_2 \not\trianglelefteq \mathcal{E}_4$. □

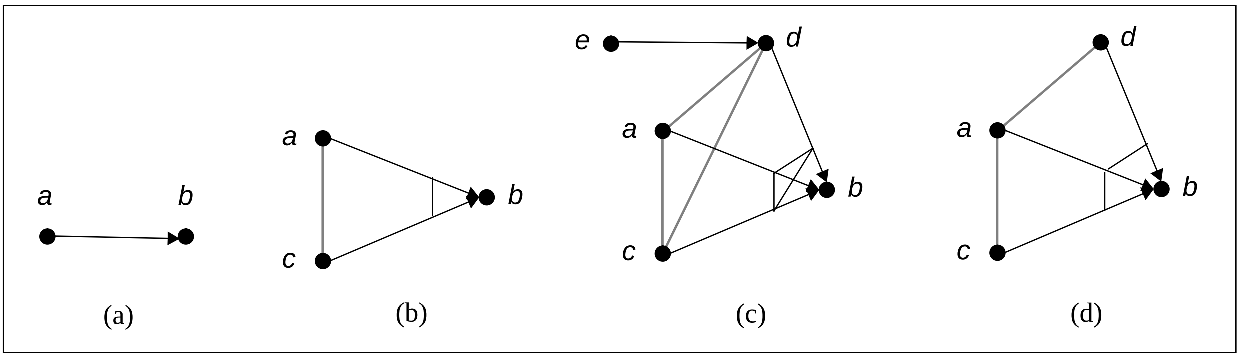


Figure 10.1: Extended bundle event structures with (a) $\trianglelefteq$ (b) $\trianglelefteq$ (c), but (b) $\not\trianglelefteq$ (d).

For chain $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \ldots$ let event structure $\bigsqcup_i \mathcal{E}_i$ be defined as follows. For the set of events and conflicts, and the labelling function, we simply take the union of all events, conflicts and labellings of the event structures in the chain. As bundles may grow this approach does not apply to the set of bundles. Suppose some $\mathcal{E}_j$ has bundle $X_j \mapsto_j e$. According to the definition of $\trianglelefteq$ there is a series of bundles $X_j \mapsto_j e$, $X_{j+1} \mapsto_{j+1} e, \ldots$ satisfying $(X_{k+1} \cap E_k) = X_k$ for $k \geqslant j$. Then $\bigsqcup_i \mathcal{E}_i$ has bundle $(\bigcup_n X_{j+n}) \mapsto e$.

10.4. DEFINITION. (*Least upper bound (under $\trianglelefteq$)*)

Let $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \ldots$ be a chain, then $\bigsqcup_i \mathcal{E}_i \triangleq (\bigcup_i E_i, \bigcup_i \rightsquigarrow_i, \mapsto, \bigcup_i l_i)$ with

$$\mapsto = \{ (\bigcup_k X_k, e) \mid \exists\, j : (\forall\, k \geqslant j : X_k \mapsto_k e \ \wedge\ X_{k+1} \cap E_k = X_k) \}.$$

□

10.5. LEMMA. $\bigsqcup_i \mathcal{E}_i$ is the least upper bound of chain $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \ldots$.

PROOF. See [89, Theorem 8.2.5]. □

Some important and useful properties are listed in the following theorem. The fact that a 'larger' event structure allows more event traces is stated in the first part of the theorem. So, $\trianglelefteq$ preserves sets of event traces. The second part of the theorem states that ordered event structures with identical sets of events are identical. As we will see in Lemma 10.11 this property is essential to prove that continuity (w.r.t. $\trianglelefteq$) boils down to continuity on events. The third part of the theorem says that the set of traces of the l.u.b. is simply the union of the sets of traces of the elements of the corresponding chain.

10.6. THEOREM. Let $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2.

1. $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \ \Rightarrow\ T(\mathcal{E}_1) \subseteq T(\mathcal{E}_2)$.
2. $(\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \ \wedge\ E_1 = E_2) \ \Rightarrow\ \mathcal{E}_1 = \mathcal{E}_2$.
3. $T(\bigsqcup_i \mathcal{E}_i) = \bigcup_i T(\mathcal{E}_i)$.

PROOF. See [89, Section 8.2]. □

The following result is used in the next sections. Let $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2.

10.7. LEMMA. $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \ \Rightarrow\ \mathit{init}(\mathcal{E}_2) \cap E_1 = \mathit{init}(\mathcal{E}_1)$.

PROOF. '$\subseteq$': by contradiction. Suppose $e \in \mathit{init}(\mathcal{E}_2) \cap E_1$, but $e \notin \mathit{init}(\mathcal{E}_1)$. From $e \notin \mathit{init}(\mathcal{E}_1)$ we infer that $(\exists\, X_1 \subseteq E_1 : X_1 \mapsto_1 e)$. But then, since $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ there exists $X_2 \mapsto_2 e$ (with $X_2 \cap E_1 = X_1$). This contradicts with $e \in \mathit{init}(\mathcal{E}_2)$.

'$\supseteq$': by contradiction. Suppose $e \in \mathit{init}(\mathcal{E}_1)$ but $e \notin \mathit{init}(\mathcal{E}_2) \cap E_1$. Since $e \notin \mathit{init}(\mathcal{E}_2)$ we have $(\exists\, X_2 \subseteq E_2 : X_2 \mapsto_2 e)$. From $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ and $e \in E_1$ we have $(X_2 \cap E_1) \mapsto_1 e$, contradicting $e \in \mathit{init}(\mathcal{E}_1)$. □

10.8. LEMMA. For σ a sequence of events in $\mathcal{E}_1$: $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \ \Rightarrow\ \mathsf{en}_2(\sigma) \cap E_1 = \mathsf{en}_1(\sigma)$.

PROOF. Straightforward and omitted. □

10.2.2 A fixed point semantics

In this section we define an event structure semantics for recursive process definitions of the form $P := B$, where B possibly contains occurrences of P. These occurrences of P in B are called *process instantiations*. For the sake of simplicity we restrict ourselves to single recursive definitions using just one process variable (that is, $P := \ldots P \ldots P \ldots$). As shown by, amongst others, Manna *et al.* [100] the generalization to a set of process definitions ($P := \ldots Q \ldots P \ldots$ and $Q := \ldots P \ldots Q \ldots$) is rather straightforward.

Like in Chapter 5 we assume all action prefix and $\surd$ occurrences to be subscripted with a Greek letter. In addition, each process instantiation is uniquely identified in the same way. For instance, $P := a\,;\,P + b\,;\,P$ becomes $P := a_\xi\,;\,P_\phi + b_\chi\,;\,P_\psi$. The occurrence identifiers are required to be globally unique.

Consider $P := B$ and let the event structure corresponding to P be denoted $\mathcal{E}$. Then the objective is to find a characterization of $\mathcal{E}$. The idea is to define a function $\mathcal{F}_B$ that substitutes an event structure for each occurrence of P in B, interpreting all operators in B as operators on event structures. To guarantee unique event names in the result of this substitution procedure each event in the event structure corresponding to P_ϕ, a process instantiation in B, is prefixed by ϕ. So, if $\mathcal{E}$ is the event structure corresponding to P, P_ϕ is replaced by $\phi(\mathcal{E})$, the structure obtained from $\mathcal{E}$ by replacing each event name e in E by ϕe and adjusting $\leadsto$, $\mapsto$ and l in an appropriate way. This renaming of event structures is formalized as follows.

10.9. DEFINITION. For $\mathcal{E} = (E, \leadsto, \mapsto, l)$ and ϕ an occurrence identifier let

$$\phi(\mathcal{E}) \triangleq (\phi E, \leadsto', \mapsto', l')$$

with $\phi E = \{\, \phi e \mid e \in E \,\}$, $\phi e \leadsto' \phi e'$ iff $e \leadsto e'$, $\phi X \mapsto' \phi e$ iff $X \mapsto e$ and $l'(\phi e) = l(e)$. □

As a second step towards the definition of $\mathcal{F}_B$ all operators in B (like ;, +, >>, ...) must be interpreted as operators on event structures. In Chapter 2 we have defined an event structure semantics of PA. Since this definition is compositional we have in fact implicitly defined operators on event structures. For example, $\mathcal{E}[\![B_1 + B_2]\!] = \mathcal{E}[\![B_1]\!] \overline{+} \mathcal{E}[\![B_2]\!]$ where $\overline{+}$ denotes the choice operator on event structures (rather than on expressions), and $\mathcal{E}[\![a_\xi\,;\,B]\!] = \overline{a_\xi\,;}\,\mathcal{E}[\![B]\!]$. In the sequel we denote for operator $\mathsf{op} \in \mathsf{PA}$ the corresponding counterpart on event structures by $\overline{\mathsf{op}}$.

Function $\mathcal{F}_B$ for $P := B$ replaces all occurrences P_ϕ in B by $\phi(\mathcal{E})$ and interprets all operators op in B as operators $\overline{\mathsf{op}}$ on (the substituted) event structures. E.g., for

$$P := a_\xi\,;\,P_\phi \,||_a\, (a_\chi\,;\,P_\psi + c_\upsilon\,;\,\mathbf{0})$$

$\mathcal{F}_B(\mathcal{E})$ is defined as

$$\mathcal{F}_B(\mathcal{E}) = \overline{a_\xi\,;}\,\phi(\mathcal{E}) \,\overline{||_a}\, (\overline{a_\chi\,;}\,\psi(\mathcal{E}) \,\overline{+}\, \overline{c_\upsilon\,;}\,\overline{\mathbf{0}}) \quad .$$

We will not bother the reader with the full definition of $\mathcal{F}_B$ here. The important thing now is that $\mathcal{F}_B(\mathcal{E})$ can be considered as a function of $\mathcal{E}$. This enables the characterization of the event structure semantics of $P := B$ as the problem of finding a solution of the equation $\mathcal{F}_B(\mathcal{E}) = \mathcal{E}$. From Section 10.1 we recall that $\mathcal{E}$ can be determined by means of approximation if $\mathcal{F}_B$ is continuous w.r.t. $\trianglelefteq$. In order to prove that $\mathcal{F}_B$ is continuous it suffices to prove that its constituents, $\overline{\mathsf{op}}$ and $\phi()$ (Definition 10.9) are continuous, for all op. As suggested by Winskel [155] we prove continuity on a set of events rather than on a c.p.o.:

10.10. Definition. (*Continuity on events*)

Let $\langle \mathsf{EBES}, \trianglelefteq \rangle$ be a pointed c.p.o. and $F : \mathsf{EBES} \longrightarrow \mathsf{EBES}$. F is *continuous on events* iff F is monotonic and for any chain $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \ldots$ we have $E(F(\bigsqcup_i \mathcal{E}_i)) \subseteq E(\bigsqcup_i F(\mathcal{E}_i))$. □

Here, $E(\mathcal{E})$ for event structure $\mathcal{E}$ denotes the set of events of $\mathcal{E}$.

10.11. Lemma. For $\langle \mathsf{EBES}, \trianglelefteq \rangle$ and $F : \mathsf{EBES} \longrightarrow \mathsf{EBES}$ we have: F is continuous iff F is continuous on events.

Proof. We concentrate on the proof of $\Leftarrow$, the proof for the other part is trivial. Let F be continuous on events and let $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \trianglelefteq \ldots$ be a chain.

$$
\begin{array}{ll}
 & \forall\, i : \mathcal{E}_i \trianglelefteq \bigsqcup_i \mathcal{E}_i \\
\Rightarrow & \{\ F \text{ is monotonic }\} \\
 & \forall\, i : F(\mathcal{E}_i) \trianglelefteq F(\bigsqcup_i \mathcal{E}_i) \\
\Rightarrow & \{\ \bigsqcup_i F(\mathcal{E}_i) \text{ is the l.u.b. of } F(\mathcal{E}_1) \trianglelefteq F(\mathcal{E}_2) \trianglelefteq \ldots\ \} \\
 & \bigsqcup_i F(\mathcal{E}_i) \trianglelefteq F(\bigsqcup_i \mathcal{E}_i) \\
\Leftrightarrow & \{\ \text{definition of } \trianglelefteq\ \} \\
 & \bigsqcup_i F(\mathcal{E}_i) \trianglelefteq F(\bigsqcup_i \mathcal{E}_i) \ \wedge\ E(\bigsqcup_i F(\mathcal{E}_i)) \subseteq E(F(\bigsqcup_i \mathcal{E}_i)) \\
\Leftrightarrow & \{\ F \text{ is continuous on events }\} \\
 & \bigsqcup_i F(\mathcal{E}_i) \trianglelefteq F(\bigsqcup_i \mathcal{E}_i) \ \wedge\ E(\bigsqcup_i F(\mathcal{E}_i)) = E(F(\bigsqcup_i \mathcal{E}_i)) \\
\Rightarrow & \{\ \text{Theorem 10.6}\ \} \\
 & \bigsqcup_i F(\mathcal{E}_i) = F(\bigsqcup_i \mathcal{E}_i) \quad .
\end{array}
$$

This proves that F preserves l.u.b.'s and, so that F is continuous (see also Appendix B). □

10.12. Theorem. $\overline{a_\xi\,;}, \overline{+}, \overline{||_G}, \ldots$ and $\phi()$ are continuous on $\langle \mathsf{EBES}, \trianglelefteq \rangle$.

Proof. See [89, Theorem 8.3.8]. □

10.13. Definition. For $P := B$ a process definition let $\mathcal{E}[\![\, P \,]\!] \triangleq \bigsqcup_i \mathcal{F}_B^i(\bot)$. □

10.14. Example. As an example of the semantics of a recursive process definition, consider $P := a\,;\,(b\,;\ P{+}c\,;\,d\,;\ P)$. $\bot$ is the empty event structure. $\mathcal{F}_B(\bot)$ is depicted in Figure 10.2(a). By repeated substitution we obtain the event structure of Figure 10.2(b). □

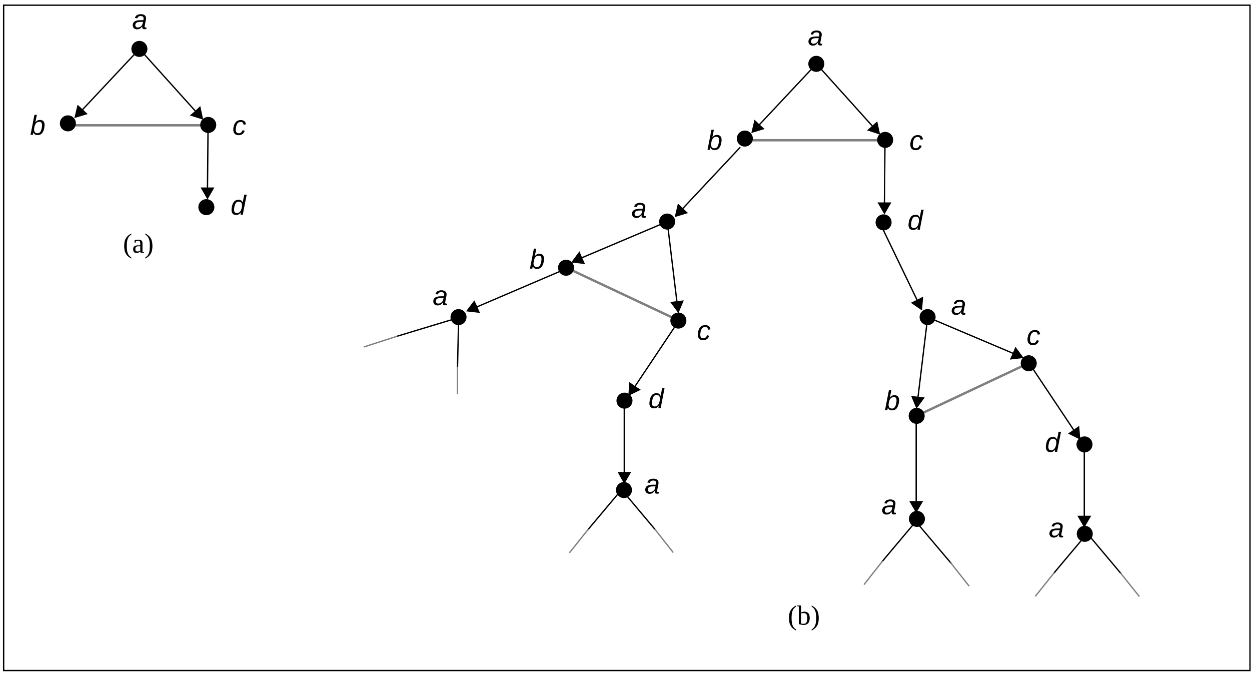


Figure 10.2: Example of semantics for a recursive process definition in PA.

10.3 Timed event structures

In this section we apply the approach of the previous section to timed event structures as introduced in Chapter 4. A partial order $\trianglelefteq_t$ on timed event structures is defined as a conservative extension of $\trianglelefteq$. The l.u.b. of a sequence of timed event structures is characterized as a straightforward generalization of the untimed case. These ingredients, introduced in Section 10.3.1, provide the basis for a fixed point semantics of PA_T. This semantics is presented in Section 10.3.2. In Chapter 5 we have proven the consistency between the causality-based semantics of PA_T and an event-based operational semantics based on timed actions. The extension of this study towards recursive behaviours is provided in Section 10.3.3.

10.3.1 A pointed complete partial order

We start by reconsidering the definition of time in Chapter 4. Since we now deal with event structures that potentially have an infinite number of events there maybe an infinite number of bundles pointing to an event. The enabling time of an event after trace σ was defined as the maximum of a set of time instants. In order to deal with sets of infinite size we adjust the definition as follows:

10.15. Definition. For σ a sequence of timed events $(e_1, t_1) \dots (e_n, t_n)$ with $e_i \in E$, $t_i \in \mathsf{Time}$ for $0 < i \leqslant n$, and $e \in \mathsf{en}([\sigma])$, let

$$\mathsf{time}(\sigma, e) \triangleq \mathit{Sup}(\{\, \mathcal{D}(e)\,\} \cup H_1 \cup H_2) \text{ where}$$
$$H_1 = \{\, t + t_j \mid \exists\, X \subseteq E : X \overset{t}{\mapsto} e \;\wedge\; X \cap \overline{[\sigma]} = \{\, e_j \,\}\,\}$$

$$H_2 = \{ t_j \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow e \} \quad .$$

□

Since infinite suprema cannot appear in our setting it suffices to consider finite suprema.

The definitions and theorems in this section are all relative to timed event structures $\Gamma_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i \rangle$ with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2.

10.16. Definition. (*Partial order on timed event structures*)

$\Gamma_1 \trianglelefteq_t \Gamma_2$ iff

1. $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$
2. $\mathcal{D}_2 \restriction E_1 = \mathcal{D}_1$
3. $\forall\, e \in E_1 : \mathcal{T}_2((X, e)) = \mathcal{T}_1((X \cap E_1, e))$.

□

In addition to the constraints for $\trianglelefteq$ (cf. Definition 10.1) we require that event delays of events that are already in Γ_1 are unaffected. Bundles can grow in such a way that the old bundle is contained (as in the untimed case) and the bundle delay is kept the same.

10.17. Lemma. $\langle \mathsf{EBES}_T, \trianglelefteq_t \rangle$ is a pointed c.p.o..

Proof. Routine and omitted. □

It is easy to show that $\bot_t = \langle \bot, \varnothing, \varnothing \rangle$, the empty timed event structure, is the least element under $\trianglelefteq_t$.

10.18. Example. Consider the timed event structures of Figure 10.3, referred to as (a) Γ_1, (b) Γ_2 and (c) Γ_3, and assume equally labelled events to be identical. We have that $\Gamma_1 \trianglelefteq_t \Gamma_2$, since $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ (see Example 10.3) and the timing of e_a, e_b and $\{ e_a \} \xmapsto{1} e_b$ is preserved. $\Gamma_2 \not\trianglelefteq_t \Gamma_3$, however, since Γ_3 violates the third constraint from Definition 10.16—the timing of bundle $\{ e_a, e_c, e_d \} \mapsto e_b$ should be 1 rather than 2 in order to let $\Gamma_2 \trianglelefteq_t \Gamma_3$. □

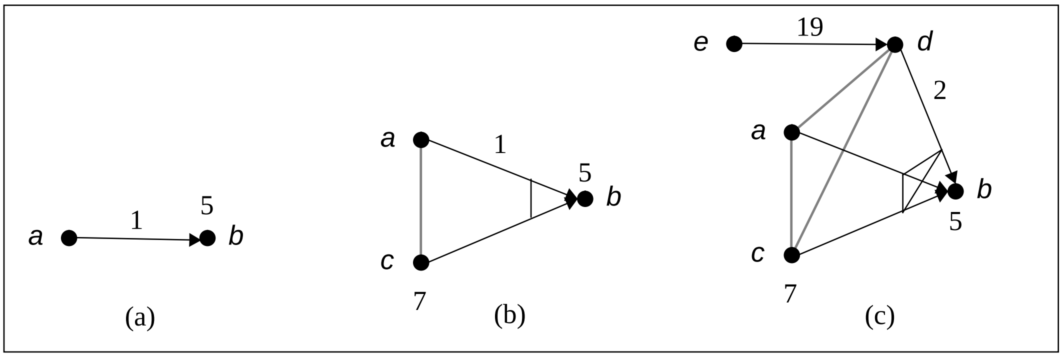


Figure 10.3: Timed event structures with (a) $\trianglelefteq_t$ (b), but (b) $\not\trianglelefteq_t$ (c).

The following lemma is needed to reduce continuity (w.r.t. $\trianglelefteq_t$) to continuity on events.

10.19. LEMMA. $(\Gamma_1 \trianglelefteq_t \Gamma_2 \ \wedge \ E_1 = E_2) \ \Rightarrow \ \Gamma_1 = \Gamma_2$.

PROOF. Assume $\Gamma_1 \trianglelefteq_t \Gamma_2$ and $E_1 = E_2$. We prove $\Gamma_1 = \Gamma_2$ component-wise:

1. $\Gamma_1 \trianglelefteq_t \Gamma_2 \ \wedge \ E_1 = E_2 \ \Rightarrow \ \mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \ \wedge \ E_1 = E_2 \ \Rightarrow \ \{$ Theorem 10.6 $\}$ $\mathcal{E}_1 = \mathcal{E}_2$.
2. $\mathcal{D}_1 = \mathcal{D}_2 \restriction E_1 = \mathcal{D}_2 \restriction E_2 = \mathcal{D}_2$.
3. $\Gamma_1 \trianglelefteq_t \Gamma_2 \ \Rightarrow \ \forall e \in E_1 : \mathcal{T}_2((X, e)) = \mathcal{T}_1((X \cap E_1, e)) \Leftrightarrow \{ E_1 = E_2 \} \ \mathcal{T}_2 = \mathcal{T}_1$.

□

For chain $\Gamma_1 \trianglelefteq \Gamma_2 \trianglelefteq \ldots$ let $\bigsqcup_i \Gamma_i$ be defined as follows. The untimed part is constructed according to Definition 10.4. The event delays are the union of all delays of the timed event structures in the chain. $\bigsqcup_i \Gamma_i$ contains bundles of the form $(\bigcup_n X_{j+n}) \mapsto e$ where $X_j \mapsto_j e$, $X_{j+1} \mapsto_{j+1} e, \ldots$ is a series of bundles satisfying $(X_{k+1} \cap E_k) = X_k$ for $k \geqslant j$. As all bundles in a series retain the same timing the bundle delay is the union of the bundle delays of the structures in the chain.

10.20. DEFINITION. (*Least upper bound (under $\trianglelefteq_t$)*)

Let $\Gamma_1 \trianglelefteq \Gamma_2 \trianglelefteq \ldots$ be a chain, then $\bigsqcup_i \Gamma_i \triangleq \langle \bigsqcup_i \mathcal{E}_i, \bigcup_i \mathcal{D}_i, \mathcal{T} \rangle$ with

$$\mathcal{T} = \{ \, ((\bigcup_k X_k, e), t) \mid \exists\, j : (\forall\, k \geqslant j : X_k \overset{t}{\mapsto}_k e \ \wedge \ X_{k+1} \cap E_k = X_k) \, \}.$$

□

10.21. LEMMA. $\bigsqcup_i \Gamma_i$ is the least upper bound of chain $\Gamma_1 \trianglelefteq_t \Gamma_2 \trianglelefteq_t \ldots$.

PROOF. The proof of this lemma is carried out in two parts. We first prove that $\bigsqcup_i \Gamma_i$ is an upper bound, that is, $\forall i \geqslant 0 : \Gamma_i \trianglelefteq_t \bigsqcup_i \Gamma_i$, and secondly, we prove that it is the *least* upper bound. Let $\bigsqcup_i \Gamma_i = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$.

1. $\forall i \geqslant 0 : \Gamma_i \trianglelefteq_t \bigsqcup_i \Gamma_i$. From Theorem 10.5 we have $\mathcal{E}_i \trianglelefteq \bigsqcup_i \mathcal{E}_i$. In addition, it easily follows that $\mathcal{D} \restriction E_i = (\bigcup_i \mathcal{D}_i) \restriction E_i = \mathcal{D}_i$. Let $X \mapsto e$ a bundle in $\bigsqcup_i \Gamma_i$ with $e \in E_i$; from the untimed case we know that $(X \cap E_i) \mapsto_i e$. Then:

$$\begin{array}{cl} & \mathcal{T}((X, e)) \\ = & \{ \text{ Definition 10.4 } \} \\ & \mathcal{T}((\bigcup_k X_k, e)) \\ = & \{ \text{ Definition 10.20 } \} \\ & \mathcal{T}_i((\bigcup_k X_k \cap E_i, e)) \quad . \end{array}$$

2. We prove by contradiction that $\bigsqcup_i \Gamma_i$ is the least upper bound under $\trianglelefteq_t$. Suppose there is another upper bound $\Gamma' = \langle \mathcal{E}', \mathcal{D}', \mathcal{T}' \rangle$ of the chain $\Gamma_1 \trianglelefteq_t \Gamma_2 \trianglelefteq_t \ldots$ such that $\Gamma' \trianglelefteq_t \bigsqcup_i \Gamma_i$. This means $E' \subseteq \bigcup_i E_i$. Since Γ' is an upper bound we have $E_i \subseteq E'$, for all i, so $\bigcup_i E_i \subseteq E'$. It follows that $\bigcup_i E_i = E'$. But then according to Theorem 10.19 $\Gamma' = \bigsqcup_i \Gamma_i$. Contradiction.

□

As a next result we prove that $\trianglelefteq_t$ preserves timed trace sets. It is technically convenient to have the following result:

10.22. LEMMA. Let σ a sequence of timed events in E_1 and $e \in \mathsf{en}([\sigma])$. Then:

$$\Gamma_1 \trianglelefteq_t \Gamma_2 \;\Rightarrow\; \mathsf{time}_1(\sigma, e) = \mathsf{time}_2(\sigma, e) \quad .$$

PROOF. Assume $\Gamma_1 \trianglelefteq_t \Gamma_2$, let σ be a sequence of timed events in E_1 and $e \in \mathsf{en}([\sigma])$. From Lemma 10.8 it follows $\mathsf{en}_1([\sigma]) = \mathsf{en}_2([\sigma]) \cap E_1$. Thus, $\mathsf{time}_1(\sigma, e)$ and $\mathsf{time}_2(\sigma, e)$ are both defined. Then:

$$\begin{array}{cl}
 & \mathsf{time}_1(\sigma, e) \\
= & \{\text{ definition of } \mathsf{time} \;\} \\
 & Sup(\{\,\mathcal{D}_1(e)\,\} \cup H_1 \cup H_2) \text{ where} \\
 & \quad H_1 = \{\, t{+}t_j \mid \exists\, X_1 \subseteq E_1 : X_1 \overset{t}{\mapsto}_1 e \;\wedge\; X_1 \cap \overline{[\sigma]} = \{\, e_j \,\}\,\} \text{ and} \\
 & \quad H_2 = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow_1 e \,\} \\
= & \{\; \Gamma_1 \trianglelefteq_t \Gamma_2 \text{ using } e \in E_1 \;\} \\
 & Sup(\{\,\mathcal{D}_2(e)\,\} \cup H_1 \cup H_2) \text{ where} \\
 & \quad H_1 = \{\, t{+}t_j \mid \exists\, X_2 \subseteq E_2 : X_2 \overset{t}{\mapsto}_2 e \;\wedge\; X_2 \cap E_1 = X_1 \;\wedge\; X_1 \cap \overline{[\sigma]} = \{\, e_j \,\}\,\} \text{ and} \\
 & \quad H_2 = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow_2 e \,\} \\
= & \{\; \overline{[\sigma]} \subseteq E_1 \;\} \\
 & Sup(\{\,\mathcal{D}_2(e)\,\} \cup H_1 \cup H_2) \text{ where} \\
 & \quad H_1 = \{\, t{+}t_j \mid \exists\, X_2 \subseteq E_2 : X_2 \overset{t}{\mapsto}_2 e \;\wedge\; X_2 \cap \overline{[\sigma]} = \{\, e_j \,\}\,\} \text{ and} \\
 & \quad H_2 = \{\, t_j \mid \exists\, e_j \in \overline{[\sigma]} : e_j \rightsquigarrow_2 e \,\} \\
= & \{\text{ definition of } \mathsf{time} \;\} \\
 & \mathsf{time}_2(\sigma, e) \quad .
\end{array}$$

□

10.23. THEOREM. $\Gamma_1 \trianglelefteq_t \Gamma_2 \;\Rightarrow\; T_T(\Gamma_1) \subseteq T_T(\Gamma_2)$.

PROOF. Straightforward from the fact that traces of $\mathcal{E}_1$ are also traces of $\mathcal{E}_2$ (cf. Theorem 10.6), and the fact that the enabling times of events in Γ_1 are unaffected in Γ_2 (cf. Lemma 10.22). □

The set of timed event traces of $\bigsqcup_i \Gamma_i$ can be characterized as the union of the sets of timed event traces of the event structures $\Gamma_1 \trianglelefteq_t \Gamma_2 \trianglelefteq_t \ldots$.

10.24. THEOREM. For $\Gamma_1 \trianglelefteq_t \Gamma_2 \trianglelefteq_t \ldots$ a chain: $T_T(\bigsqcup_i \Gamma_i) = \bigcup_i T_T(\Gamma_i)$.

PROOF. $\supseteq$': then we derive:

$$\begin{array}{cl}
 & \text{true} \\
\Leftrightarrow & \{\text{ Lemma 10.21 }\} \\
 & \forall\, i : \Gamma_i \trianglelefteq_t \bigsqcup_i \Gamma_i \\
\Rightarrow & \{\text{ Theorem 10.23 }\} \\
 & \forall\, i : T_T(\Gamma_i) \subseteq T_T(\bigsqcup_i \Gamma_i) \\
\Rightarrow & \{\text{ set calculus }\} \\
 & \bigcup_i T_T(\Gamma_i) \subseteq T_T(\bigsqcup_i \Gamma_i) \quad .
\end{array}$$

'$\subseteq$': let $\sigma \in T_T(\bigsqcup_i \Gamma_i)$ for $\bigsqcup_i \Gamma_i = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$. Let $\Gamma_k = \langle \mathcal{E}_k, \mathcal{D}_k, \mathcal{T}_k \rangle$ such that $\overline{[\sigma]} \subseteq E_k$. Since $E = \bigcup_i E_i$, Γ_k is a member of the chain. We prove that $\sigma \in T_T(\Gamma_k)$ by systematically checking the conditions of being a timed event trace. Let $\sigma = (e_1, t_1) \ldots (e_n, t_n)$.

1.
$$
\begin{array}{ll}
 & e_1 \dots e_n \in T(\mathcal{E}) \\
\Leftrightarrow & \{ \text{ Definition 4.5 } \} \\
 & \forall i : e_i \in \mathsf{en}([\sigma_i]) \\
\Leftrightarrow & \{ \ \Gamma_k \trianglelefteq_t \bigsqcup_i \Gamma_i;\ \overline{[\sigma]} \subseteq E_k;\ \text{Lemma 10.8 } \} \\
 & \forall i : e_i \in \mathsf{en}_k([\sigma_i]) \\
\Leftrightarrow & \{ \text{ Definition 4.5 } \} \\
 & e_1 \dots e_n \in T(\mathcal{E}_k) \quad .
\end{array}
$$

2.
$$
\begin{array}{ll}
 & \forall i : t_i \geqslant \mathsf{time}(\sigma_i, e_i) \\
\Leftrightarrow & \{ \ \Gamma_k \trianglelefteq_t \bigsqcup_i \Gamma_i;\ \overline{[\sigma]} \subseteq E_k;\ \text{Lemma 10.22 } \} \\
 & \forall i : t_i \geqslant \mathsf{time}_k(\sigma_i, e_i)
\end{array}
$$

Hence, each timed event trace σ in $\bigsqcup_i \Gamma_i$ with $\overline{[\sigma]} \subseteq E_k$ belongs to $T_T(\Gamma_k)$ which proves that $T_T(\bigsqcup_i \Gamma_i) \subseteq \bigcup_i T_T(\Gamma_i)$. □

A result that will be used in the next section is:

10.25. LEMMA. $\Gamma_1 \trianglelefteq_t \Gamma_2 \ \Rightarrow \ pos(\Gamma_2) \cap E_1 = pos(\Gamma_1)$.

PROOF. '$\subseteq$': by contradiction. Suppose $e \in pos(\Gamma_2) \cap E_1$ but $e \notin pos(\Gamma_1)$. Thus, $\mathcal{D}_1(e) = 0$. From $\Gamma_1 \trianglelefteq_t \Gamma_2$ it follows that $\mathcal{D}_2 \restriction E_1 = \mathcal{D}_1$. So, $\mathcal{D}_2(e) = \mathcal{D}_1(e) = 0$, contradicting $e \in pos(\Gamma_2)$.

'$\supseteq$': similar to the above case and omitted here. □

10.26. COROLLARY. $\Gamma_1 \trianglelefteq_t \Gamma_2 \ \Rightarrow \ pin(\Gamma_2) \cap E_1 = pin(\Gamma_1)$.

PROOF. Directly from Lemma 10.7 and 10.25, using that $pin(\Gamma) = init(\Gamma) \cup pos(\Gamma)$. □

10.3.2 A fixed point semantics

In this section we consider the timed event structure semantics of $P := B$ where $B \in \mathsf{PA}_T$. In order to adopt the approach of Section 10.2.2 the crucial issue is to prove that the operators $\overline{(t)\ a_\xi\,;}$, $\overline{+}$, ... are continuous in the timed setting[1].

10.27. LEMMA. For $\langle \mathsf{EBES}_T, \trianglelefteq_t \rangle$ and $F : \mathsf{EBES}_T \longrightarrow \mathsf{EBES}_T$ we have: F is continuous iff F is continuous on events.

PROOF. Similar to the untimed case (cf. Lemma 10.11). □

According to this lemma it suffices to prove continuity on events. That is, are the operators $\overline{\mathsf{op}}$ monotonic w.r.t. $\trianglelefteq_t$ (for instance, if $\Gamma_1 \trianglelefteq_t \Gamma_2$ do we have $\overline{(t)\ a_\xi\,;}\,\Gamma_1 \trianglelefteq_t \overline{(t)\ a_\xi\,;}\,\Gamma_2$) and do we have that the set of events of $\overline{\mathsf{op}}$ applied to the l.u.b. of chain $\Gamma_1 \trianglelefteq_t \Gamma_2 \trianglelefteq_t \dots$ is contained

[1] Strictly speaking we would need to distinguish between $\overline{+}$ for PA and $\overline{+}$ for PA_T. Throughout this chapter we will use the same notation for all cases for the sake of convenience.

in the set of events of the l.u.b. of chain $\overline{\mathsf{op}}(\Gamma_1) \trianglelefteq_t \overline{\mathsf{op}}(\Gamma_2) \trianglelefteq_t \ldots$? These issues will be considered in this section.

We start by extending the renaming operator on event structure $\mathcal{E}$, $\phi(\mathcal{E})$, to timed event structures (cf. Definition 10.9).

10.28. DEFINITION. For $\Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ and ϕ an occurrence identifier let
$\phi(\Gamma) \triangleq \langle \phi(\mathcal{E}), \mathcal{D}', \mathcal{T}' \rangle$ with $\mathcal{D}'(\phi e) = \mathcal{D}(e)$, and $\mathcal{T}'((\phi X, \phi e)) = \mathcal{T}((X, e))$. □

10.29. THEOREM. $\overline{(t)\ a_\xi\, ;}$, $\overline{+}$, ... and $\phi()$ are continuous on $\langle \mathsf{EBES}_T, \trianglelefteq_t \rangle$.

PROOF. We prove that the operators are continuous on events, which—by Lemma 10.27—proves the case. For the renaming operators $\phi()$ these proofs are trivial and omitted. We prove the theorem for $\overline{(t)\ a_\xi\, ;}$ and $\overline{||_G}$. The proofs for the other operators are similar and omitted here. In this proof let $\Gamma_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i \rangle$ with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1,2. Similarly Γ_i' is defined.

1. Action-prefix. Suppose $\Gamma_1 \trianglelefteq_t \Gamma_2$, and let $\Gamma_1' = \overline{(t)\ a_\xi\, ;}\Gamma_1$ and $\Gamma_2' = \overline{(t)\ a_\xi\, ;}\Gamma_2$. The proof obligation is $\Gamma_1 \trianglelefteq_t \Gamma_2 \;\Rightarrow\; \Gamma_1' \trianglelefteq_t \Gamma_2'$. This is proven by systematically checking the conditions of $\trianglelefteq_t$ (cf. Definition 10.16).

   (a) In order to prove $\mathcal{E}_1' \trianglelefteq \mathcal{E}_2'$ it suffices to concentrate on the bundle constraints; the sets of events, conflicts and labelling of events are identical to action-prefix for the untimed case, so for these components the constraints hold (cf. Theorem 10.12). For the bundle constraint we have—according to Definition 10.16—to check:

$$
\begin{aligned}
& \mapsto_1' \\
= \; & \{ \text{ definition } \mathcal{E}_T[\![\;]\!] \} \\
& \mapsto_1 \cup (\{\{\xi\}\} \times pin(\Gamma_1)) \\
= \; & \{ \text{ Corollary 10.26 } \} \\
& \mapsto_1 \cup (\{\{\xi\}\} \times (pin(\Gamma_2) \cap E_1)) \\
= \; & \{ \Gamma_1 \trianglelefteq_t \Gamma_2 \} \\
& \{ (X \cap E_1, e) \mid e \in E_1 \;\wedge\; X \mapsto_2 e \} \cup (\{\{\xi\}\} \times (pin(\Gamma_2) \cap E_1)) \\
= \; & \{ \text{ definition } \mathcal{E}_T[\![\;]\!];\ E_i' = E_i \cup \{\xi\} \text{ for } i{=}1,2 \} \\
& \{ (X \cap E_1', e) \mid e \in E_1' \;\wedge\; X \mapsto_2' e \} \quad .
\end{aligned}
$$

   (b) $\mathcal{D}_2' \restriction E_1' = \mathcal{D}_2' \restriction (\{\xi\} \cup E_1) = \{(\xi, t)\} \cup (E_1 \times \{0\}) = \mathcal{D}_1'$.

   (c) For $e \in E_1'$ we derive

$$
\begin{aligned}
& \mathcal{T}_1'((X_2' \cap E_1', e)) \\
= \; & \{ \text{ definition } \mathcal{E}_T[\![\;]\!] \} \\
& \begin{cases} \mathcal{T}_1((X_2' \cap E_1', e)) & \text{if } X_2' \cap E_1' \mapsto_1 e \\ \mathcal{D}_1(e) & \text{if } X_2' = \{\xi\} \end{cases} \\
= \; & \{ \Gamma_1 \trianglelefteq_t \Gamma_2;\ e \in E_1 \} \\
& \begin{cases} \mathcal{T}_1((X_2' \cap E_1, e)) & \text{if } X_2' \cap E_1 \mapsto_1 e \\ \mathcal{D}_2(e) & \text{if } X_2' = \{\xi\} \end{cases} \\
= \; & \{ \Gamma_1 \trianglelefteq_t \Gamma_2 \}
\end{aligned}
$$

$$\begin{cases} \mathcal{T}_2((X_2', e)) & \text{if } X_2' \mapsto_2 e \\ \mathcal{D}_2(e) & \text{if } X_2' = \{\,\xi\,\} \end{cases}$$
$= \quad \{$ definition $\mathcal{E}_T[\![\ ]\!]$ $\}$
$\mathcal{T}_2'((X_2', e))$.

This proves that $\overline{(t)\ a_\xi\ ;}$ is monotonic. It remains to prove:

$E(\overline{(t)\ a_\xi\ ;}\ \bigsqcup_i \Gamma_i)$
$= \quad \{$ definition $\mathcal{E}_T[\![\ ]\!]$ $\}$
$\{\,\xi\,\} \cup E(\bigsqcup_i \Gamma_i)$
$= \quad \{$ Definition 10.20 $\}$
$\{\,\xi\,\} \cup \bigcup_i E(\Gamma_i)$
$= \quad \{$ set calculus $\}$
$\bigcup_i (\{\,\xi\,\} \cup E(\Gamma_i))$
$= \quad \{$ definition $\mathcal{E}_T[\![\ ]\!]$ $\}$
$\bigcup_i E(\overline{(t)\ a_\xi\ ;}\Gamma_i)$
$= \quad \{$ Definition 10.20 $\}$
$E(\bigsqcup_i \overline{(t)\ a_\xi\ ;}\Gamma_i)$.

2. Parallel composition. Suppose $\Gamma_1 \trianglelefteq_t \Gamma_2$, and let $\Gamma_1' = \Gamma_1 \overline{||_G} \Gamma$ and $\Gamma_2' = \Gamma_2 \overline{||_G} \Gamma$ where $\Gamma = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ with $\mathcal{E} = (E, \rightsquigarrow, \mapsto, l)$. We prove $\Gamma_1' \trianglelefteq_t \Gamma_2'$ by checking the conditions of $\trianglelefteq_t$.
   (a) $\mathcal{E}_1' \trianglelefteq \mathcal{E}_2'$ follows directly from the untimed case (cf. Theorem 10.12) and the fact that $\mathcal{E}_T[\![\ ]\!]$ is a conservative extension of $\mathcal{E}'[\![\ ]\!]$.
   (b) $\mathcal{D}_1' = \mathcal{D}_2' \restriction E_1'$. Recall that events are pairs (e_1, e_2) where possibly one of the two events equals '$*$'. We consider the following cases
      i. (e_1, e_2) is a synchronization event, so $e_1 \in E_1^s$ and $e_2 \in E^s$.

      $\mathcal{D}_1'((e_1, e_2))$
      $= \quad \{$ definition $\mathcal{E}_T[\![\ ]\!]$ $\}$
      $max(\mathcal{D}_1(e_1), \mathcal{D}(e_2))$
      $= \quad \{\ \Gamma_1 \trianglelefteq_t \Gamma_2 \ \Rightarrow\ \mathcal{D}_1(e_1) = \mathcal{D}_2(e_1)\ \}$
      $max(\mathcal{D}_2(e_1), \mathcal{D}(e_2))$
      $= \quad \{\ \mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \ \Rightarrow\ E_1^s \subseteq E_2^s\ ;\ e_1 \in E_1^s$; definition $\mathcal{E}_T[\![\ ]\!]$ $\}$
      $\mathcal{D}_2'((e_1, e_2))$.

      ii. (e_1, e_2) is a non-synchronizing event, say $e_1 \in E_1^f$ and $e_2 = *$. Then $\mathcal{D}_2'((e_1, *)) = \mathcal{D}_2(e_1) = \mathcal{D}_1(e_1) = \mathcal{D}_1'((e_1, *))$.
      iii. for $(*, e_2)$ with $e_2 \neq *$ the proof is similar and omitted.

   This proves $\mathcal{D}_1' = \mathcal{D}_2' \restriction E_1'$.
   (c) Let $e = (e_1, e_2)$ an event in E_1'. Then we derive:

   $\mathcal{T}_2'((X_2', (e_1, e_2)))$
   $= \quad \{$ definition $\mathcal{E}_T[\![\ ]\!]$ $\}$

$$\begin{aligned}
& \mathit{max}(\mathcal{T}_2((\mathsf{pr}_1(X_2'), e_1)), \mathcal{T}((\mathsf{pr}_2(X_2'), e_2))) \\
= \quad & \{\ \Gamma_1 \trianglelefteq_t \Gamma_2\ \} \\
& \mathit{max}(\mathcal{T}_1((\mathsf{pr}_1(X_2') \cap E_1, e_1)), \mathcal{T}((\mathsf{pr}_2(X_2'), e_2))) \\
= \quad & \{\ \text{calculus}\ \} \\
& \mathit{max}(\mathcal{T}_1((\mathsf{pr}_1(X_2' \cap E_1'), e_1)), \mathcal{T}((\mathsf{pr}_2(X_2' \cap E_1'), e_2))) \\
= \quad & \{\ \text{definition } \mathcal{E}_T[\![\ ]\!]\ \} \\
& \mathcal{T}_1'(((X_2' \cap E_1'), (e_1, e_2))) \quad .
\end{aligned}$$

This proves that $\overline{||_G}$ is monotonic in the left argument. By symmetry, the proof for monotonicity in the right argument is obtained by reversing the arguments in the above proof. The fact that $\overline{||_G}$ is continuous on events follows from the fact that in the untimed case this holds and the fact that the construction of the set of events in the timed case is identical to the untimed case.

□

In the following definition let $\mathcal{G}_B$ be the timed counterpart of $\mathcal{F}_B$. $\mathcal{G}_B$ is a function determined by $\overline{\mathsf{op}}$ and $\phi()$. From the previous theorem it follows that $\mathcal{G}_B$ is continuous on timed event structures ordered under $\trianglelefteq_t$. This means that the semantics of $P := B$ for $B \in \mathsf{PA}_T$ can now be computed as the l.u.b. of sequence $\bot_t, \mathcal{G}_B(\bot_t), \mathcal{G}_B(\mathcal{G}_B(\bot_t)), \ldots$.

10.30. DEFINITION. For $P := B$ a process definition let $\mathcal{E}_T[\![\, P \,]\!] \triangleq \bigsqcup_i \mathcal{G}_B^i(\bot_t)$. □

10.31. EXAMPLE. As an example of a recursive process definition in PA_T we consider

$$P := (3)\ a\,;\ ((14)\ b\,;\ P + (1)\ c\,;\ (\pi)\ d\,;\ P) \quad .$$

The first approximation of the timed event structure semantics of this definition is $\bot_t$, the empty structure. The second approximation $\mathcal{G}_B(\bot_t)$ is depicted in Figure 10.4(a). By repeated substitution we obtain the timed event structure depicted in Figure 10.4(b). □

For $P := B$ let $\Phi_T(P)$ the corresponding untimed behaviour of P. For instance, $\Phi_T(P)$ for the process of the above example equals $a\,;\,(b\,;\,P + c\,;\,d\,;\,P)$. The next theorem extends the compatibility result of Chapter 4 (Theorem 4.36). We first introduce

10.32. LEMMA. For all $i \geqslant 0$ and $\mathcal{G}_B^i(\bot_t) = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i \rangle$ we have $L(\mathcal{E}_i) = L(\mathcal{F}^i_{\Phi_T(B)}(\bot))$.

PROOF. Straightforward by induction on i, using the fact that $\overline{(t)\ a_\xi\,;}, \overline{+}, \overline{||_G}, \ldots$ preserve lposet equivalence (cf. Theorem 4.36). □

10.33. THEOREM. For $\mathcal{E}_T[\![\, P \,]\!] = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ we have $L(\mathcal{E}) = L(\mathcal{E}[\![\, \Phi_T(P) \,]\!])$.

PROOF. Let $P := B$, $\mathcal{E}_T[\![\, P \,]\!] = \bigsqcup_i \mathcal{G}_B^i(\bot_t) = \langle \mathcal{E}, \mathcal{D}, \mathcal{T} \rangle$ where $\mathcal{G}_B^i(\bot_t) = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i \rangle$. Then:

$$\begin{aligned}
& \text{true} \\
\Leftrightarrow \quad & \{\ \text{Lemma 10.32}\ \}
\end{aligned}$$

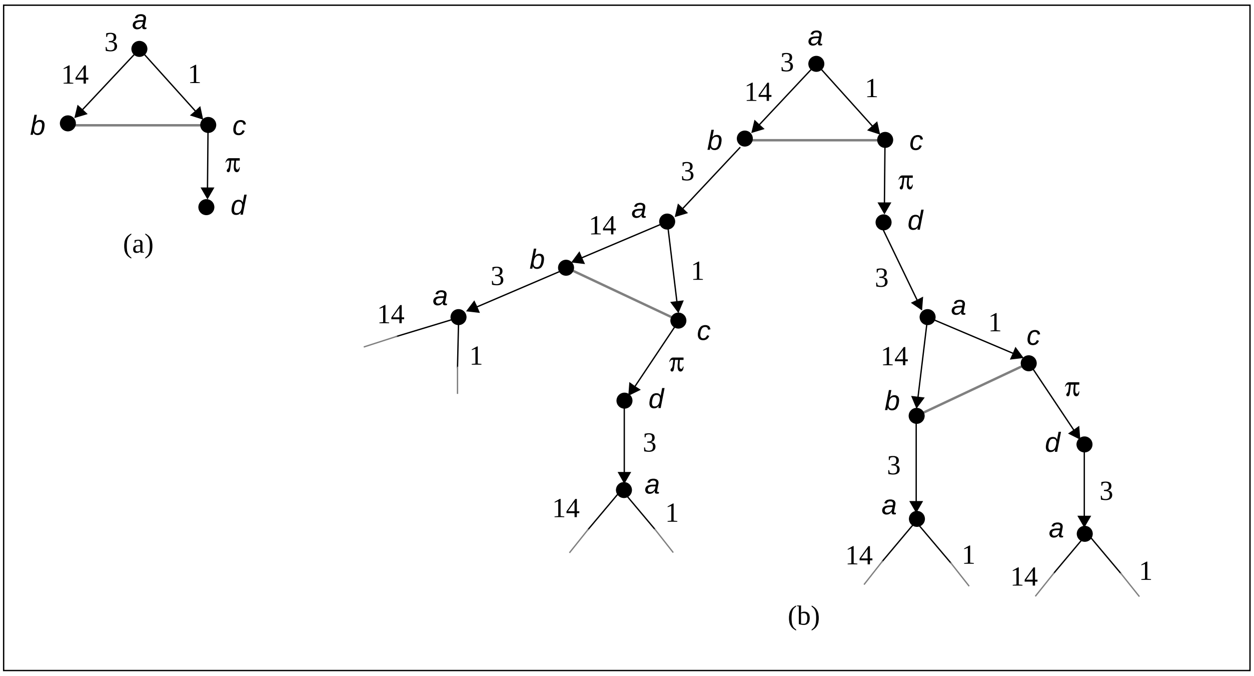


Figure 10.4: Example of semantics for a recursive process definition in PA_T.

$$
\begin{array}{ll}
& \forall\, i : L(\mathcal{E}_i) = L(\mathcal{F}^i_{\Phi_T(B)}(\bot)) \\
\Leftrightarrow & \{\ L(\mathcal{E}) = L(\mathcal{E}') \Leftrightarrow T(\mathcal{E}) = T(\mathcal{E}')\ \} \\
& \forall\, i : T(\mathcal{E}_i) = T(\mathcal{F}^i_{\Phi_T(B)}(\bot)) \\
\Rightarrow & \{\ \text{set calculus}\ \} \\
& \bigcup_i T(\mathcal{E}_i) = \bigcup_i T(\mathcal{F}^i_{\Phi_T(B)}(\bot)) \\
\Leftrightarrow & \{\ \text{Theorem 10.24}\ \} \\
& T(\bigsqcup_i \mathcal{E}_i) = T(\bigsqcup_i \mathcal{F}^i_{\Phi_T(B)}(\bot)) \\
\Leftrightarrow & \{\ L(\mathcal{E}) = L(\mathcal{E}') \Leftrightarrow T(\mathcal{E}) = T(\mathcal{E}')\ \} \\
& L(\bigsqcup_i \mathcal{E}_i) = L(\bigsqcup_i \mathcal{F}^i_{\Phi_T(B)}(\bot)) \\
\Leftrightarrow & \{\ \text{Definition 10.20; Definition 10.13}\ \} \\
& L(\mathcal{E}) = L(\mathcal{E}[\![\, \Phi_T(B)\,]\!]) \quad .
\end{array}
$$
□

We conclude this section by discussing the notion of *finite variability*. According to Nicollin & Sifakis [112] a behaviour possesses the so-called finite variability property iff it cannot perform infinitely many events in a finite amount of time. Such behaviours are also known as *non-Zeno* behaviours. Several timed process algebras explicitly abandon Zeno behaviours—behaviours that may execute an infinite amount of events in finite time. For instance, in a former proposal for timed CSP by Reed & Roscoe [124] a small delay is associated to each action such that Zeno-processes cannot be expressed. In our case we permit Zeno behaviours, for instance, $P := (0)\ a\,;\ P$ is a behaviour that may perform infinitely many a actions in finite time. In the same way as we are able to construct specifications in which deadlocks and/or livelocks can occur we consider it sufficient to be able to verify that a specification has such (possibly undesired) behaviour. Example algorithms to detect whether a recursive process definition allows Zeno behaviours can be found in the thesis of Hansson [64].

10.3.3 Event-based operational semantics

This section extends the event-based operational semantics of PA_T with recursion. We follow the approach of Langerak [89, Section 8.4]

It is assumed that each process instantiation of P is uniquely identified, as well as all occurrences of action-prefix and $\surd$. Different occurrences of the same process instantiation should produce different event transitions. In addition, event transitions cannot be repeated. For $P := (2)\ a_\xi\ ;\ P_\phi$ we first have an event transition with (ξ, a, t) for $t \geqslant 2$; the next time that action a occurs it should be labelled with a label different from ξ. These complications are resolved by using an event renaming operator that prefixes all events in a behaviour with a certain occurrence identifier. $\phi(B)$ is behaviour B where all event identifiers in B are prefixed with ϕ. For these renamed behaviours we have the simple rule that whenever $B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'$ then $\phi(B)$ can perform $(\phi\xi, a, t)$ evolving into $\phi(B')$. The inference rules for process instantiation are presented in Table 10.1.

10.34. Example. For example, for $P := B$ with $B = (4)\ a_\xi\ ;\ P_\phi + (1)\ b_\chi\ ;\ P_\psi$ we have the following derivation:

$$P \xrightarrow{(\xi,a,7)}\!\!\!\!\rightarrow \varepsilon({}^7[\,P_\phi\,]) \xrightarrow{(\phi\chi,b,8)}\!\!\!\!\rightarrow {}^7[\,\phi({}^1[\,P_\psi\,])\,] \xrightarrow{(\phi\psi\xi,a,12)}\!\!\!\!\rightarrow {}^7[\,\phi({}^1[\,\psi({}^4[\,P_\phi\,])\,])\,]$$

where ε is the empty prefix. The third transition is derived as follows:

$$\begin{array}{ll}
 & B \xrightarrow{(\xi,a,4)}\!\!\!\!\rightarrow {}^4[\,P_\phi\,] \\
\Rightarrow & \{\ \text{SOS-rule for } P_\phi\ \} \\
 & P_\psi \xrightarrow{(\psi\xi,a,4)}\!\!\!\!\rightarrow \psi({}^4[\,P_\phi\,]) \\
\Rightarrow & \{\ \text{SOS-rule for } {}^t[\,B\,]\ \} \\
 & {}^1[\,P_\psi\,] \xrightarrow{(\psi\xi,a,5)}\!\!\!\!\rightarrow {}^1[\,\psi({}^4[\,P_\phi\,])\,] \\
\Rightarrow & \{\ \text{SOS-rule for } \phi(B)\ \} \\
 & \phi({}^1[\,P_\psi\,]) \xrightarrow{(\phi\psi\xi,a,5)}\!\!\!\!\rightarrow \phi({}^1[\,\psi({}^4[\,P_\phi\,])\,]) \\
\Rightarrow & \{\ \text{SOS-rule for } {}^t[\,B\,]\ \} \\
 & {}^7[\,\phi({}^1[\,P_\psi\,])\,] \xrightarrow{(\phi\psi\xi,a,12)}\!\!\!\!\rightarrow {}^7[\,\phi({}^1[\,\psi({}^4[\,P_\phi\,])\,])\,]\quad .
\end{array}$$

□

$$\frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{P_\phi \xrightarrow{(\phi\xi,a,t)}\!\!\!\!\rightarrow \phi(B')}\ (P := B) \qquad \frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{\phi(B) \xrightarrow{(\phi\xi,a,t)}\!\!\!\!\rightarrow \phi(B')}$$

Table 10.1: Additional transition rules for PA_T.

The following theorem extends Theorem 5.10:

10.35. THEOREM. For $P := B$ we have $\Phi(\mathsf{TS}_T(P)) = \mathsf{TS}(\Phi_T(P))$.

PROOF. If we delete all event name information and timing information from the rules in Table 10.1 we obtain the following rules

$$\frac{B \xrightarrow{a} B'}{P \xrightarrow{a} B'} \quad (P := B) \qquad \frac{B \xrightarrow{a} B'}{B \xrightarrow{a} B'}$$

The left-hand rule is the standard derivation rule for process definition and the second is a tautology. □

Like for the nonrecursive case (cf. Chapter 5) the resulting timed event transition system is *deterministic*. This implies that the operational semantics of a behaviour can also be given by its set of timed event traces. In the remainder of this section we would like to prove that the operational semantics coincides with the causality-based semantics given in the previous section, in the sense that both semantic models generate identical sets of timed event traces. In this study we could consider traces of infinite length (ω-traces) but this would not enhance expressivity. We can safely restrict ourselves to finite traces, since two transition systems having the same set of finite traces also have the same set of infinite traces in case the transition systems are deterministic.

In order to obtain the set of timed event traces of a process definition $P := B$ the idea is to define a function $\mathcal{G}'_B$ that substitutes a set of timed event traces for each occurrence of P in B, interpreting all operators in B as operators on timed traces. (Notice the similarity with $\mathcal{G}_B$.) We follow a similar procedure as in Section 10.2.2 and start by defining a renaming operator on sets of timed traces.

10.36. DEFINITION. For T a set of timed event traces and ϕ an occurrence identifier let $\phi(T) \triangleq \{\, \phi(\sigma) \mid \sigma \in T \,\}$ where $\phi(\varepsilon) \triangleq \varepsilon$ and $\phi((e,t)\ \sigma) \triangleq (\phi e, t)\ \phi(\sigma)$. □

As a second step towards the definition of $\mathcal{G}'_B$ all operators in B (like ;, +, >>, ...) must be interpreted as operators on timed event traces. In Chapter 5 we have defined a timed event trace semantics of PA_T. Since this definition is compositional we have in fact implicitly defined operators on timed traces. For example, $\mathcal{T}_T[\![\, B_1 + B_2 \,]\!] = \mathcal{T}_T[\![\, B_1 \,]\!] \,\overline{+}'\, \mathcal{T}_T[\![\, B_2 \,]\!]$ where $\overline{+}'$ denotes the choice-operation on timed traces (rather than on expressions). In the sequel we denote for operator $\mathsf{op} \in \mathsf{PA}_T$ the corresponding counterpart on timed traces by $\overline{\mathsf{op}}'$. $\mathcal{G}'_B$ for $P := B$ replaces all occurrences P_ϕ in B by $\phi(T)$ and interprets all operators op in B as operators $\overline{\mathsf{op}}'$ on (the substituted) timed traces.

10.37. DEFINITION. The depth of an event identifier is defined as follows:

1. $\mathsf{dp}(\xi) = 1$
2. $\mathsf{dp}(\xi e) = \mathsf{dp}(e) + 1$
3. $\mathsf{dp}((e_1, e_2)) = max(\mathsf{dp}(e_1), \mathsf{dp}(e_2))$.

□

10.38. Definition. A timed event trace σ is an i-trace ($i > 0$) iff the depth of each event in σ is at most i, that is, $\forall\, e_i \in \overline{[\sigma]} : \mathsf{dp}(e_i) \leqslant i$. $\square$

10.39. Lemma. For $P := B$ the set of i-traces of P equals $\mathcal{G'}^{i}_{B}(\varnothing)$.

Proof. By induction on i. Similar to the untimed case [89, Lemma 8.4.6] and omitted here. $\square$

The set of timed event traces of P is equal to the union of the sets of i-traces of P for all i. That is, $\mathcal{T}_T[\![\, P \,]\!] \triangleq \mathcal{G'}^{i}_{B}(\varnothing)$. The following theorem extends the compatibility result of Chapter 5 towards recursive process definitions.

10.40. Theorem. For $P := B$ we have $T_T(\mathcal{E}_T[\![\, P \,]\!]) = \mathcal{T}_T[\![\, P \,]\!]$.

Proof.

$$
\begin{array}{ll}
 & \text{true} \\
\Leftrightarrow & \{ \text{ Theorem 5.18 } \} \\
 & \forall\, i : T_T(\mathcal{G}^i_B(\bot_t)) = \mathcal{G'}^{i}_{B}(\varnothing) \\
\Rightarrow & \{ \text{ set calculus } \} \\
 & \bigcup_i T_T(\mathcal{G}^i_B(\bot_t)) = \bigcup_i \mathcal{G'}^{i}_{B}(\varnothing) \\
\Leftrightarrow & \{ \text{ Theorem 10.24 } \} \\
 & T_T(\bigsqcup_i \mathcal{G}^i_B(\bot_t)) = \bigcup_i \mathcal{G'}^{i}_{B}(\varnothing) \\
\Leftrightarrow & \{ \text{ Definition 10.30; see above } \} \\
 & T_T(\mathcal{E}_T[\![\, P \,]\!]) = \mathcal{T}_T[\![\, P \,]\!] \quad .
\end{array}
$$

$\square$

Similar as for the finite case this result can be strengthened towards strong bisimulation equivalence of the transition system deduced from the operational semantics and the transition system obtained from the denotational semantics by considering timed remainders after traces of length 1.

10.4 Urgent event structures

This section treats the extension of PA_U with recursion. It basically deals with the extension of the material of Section 10.3 with the notion of urgency. Section 10.4.1 introduces the pointed c.p.o. $\trianglelefteq_u$, characterizes the l.u.b. of a chain of urgent event structures ordered by $\trianglelefteq_u$, and considers some properties of this ordering. $\trianglelefteq$ and $\trianglelefteq_t$ were shown before to preserve trace sets. That is, $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \Rightarrow T(\mathcal{E}_1) \subseteq T(\mathcal{E}_2)$, and similarly for the timed case. It will be shown that due to the presence of urgent events this property does not hold in general for urgent event structures. Conditions will be provided under which trace set inclusion is still preserved, and a somewhat weaker notion of trace set inclusion will be considered. This is presented in Section 10.4.1. The denotational and operational semantics of $P := B$ with $B \in \mathsf{PA}_U$ is provided in Sections 10.4.2 and 10.4.3, respectively. The consistency proof of these two semantics is also given in Section 10.4.3.

10.4.1 A pointed complete partial order

The definitions and theorems in this section are all relative to urgent event structures $\Psi_i = \langle \Gamma_i, \mathcal{U}_i \rangle$ with $\Gamma_i = \langle (E_i, \rightsquigarrow_i, \mapsto_i, l_i), \mathcal{D}_i, \mathcal{T}_i \rangle$ for i=1, 2.

10.41. DEFINITION. (*Partial order on urgent event structures*)

$\Psi_1 \trianglelefteq_u \Psi_2$ iff $\Gamma_1 \trianglelefteq_t \Gamma_2$ and $\mathcal{U}_1 = \mathcal{U}_2 \upharpoonright E_1$. □

In addition to the constraints for $\trianglelefteq_t$ (cf. Definition 10.16) we require that the urgency predicate for events that are already in Ψ_1 is unaffected.

10.42. LEMMA. $\langle \mathsf{EBES}_U, \trianglelefteq_u \rangle$ is a pointed c.p.o..

PROOF. Routine and omitted. □

It is easy to verify that $\bot_u = \langle \bot_t, \varnothing \rangle$ is the least element under $\trianglelefteq_u$.

For chain $\Psi_1 \trianglelefteq \Psi_2 \trianglelefteq \ldots$ we define the following urgent event structure.

10.43. DEFINITION. (*Least upper bound (under $\trianglelefteq_u$)*)

Let $\Psi_1 \trianglelefteq \Psi_2 \trianglelefteq \ldots$ be a chain, then $\bigsqcup_i \Psi_i \triangleq \langle \bigsqcup_i \Gamma_i, \bigcup_i \mathcal{U}_i \rangle$. □

10.44. LEMMA. $\bigsqcup_i \Psi_i$ is the least upper bound of chain $\Psi_1 \trianglelefteq_u \Psi_2 \trianglelefteq_u \ldots$.

PROOF. Similar to the proof of Lemma 10.21. □

Two urgent event structures that are ordered under $\trianglelefteq_u$ and that have identical sets of events are identical.

10.45. THEOREM. $(\Psi_1 \trianglelefteq_u \Psi_2 \ \wedge\ E_1 = E_2) \ \Rightarrow\ \Psi_1 = \Psi_2$.

PROOF. From Theorem 10.23 and $\mathcal{U}_1 = \mathcal{U}_2 \upharpoonright E_1 = \mathcal{U}_2 \upharpoonright E_2 = \mathcal{U}_2$. □

For the timed case we had the nice property that a timed trace of Γ_1 is also a timed trace of Γ_2 if Γ_1 is smaller than Γ_2 in the ordering ($\trianglelefteq_t$). This property conforms to the intuition that possible executions of an approximation Γ_{i+1} are consistent *extensions* of possible runs of Γ_i. As we will show below a similar property for urgent timed event structures does not hold in general, since new urgent events in Ψ_{i+1} may restrict (or, even prevent) the occurrence of events in Ψ_i.

A timed event trace σ of Ψ_i disappears in approximation Ψ_{i+1} if there is an urgent event that could occur earlier than some event in σ. Stated otherwise, the only reason that a trace disappears in a next approximation is by violating the third constraint of being a timed event trace (cf. Definition 6.3).

10.46. LEMMA. For $\Psi_1 \trianglelefteq_u \Psi_2$ we have:

$$\sigma \in T_U(\Psi_1) \setminus T_U(\Psi_2) \ \Rightarrow$$
$$(\exists\, e \in E_2, e_i \in \overline{[\sigma]} : \mathcal{U}_2(e) \ \wedge\ e \in \mathsf{en}_2([\sigma_i]) \ \wedge\ \mathsf{time}_2(\sigma_i, e_i) > \mathsf{time}_2(\sigma_i, e)).$$

PROOF. Assume $\Psi_1 \trianglelefteq_u \Psi_2$ and let $\sigma \in T_U(\Psi_1)$ and $\sigma \notin T_U(\Psi_2)$. We systematically check the conditions of $\sigma \notin T_U(\Psi_2)$.

1. $[\sigma] \notin T(\mathcal{E}_2)$. But $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ and $[\sigma] \in T(\mathcal{E}_1)$ implies that $[\sigma] \in T(\mathcal{E}_2)$. Contradiction.
2. there exists i such that $\neg\mathcal{U}_2(e_i)$ and $t_i < \mathsf{time}_2(\sigma_i, e_i)$ or $\mathcal{U}_2(e_i)$ and $t_i \neq \mathsf{time}_2(\sigma_i, e_i)$. But, since $\overline{[\sigma]} \subseteq E_1$ we have that $\mathcal{U}_1(e_i) = \mathcal{U}_2(e_i)$ and $\mathsf{time}_1(\sigma_i, e_i) = \mathsf{time}_2(\sigma_i, e_i)$, and since $\sigma \in T_U(\Psi_1)$ it follows that the timing of e_i is correct. Contradiction.
3. σ is not time-consistent. Contradiction with $\sigma \in T_U(\Psi_1)$.

So, σ satisfies three of the four conditions of being a timed event trace of Ψ_2, and $\sigma \notin T_U(\Psi_2)$ can only be caused by violation of the third constraint. $\square$

As a next step we investigate under which conditions trace sets are preserved and under which conditions a somewhat weaker notion of trace inclusion (but still a rather intuitive notion) is preserved. It is intuitively not hard to see that trace inclusion is preserved when the set of urgent events does not 'grow' in subsequent approximations. This is shown in the following lemma. Let $U(\Psi)$ be the set of urgent events of Ψ, i.e., $U(\Psi) \triangleq \{\, e \in E \mid \mathcal{U}(e) = \text{true}\,\}$.

10.47. LEMMA. $(\Psi_1 \trianglelefteq_u \Psi_2 \;\wedge\; U(\Psi_1) = U(\Psi_2)) \;\Rightarrow\; T_U(\Psi_1) \subseteq T_U(\Psi_2)$.

PROOF. Let $\sigma = (e_1, t_1)\ldots(e_n, t_n)$ in $T_U(\Psi_1)$. We prove that $\sigma \in T_U(\Psi_2)$ by systematically checking the conditions of being a timed event trace of Ψ_2.

1. $$e_1 \ldots e_n \in T(\mathcal{E}_1)\{\text{ Theorem 10.6 }\}e_1 \ldots e_n \in T(\mathcal{E}_2) \quad .$$

2. $$\begin{array}{ll} & \forall i : (\mathcal{U}_1(e_i) \;\Rightarrow\; t_i = \mathsf{time}_1(\sigma_i, e_i)) \;\wedge\; (\neg\mathcal{U}_1(e_i) \;\Rightarrow\; t_i \geqslant \mathsf{time}_1(\sigma_i, e_i)) \\ \Rightarrow & \{\; U(\Psi_1) = U(\Psi_2) \text{ and } \mathcal{U}_2 \restriction E_1 = \mathcal{U}_1 \;\} \\ & (\forall i : \mathcal{U}_2(e_i) \;\Rightarrow\; t_i = \mathsf{time}_1(\sigma_i, e_i)) \;\wedge\; (\neg\mathcal{U}_2(e_i) \;\Rightarrow\; t_i \geqslant \mathsf{time}_1(\sigma_i, e_i)) \\ \Leftrightarrow & \{\; \overline{[\sigma]} \subseteq E_1 \;\Rightarrow\; \overline{[\sigma_i]} \subseteq E_1, \text{ for all } i\text{; Lemma 10.22 } \} \\ & \forall i : (\mathcal{U}_2(e_i) \;\Rightarrow\; t_i = \mathsf{time}_2(\sigma_i, e_i)) \;\wedge\; (\neg\mathcal{U}_2(e_i) \;\Rightarrow\; t_i \geqslant \mathsf{time}_2(\sigma_i, e_i)) \quad . \end{array}$$

3. σ is time-consistent since $\sigma \in T_U(\Psi_1)$.

4. $$\begin{array}{ll} & \forall i, e \in E_1 : e \in \mathsf{en}_1([\sigma_i]) \;\wedge\; \mathcal{U}_1(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}_1(\sigma_i, e) \\ \Leftrightarrow & \{\; U(\Psi_1) = U(\Psi_2) \;\} \\ & \forall i, e \in E_2 : e \in \mathsf{en}_1([\sigma_i]) \;\wedge\; \mathcal{U}_2(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}_1(\sigma_i, e) \\ \Leftrightarrow & \{\; \overline{[\sigma]} \subseteq E_1 \;\Rightarrow\; \overline{[\sigma_i]} \subseteq E_1, \text{ for all } i\text{; Lemma 10.22 } \} \\ & \forall i, e \in E_2 : e \in \mathsf{en}_1([\sigma_i]) \;\wedge\; \mathcal{U}_2(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}_2(\sigma_i, e) \\ \Leftrightarrow & \{\text{ Lemma 10.8 }\} \\ & \forall i, e \in E_2 : e \in \mathsf{en}_2([\sigma_i]) \;\wedge\; \mathcal{U}_2(e) \;\Rightarrow\; t_i \leqslant \mathsf{time}_2(\sigma_i, e) \quad . \end{array}$$

$\square$

10.48. COROLLARY. For chain $\Psi_1 \trianglelefteq_u \Psi_2 \trianglelefteq_u \ldots$ with $U(\Psi_{i+1}) = U(\Psi_i)$, for $i > 0$:

$$T_U(\bigsqcup_i \Psi_i) = \bigcup_i T_U(\Psi_i) \quad .$$

PROOF. Straightforward from the previous lemma and Theorem 10.45. □

$\trianglelefteq_u$ corresponds to a weaker notion of trace set inclusion in case the introduction of new urgent events is allowed, but only in such a way that the introduction of conflicts $e \rightsquigarrow e'$, where e' is a new urgent event and e an already existing one, is prohibited. In this case new urgent events will not restrict the occurrence of already existing events, but the 'old' events may be preceded by the new urgent events. For example, in

$$\overset{7}{\circ}\; b \quad \trianglelefteq_u \quad \overset{7}{\circ}\; b \qquad \overset{2}{\circ}\; a$$

$(e_b, 7)$ is not a timed trace of the 'larger' structure, but $(e_a, 2)(e_b, 7)$ is—event e_b is preceded by a new urgent event, but is not excluded.

As a subsidiary notion we define an ordering relation on sets of timed traces, called *weak trace set inclusion*. This ordering relation is based on restriction of timed traces on sets of events.

10.49. DEFINITION. For timed event trace σ and set of events E, $\sigma \restriction E$ is defined by

1. $\varepsilon \restriction E \triangleq \varepsilon$
2. $((e, a, t)\,\sigma) \restriction E \triangleq \begin{cases} (e, a, t)\,(\sigma \restriction E) & \text{if } e \in E \\ \sigma \restriction E & \text{if } e \notin E. \end{cases}$

□

10.50. DEFINITION. For T_1, T_2 sets of timed event traces let

$$T_1 \sqsubseteq T_2 \iff (\forall\, \sigma_1 \in T_1 : (\exists\, \sigma_2 \in T_2 : \sigma_2 \restriction \overline{[\sigma_1]} = \sigma_1)) \quad .$$

□

We now have the following result concerning weak trace set inclusion:

10.51. THEOREM. *Weak trace set inclusion theorem*

$$\Psi_1 \trianglelefteq_u \Psi_2 \;\wedge\; (\forall\, e \in E_1, e' \in E_2 \setminus E_1 : e \rightsquigarrow_2 e' \;\Rightarrow\; \neg\,\mathcal{U}_2(e')) \;\Rightarrow\; T_U(\Psi_1) \sqsubseteq T_U(\Psi_2).$$

PROOF. Assume $\Psi_1 \trianglelefteq_u \Psi_2$ and let $\sigma = (e_1, t_1) \ldots (e_n, t_n)$ with $\sigma \in T_U(\Psi_1)$. The proof is as follows. We first provide a recipe to generate from σ a sequence σ' of the following form $\sigma' = \sigma'^1\ (e_1, t_1)\ \sigma'^2\ (e_2, t_2) \ldots \sigma'^n\ (e_n, t_n)$ and subsequently prove that $\sigma' \in T_U(\Psi_2)$.

The algorithm to compute subsequences σ'^i is as follows:

for $0 < i \leqslant n$
do $\sigma'^i := \varepsilon$;
  $\sigma'' := \sigma'^1\ (e_1, t_1)\ \sigma'^2 \ldots \sigma'^{i-1}\ (e_{i-1}, t_{i-1})\ \sigma'^i$;
  $S_i := \{\, (e, t) \mid e \in \mathsf{en}_2([\sigma'']) \;\wedge\; \mathcal{U}_2(e) \;\wedge\; t_i > \mathsf{time}(\sigma'', e) = t \,\}$;

```
    while $S_i \neq \varnothing$
    do choose $(e,t) \in S_i$ such that $\forall\,(e',t') \in S_i : t \leqslant t'$
        $\sigma'^i := \sigma'^i\ (e,t)$;
        $\sigma'' := \sigma''\ (e,t)$;
        $S_i := \{\,(e,t) \mid e \in \mathsf{en}_2([\sigma'']) \ \wedge\ \mathcal{U}_2(e) \ \wedge\ t_i > \mathsf{time}(\sigma'',e) = t\,\}$;
    od
od.
```

Obviously, this algorithm should terminate since σ is finite and Ψ_2 contains a finite set of events. We prove that σ' is a timed event trace of Ψ_2 by checking the conditions of being a timed event trace:

1. the proof that $[\sigma'] \in T(\mathcal{E}_2)$ is by contradiction. Suppose $[\sigma'] \notin T(\mathcal{E}_2)$. Then this could only be because one of the following reasons:

   (a) $\exists\, e_i, e_j : e_i \rightsquigarrow_2 e_j$ and $j \leqslant i$. Consider the following cases:

      i. $e_i, e_j \in \overline{[\sigma]}$. But then $e_i \rightsquigarrow_1 e_j$ and $[\sigma]$ would not be an event trace of $T(\mathcal{E}_1)$. Contradiction.

      ii. $e_i, e_j \notin \overline{[\sigma]}$. But then $e_i \rightsquigarrow_2 e_j$ which is impossible by construction of σ'.

      iii. $e_i \in \overline{[\sigma]}, e_j \notin \overline{[\sigma]}$. But then $\mathcal{U}_2(e_j)$ and $e_i \in E_1$ and $e_i \rightsquigarrow_2 e_j$. Contradiction.

      iv. $e_i \notin \overline{[\sigma]}, e_j \in \overline{[\sigma]}$. But then $e_i \rightsquigarrow_2 e_j$ which is impossible by construction of σ'.

   (b) $\exists\, X_2 \subseteq E_2 : X_2 \mapsto_2 e_i \ \wedge\ X_2 \cap \overline{[\sigma'_i]} = \varnothing$. Consider

      i. $e_i \notin \overline{[\sigma]}$. But then e_i would not be enabled in σ' which is impossible by construction of σ'.

      ii. $e_i \in \overline{[\sigma]}$. But then $e_i \in E_1$ and $X_1 \mapsto_1 e_i$ such that $X_2 \cap E_1 = X_1$. Since $\sigma \in T_U(\Psi_1)$ we have that there exists e_j $(j < i)$ in σ such that $e_j \in X_1$. By construction it follows that e_j in σ'. Contradiction.

   This proves that $[\sigma'] \in T(\mathcal{E}_2)$.

2. since $\overline{[\sigma]} \subseteq E_1$ and $\Psi_1 \trianglelefteq_u \Psi_2$ it follows from Lemma 10.22 that $\mathsf{time}_2(\sigma,e) = \mathsf{time}_1(\sigma,e)$ for $e \in E_1$. Since the new urgent events in Ψ_2 are not in conflict with any event in Ψ_1 we have $\mathsf{time}_1(\sigma,e) = \mathsf{time}_2(\sigma',e)$ for $e \in E_1$. In addition, $\Psi_1 \trianglelefteq_u \Psi_2 \Rightarrow \mathcal{U}_2 \restriction E_1 = \mathcal{U}_1$. From this it follows that events in σ have associated a correct timing in σ'. From the algorithm it is evident that σ'^i consists solely of urgent events, and these events occur as soon as they are enabled. This proves that all events in σ' have associated a correct timing.

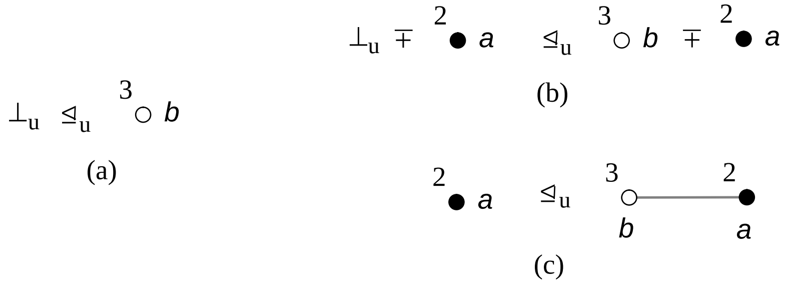
3. since $\sigma \in T_U(\Psi_1)$, σ is time-consistent. In addition, from the algorithm it is evident that (a) σ'^i is time-consistent, and (b) all events in σ'^i have a timing of at least t_{i-1} and at most t_i. This proves that σ' is time-consistent.

4. from the algorithm it follows that for each event e_i in σ there does not exist an urgent event that could have occurred earlier—otherwise such urgent event is included in σ'^i. The same applies to each σ'^i: suppose there is for e'_j in σ'^i an urgent event that could occur earlier, then it would precede e'_j in σ'^i. This proves that for each event in σ' there is no urgent event that could occur earlier.

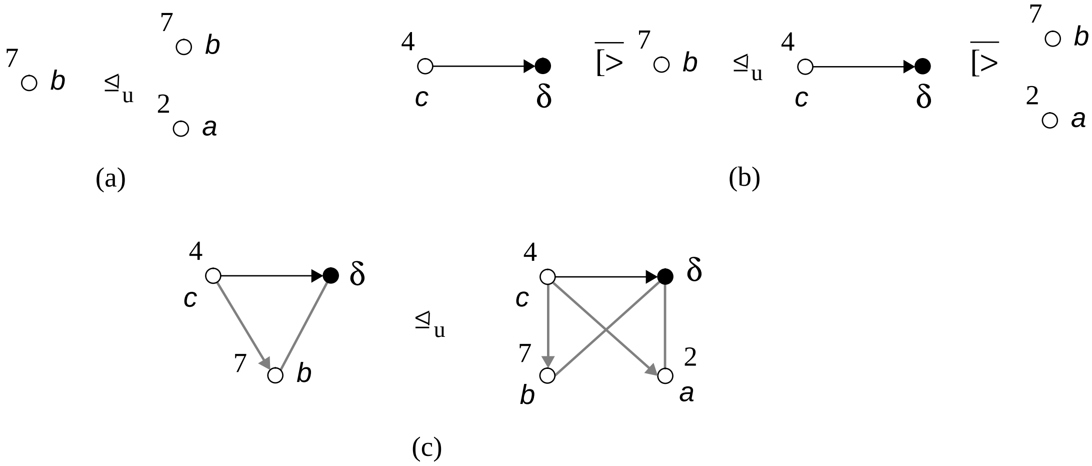
□

Given this result the question arises whether we cannot strengthen Definition 10.41 such that conflicts between urgent events in a next approximation and already existing events are explicitly forbidden. The following examples show that this would not be a solution.

Consider the urgent event structures in the following figure. Obviously, the structures in (a) are ordered, since $\perp_u$ is the least element. Since we want the choice operator to be monotonic we then also would have (b) which equals (c).



In addition, consider the urgent event structures in the following figure. Since we would expect (a) and we want the disable operator to be monotonic we then also have (b) which equals (c).



These examples show that the aforementioned suggestion is not a solution to our problem. So, we should allow the inclusion of new urgent events in conflict with already existing ones.

We conclude this section by characterizing the set of timed event traces of the l.u.b. $\bigsqcup_i \Psi_i$. The following results are all relative to a chain $\Psi_1 \trianglelefteq_u \Psi_2 \trianglelefteq_u \ldots$. It is technically convenient to introduce the following result:

10.52. LEMMA. For $\sigma \in T_U(\bigsqcup_i \Psi_i)$ we have: $\forall\, k : \overline{[\sigma]} \subseteq E_k \;\Rightarrow\; \sigma \in T_U(\Psi_k)$.

PROOF. Let $\sigma \in T_U(\bigsqcup_i \Psi_i)$ for $\bigsqcup_i \Psi_i = \langle \mathcal{E}, \mathcal{D}, \mathcal{T}, \mathcal{U} \rangle$. Let $\Psi_k = \langle \mathcal{E}_k, \mathcal{D}_k, \mathcal{T}_k, \mathcal{U}_k \rangle$ such that $\overline{[\sigma]} \subseteq E_k$. Since $E = \bigcup_i E_i$, Ψ_k is a member of the chain. We prove that $\sigma \in T_U(\Psi_k)$ by systematically checking the conditions of being a timed event trace. Let $\sigma = (e_1, t_1) \ldots (e_n, t_n)$.

1. the proof that $e_1 \ldots e_n \in T(\mathcal{E}) \;\Rightarrow\; e_1 \ldots e_n \in T(\mathcal{E}_k)$ is identical to the proof of Theorem 10.24.

2. $$\forall\, i : (\neg \mathcal{U}(e_i) \;\Rightarrow\; t_i \geqslant \mathsf{time}(\sigma_i, e_i)) \;\wedge\; (\mathcal{U}(e_i) \;\Rightarrow\; t_i \geqslant \mathsf{time}(\sigma_i, e_i))$$

$$\begin{array}{ll}\Leftrightarrow & \{\ \Psi_k \trianglelefteq_t \bigsqcup_i \Psi_i;\ \overline{[\sigma]} \subseteq E_k;\ \text{Lemma 10.22}\ \}\\ & \forall i : (\neg\mathcal{U}(e_i)\ \Rightarrow\ t_i \geqslant \mathsf{time}_k(\sigma_i, e_i))\ \wedge\ (\mathcal{U}(e_i)\ \Rightarrow\ t_i \geqslant \mathsf{time}_k(\sigma_i, e_i))\\ \Leftrightarrow & \{\ \Psi_k \trianglelefteq_t \bigsqcup_i \Psi_i\ \Rightarrow\ \mathcal{U}(e_i) = \mathcal{U}_k(e_i) \text{ for } e_i \in E_k\ \}\\ & \forall i : (\neg\mathcal{U}_k(e_i)\ \Rightarrow\ t_i \geqslant \mathsf{time}_k(\sigma_i, e_i))\ \wedge\ (\mathcal{U}_k(e_i)\ \Rightarrow\ t_i \geqslant \mathsf{time}_k(\sigma_i, e_i))\quad .\end{array}$$

3. by definition, σ is time-consistent.

4.
$$\begin{array}{ll} & \forall i, e \in E : e \in \mathsf{en}([\sigma_i])\ \wedge\ \mathcal{U}(e_i)\ \Rightarrow\ t_i \leqslant \mathsf{time}(\sigma_i, e_i)\\ \Rightarrow & \{\ E_k \subseteq E\ \}\\ & \forall i, e \in E : e \in \mathsf{en}([\sigma_i]) \cap E_k\ \wedge\ \mathcal{U}(e_i)\ \Rightarrow\ t_i \leqslant \mathsf{time}(\sigma_i, e_i)\\ \Leftrightarrow & \{\ \Psi_k \trianglelefteq_t \bigsqcup_i \Psi_i\ \Rightarrow\ \mathcal{U}(e_i) = \mathcal{U}_k(e_i) \text{ for } e_i \in E_k\ \}\\ & \forall i, e \in E_k : e \in \mathsf{en}([\sigma_i]) \cap E_k\ \wedge\ \mathcal{U}_k(e_i)\ \Rightarrow\ t_i \leqslant \mathsf{time}(\sigma_i, e_i)\\ \Leftrightarrow & \{\ \Psi_k \trianglelefteq_t \bigsqcup_i \Psi_i;\ \overline{[\sigma]} \subseteq E_k;\ \text{Lemma 10.8; Lemma 10.22}\ \}\\ & \forall i, e \in E_k : e \in \mathsf{en}_k([\sigma_i])\ \wedge\ \mathcal{U}_k(e_i)\ \Rightarrow\ t_i \leqslant \mathsf{time}_k(\sigma_i, e_i)\quad .\end{array}$$

□

Timed event traces that are present in each approximation from the n-th approximation on are called *n-persistent*.

10.53. Definition. (*n-persistent trace*)

A sequence σ of timed events is *n-persistent* iff $\exists\, n : (\forall\, j \geqslant n : \sigma \in T_U(\Psi_j))$. □

The set of timed event traces of $\bigsqcup_i \Psi_i$ can be characterized as the union of the n-persistent timed traces of its approximations.

10.54. Theorem. $T_U(\bigsqcup_i \Psi_i) = \bigcup_i \bigcap_{j \geqslant i} T_U(\Psi_j)$.

Proof. '$\subseteq$': follows directly from Lemma 10.52.

'$\supseteq$': let $\sigma \in \bigcap_{j \geqslant n} T_U(\Psi_j)$, for some n. Then σ is n-persistent. We prove that $\sigma \in T_U(\bigsqcup_i \Psi_i)$ by contradiction. Assume $\sigma \notin T_U(\bigsqcup_i \Psi_i)$. Since we have that $\Psi_i \trianglelefteq_u \bigsqcup_i \Psi_i$ it follows from Lemma 10.46 that there exists $e \in E$ and e_i in σ such that

$$\mathcal{U}(e)\ \wedge\ e \in \mathsf{en}([\sigma])\ \wedge\ \mathsf{time}(\sigma_i, e_i) > \mathsf{time}(\sigma_i, e).$$

But since $E = \bigcup_i E_i$ and $\mathcal{U} = \bigcup_i \mathcal{U}_i$ it follows that there exists an m with $e \in E_m$, $\overline{[\sigma]} \subseteq E_m$ and $\mathcal{U}_m(e)$ and $\mathsf{time}_m(\sigma_i, e') = \mathsf{time}(\sigma_i, e')$ for all $e' \in E_m$. But this would mean that $\sigma \notin T_U(\Psi_k)$, for all $k \geqslant m$. This contradicts with the fact that σ is n-persistent. □

10.4.2 A fixed point semantics

In this section we consider the denotational semantics of $P := B$ where $B \in \mathsf{PA}_U$. In order to adopt the approach of Sections 10.2.2 and 10.3.2 the crucial issue is to prove that the operators $\overline{(t)\ a_\xi\ ;}$, $\overline{+}$, ..., $\overline{\mathcal{U}_U()}$ are continuous w.r.t. $\trianglelefteq_u$. As for the timed case, it suffices to consider continuity on events.

10.55. LEMMA. For $\langle \mathsf{EBES}_U, \trianglelefteq_u \rangle$ and $F : \mathsf{EBES}_U \longrightarrow \mathsf{EBES}_U$ we have: F is continuous iff F is continuous on events.

PROOF. Similar as the proof of Lemma 10.11. □

Since $\trianglelefteq_u$ is a conservative extension of $\trianglelefteq_t$ it suffices to only prove for all operators in PA_T that the additional constraint on urgent events is satisfied (cf. Definition 10.41), and that the new operator $\overline{\mathcal{U}_U()}$ is continuous. The renaming operator $\phi()$ is defined on urgent event structures as follows.

10.56. DEFINITION. For $\Psi = \langle \Gamma, \mathcal{U} \rangle$ and ϕ an occurrence identifier let $\phi(\Psi) \triangleq \langle \phi(\Gamma), \mathcal{U}' \rangle$ with $\mathcal{U}'(\phi e) = \mathcal{U}(e)$. □

10.57. THEOREM. $\overline{(t)\ a_\xi\,;}$, $\overline{+}$, ..., $\overline{\mathcal{U}_U()}$ and $\phi()$ are continuous on $\langle \mathsf{EBES}_U, \trianglelefteq_u \rangle$.

PROOF. We prove that the operators are continuous on events, which—by Lemma 10.55—proves the case. For the renaming operators $\phi()$ these proofs are trivial and omitted. We prove the theorem for $\overline{(t)\ a_\xi\,;}$, $\overline{+}$, $\overline{||_G}$ and $\overline{\mathcal{U}_U()}$. For each of these constructs we prove continuity on events. The proofs for the other operators are similar and are omitted here. For all cases it suffices to only consider the additional constraints of Definition 10.41 on urgent events. In this proof let $\Psi_i = \langle \Gamma_i, \mathcal{U}_i \rangle$ with $\Gamma_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i \rangle$ and $\mathcal{E}_i = (E_i, \leadsto_i, \mapsto_i, l_i)$ for i=1,2. Similarly Ψ_i' is defined.

1. Action-prefix. Suppose $\Psi_1 \trianglelefteq_u \Psi_2$, let $\Psi_1' = \overline{(t)\ a_\xi\,;}\,\Psi_1$ and $\Psi_2' = \overline{(t)\ a_\xi\,;}\,\Psi_2$. Then:
$\mathcal{U}_2' \restriction E_1' = \mathcal{U}_2' \restriction (\{\,\xi\,\} \cup E_1) = \{\,(\xi, \text{false})\,\} \cup \mathcal{U}_2 \restriction E_1 = \{\,(\xi, \text{false})\,\} \cup \mathcal{U}_1 = \mathcal{U}_1'$.

2. Choice. Suppose $\Psi_1 \trianglelefteq_u \Psi_2$, let $\Psi_1' = \Psi_1 \overline{+} \Psi$ and $\Psi_2' = \Psi_2 \overline{+} \Psi$. Then:
$\mathcal{U}_2' \restriction E_1' = (\mathcal{U}_2 \cup \mathcal{U}) \restriction (E_1 \cup E) = \mathcal{U}_2 \restriction (E_1 \cup E) \cup \mathcal{U} \restriction (E_1 \cup E) = \mathcal{U}_1 \cup \mathcal{U} = \mathcal{U}_1'$.

3. Parallel composition. Suppose $\Psi_1 \trianglelefteq_u \Psi_2$, let $\Psi_1' = \Psi_1 \overline{||_G} \Psi$ and $\Psi_2' = \Psi_2 \overline{||_G} \Psi$. We then prove $\Psi_1' \trianglelefteq_u \Psi_2'$ as follows. According to the definition of $\mathcal{E}_U[\![\ ]\!]$:

$$\mathcal{U}_2'((e_1, e)) \restriction E_1' = (\mathcal{U}_2 \restriction (E_1 \cup \{\,*\,\}))(e_1) \ \vee\ (\mathcal{U} \restriction (E \cup \{\,*\,\}))(e) \quad .$$

We distinguish between the following cases

(a) (e_1, e) is a synchronization event. Then

$$\begin{aligned}
& (\mathcal{U}_2 \restriction (E_1 \cup \{\,*\,\}))(e_1) \ \vee\ (\mathcal{U} \restriction (E \cup \{\,*\,\}))(e) \\
\Leftrightarrow\ & \{\ e_1 \in E_2^s \text{ and } e \in E^s\ \} \\
& (\mathcal{U}_2 \restriction E_1)(e_1) \ \vee\ (\mathcal{U} \restriction E)(e) \\
\Leftrightarrow\ & \{\ \Psi_1 \trianglelefteq_u \Psi_2\ ;\ \mathcal{U} \restriction E = \mathcal{U}\ \} \\
& \mathcal{U}_1(e_1) \ \vee\ \mathcal{U}(e) \\
\Leftrightarrow\ & \{\ \mathcal{E}_1 \trianglelefteq \mathcal{E}_2\ \Rightarrow\ E_1^s \subseteq E_2^s;\ \text{definition } \mathcal{E}_U[\![\ ]\!]\ \} \\
& \mathcal{U}_1'((e_1, e)) \quad .
\end{aligned}$$

(b) $e_1 = *$ and $e \in E^f$. Then:

$$\begin{aligned}
& (\mathcal{U}_2 \restriction (E_1 \cup \{\,*\,\}))(e_1) \ \vee\ (\mathcal{U} \restriction (E \cup \{\,*\,\}))(e) \\
\Leftrightarrow\ & \{\ e \in E^f \text{ and } e_1 = *\ \}
\end{aligned}$$

$$\begin{aligned}
& \text{false} \ \vee \ (\mathcal{U} \restriction E)(e) \\
\Leftrightarrow \ & \{ \ \mathcal{U} \restriction E = \mathcal{U} \ \} \\
& \mathcal{U}_1(*) \ \vee \ \mathcal{U}(e) \\
\Leftrightarrow \ & \{ \text{ definition } \mathcal{E}_U[\![\ ]\!] \ \} \\
& \mathcal{U}'_1((*, e)) \quad .
\end{aligned}$$

(c) $e_1 \in E_2^f$ and $e = *$. Similar to the previous case and omitted.

4. Urgency. Suppose $\Psi_1 \trianglelefteq_u \Psi_2$, and let $\Psi'_1 = \overline{\mathcal{U}_U(\Psi_1)}$ and $\Psi'_2 = \overline{\mathcal{U}_U(\Psi_2)}$. We prove that $\Psi'_1 \trianglelefteq_u \Psi'_2$ by checking the conditions of $\trianglelefteq_u$.

   (a) Since $\Gamma'_1 = \Gamma_1$ and $\Gamma'_2 = \Gamma_2$, it follows directly from $\Psi_1 \trianglelefteq_u \Psi_2$ that $\Gamma'_1 \trianglelefteq_t \Gamma'_2$.

   (b) For $e \in E'_1$ we derive

$$\begin{aligned}
& \mathcal{U}'_2(e) \\
\Leftrightarrow \ & \{ \text{ definition } \mathcal{E}_U[\![\ ]\!] \ \} \\
& \mathcal{U}_2(e) \ \vee \ l_2(e) \in U \\
\Leftrightarrow \ & \{ \ \Psi_1 \trianglelefteq_u \Psi_2; \, e \in E'_1 \Leftrightarrow e \in E_1 \ \} \\
& \mathcal{U}_1(e) \ \vee \ l_2(e) \in U \\
\Leftrightarrow \ & \{ \ \mathcal{E}_1 \trianglelefteq \mathcal{E}_2 \ \Rightarrow \ l_1 = l_2 \restriction E_1; \, e \in E_1 \ \} \\
& \mathcal{U}_1(e) \ \vee \ l_1(e) \in U \\
\Leftrightarrow \ & \{ \text{ definition } \mathcal{E}_U[\![\ ]\!] \ \} \\
& \mathcal{U}'_1(e) \quad .
\end{aligned}$$

This proves that $\overline{\mathcal{U}_U()}$ is monotonic. Continuity on events follows from:

$$E\left(\overline{\mathcal{U}_U(\bigsqcup_i \Psi_i)}\right) = E\left(\bigsqcup_i \Psi_i\right) = \bigcup_i E(\Psi_i) = \bigcup_i E(\overline{\mathcal{U}_U(\Psi_i)}) = E\left(\bigsqcup_i \overline{\mathcal{U}_U(\Psi_i)}\right) .$$

□

In the following definition let $\mathcal{H}_B$ be the urgent counterpart of $\mathcal{F}_B$. $\mathcal{H}_B$ is a function determined by $\overline{\mathsf{op}}$ and $\phi()$. From the previous theorem it follows that $\mathcal{H}_B$ is continuous on urgent event structures ordered under $\trianglelefteq_u$. This means that the semantics of $P := B$ for $B \in \mathsf{PA}_U$ can now be computed as the l.u.b. of $\perp_u, \mathcal{H}_B(\perp_u), \mathcal{H}_B(\mathcal{H}_B(\perp_u)), \ldots$.

10.58. Definition. For $P := B$ a process definition let $\mathcal{E}_U[\![P]\!] \triangleq \bigsqcup_i \mathcal{H}_B^i(\perp_u)$. □

10.59. Example. As an example of a recursive process definition in PA_U we consider

$$P := \mathcal{U}_a((2) \ a \, ; \, P \,|||\, (11) \ b \, ; \, \mathbf{0}) \quad .$$

The first approximation is $\perp_u$. The second and third approximation $\mathcal{H}_B(\perp_u)$ resp. $\mathcal{H}_B^2(\perp_u)$ are depicted in Figure 10.5(a) and (b), respectively. Notice that $(e_a, 2)(e_b, 11)$ is a timed event trace of (a), but not of (b), since the introduced event labelled a is forced to occur at time 4, so before e_b. By repeated substitution we obtain the urgent event structure of Figure 10.5(c). □

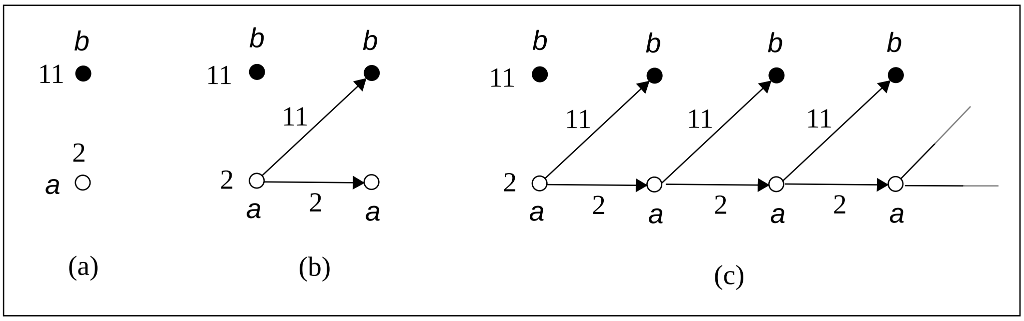


Figure 10.5: Example of semantics for a recursive process definition in PA_U.

10.4.3 Event-based operational semantics

In this section we consider the extension of the operational semantics of PA_U with recursion, and show its consistency with the causality-based semantics defined just before. For the inference rules we adopt the approach of Section 10.3.3. The additional inference rules for PA_U are presented in Table 10.2. Notice the resemblance with the rules for PA_T as listed in Table 10.1.

$$\frac{\langle B,t\rangle \xrightarrow{(\xi,a)} \langle B',t\rangle}{\langle P_\phi,t\rangle \xrightarrow{(\phi\xi,a)} \langle \phi(B'),t\rangle}\ (P := B) \qquad \frac{\langle B,t\rangle \rightsquigarrow \langle B',t'\rangle}{\langle P_\phi,t\rangle \rightsquigarrow \langle \phi(B'),t'\rangle}\ (P := B)$$

$$\frac{\langle B,t\rangle \xrightarrow{(\xi,a)} \langle B',t\rangle}{\langle \phi(B),t\rangle \xrightarrow{(\phi\xi,a)} \langle \phi(B'),t\rangle} \qquad \frac{\langle B,t\rangle \rightsquigarrow \langle B',t'\rangle}{\langle \phi(B),t\rangle \rightsquigarrow \langle \phi(B'),t'\rangle}$$

Table 10.2: Additional transition rules for PA_U.

Recall that the passage of time for $\mathcal{U}_U(B)$ is restricted by the d_{min} function. For $P := B$ let $d_{min}(a,P) \triangleq d_{min}(a,B)$. In order to let this definition make sense we require P to be *guarded*. This means that all process instantiations in the body B of P must be preceded by a timed action-prefix or a sequential composition. For instance, $P := (2)\ a\,;\, P + \surd >> Q$ is guarded, whereas $P := (2)\ a\,;\, P\,|||\,Q$ is not. A recursive process definition $P := B$ is considered to be *weakly guarded* if B becomes guarded by substituting for a finite number of times the bodies of processes for the process instantiations occurring in B. For instance, $P := (2)\ a\,;\, P\,|||\,Q$ where $Q := (3)\ b\,;\, Q$ is weakly guarded, since it can be rewritten into the guarded $P := (2)\ a\,;\, P\,|||\,(3)\ b\,;\, Q$ by a single substitution. From now on we require for $P := B$ that B is weakly guarded.

In order to prove the consistency between the denotational and event-based operational semantics for the urgent case the approach of Section 10.3.3 fails, since the set of timed event traces generated operationally cannot be characterized by substituting $\varnothing$ for all occurrences of P in B, and then continuing by approximation. We therefore take a different route here.

10.60. DEFINITION. (*Substitution on terms*)

For $B, B' \in \mathsf{PA}_U$ and P a process instantiation, $B'[P := B]$, is defined as

$$\begin{array}{rcl}
\mathbf{0}[P := B] & \triangleq & \mathbf{0} \\
\surd[P := B] & \triangleq & \surd \\
(\mathsf{op}\, B_1)[P := B] & \triangleq & \mathsf{op}\, B_1[P := B] \text{ for } \mathsf{op} \in \{\, a\,;\,, \backslash, [\,], \mathcal{U}_U()\,\} \\
(B_1\, \mathsf{op}\, B_2)[P := B] & \triangleq & B_1[P := B]\, \mathsf{op}\, B_2[P := B] \text{ for } \mathsf{op} \in \{\, +, \gg, [>, \,||\,\} \\
Q[P := B] & \triangleq & \begin{cases} \phi(B) & \text{if } Q = P_\phi \\ Q & \text{if } Q \neq P. \end{cases}
\end{array}$$

□

$B'[P := B]$ denotes behaviour B' where all occurrences of P_ϕ in B' are replaced with $\phi(B)$. As a next subsidiary notion we define the unfoldings of P.

10.61. DEFINITION. (*Unfoldings of P*)

For $P := B$ the n-th *unfolding* of P, denoted $\hat{P}^n$, is defined as:

$$\begin{array}{rcl}
\hat{P}^0 & \triangleq & P \\
\hat{P}^{n+1} & \triangleq & B[P := \hat{P}^n].
\end{array}$$

□

The n-th approximation of P is defined as the n-th unfolding where each occurrence of P is replaced by $\mathbf{0}$.

10.62. DEFINITION. (*Approximations of P*)

For $P := B$ the n-th *approximation* of P, denoted P^n, is defined as $P^n \triangleq \hat{P}^n[P := \mathbf{0}]$.
□

The set of timed event traces of P is equal to that of its n-th unfolding.

10.63. LEMMA. $\forall\, n \geqslant 0 : \mathcal{T}_U[\![\, P \,]\!] = \mathcal{T}_U[\![\, \hat{P}^n \,]\!]$.

PROOF. By induction on n. Let $P := B$.
Base: for n=0 we have according to Definition 10.61 $\mathcal{T}_U[\![\, \hat{P}^0 \,]\!] = \mathcal{T}_U[\![\, P \,]\!]$.
Induction step: Assume the lemma holds for n=k and consider k+1.

$$\begin{array}{rl}
& \mathcal{T}_U[\![\, \hat{P}^{k+1} \,]\!] \\
= & \{\text{ Definition 10.61 }\} \\
& \mathcal{T}_U[\![\, B[P := \hat{P}^k] \,]\!] \\
= & \{\text{ induction hypothesis; substitution preserves trace equivalence }\} \\
& \mathcal{T}_U[\![\, B[P := P] \,]\!] \\
= & \{\text{ Definition 10.60; } P := B \text{ }\} \\
& \mathcal{T}_U[\![\, P \,]\!] \quad .
\end{array}$$

□

Timed event traces of the n-the unfolding of P are also timed event traces of the n-th approximation of P.

10.64. LEMMA. $\forall\, n \geqslant 0 : \mathcal{T}_U[\![\, P^n \,]\!] \subseteq \mathcal{T}_U[\![\, \hat{P}^n \,]\!]$.

PROOF. By induction on n.
Base: For n=0 we derive

$$
\begin{array}{ll}
 & \sigma \in \mathcal{T}_U[\![\, P^0 \,]\!] \\
\Leftrightarrow & \{ \text{ Definition 10.62 } \} \\
 & \sigma \in \mathcal{T}_U[\![\, \hat{P}^0[P := \mathbf{0}] \,]\!] \\
\Leftrightarrow & \{ \text{ Definition 10.61 } \} \\
 & \sigma \in \mathcal{T}_U[\![\, P[P := \mathbf{0}] \,]\!] \\
\Leftrightarrow & \{ \text{ Definition 10.60 } \} \\
 & \sigma \in \mathcal{T}_U[\![\, \mathbf{0} \,]\!] \\
\Rightarrow & \{ \mathcal{T}_U[\![\, \mathbf{0} \,]\!] = \{ \varepsilon \};\ \varepsilon \in \mathcal{T}_U[\![\, B \,]\!] \text{ for all } B \ \} \\
 & \sigma \in \mathcal{T}_U[\![\, \hat{P}^0 \,]\!] \quad .
\end{array}
$$

Induction step: Assume the lemma holds for n=k and consider k+1.

$$
\begin{array}{ll}
 & \sigma \in \mathcal{T}_U[\![\, P^{k+1} \,]\!] \\
\Leftrightarrow & \{ \text{ Definition 10.62; Definition 10.61 } \} \\
 & \sigma \in \mathcal{T}_U[\![\, B[P := \hat{P}^k][P := \mathbf{0}] \,]\!] \\
\Leftrightarrow & \{ \text{ substitution property } \} \\
 & \sigma \in \mathcal{T}_U[\![\, B[P := \hat{P}^k[P := \mathbf{0}]] \,]\!] \\
\Leftrightarrow & \{ \text{ Definition 10.62 } \} \\
 & \sigma \in \mathcal{T}_U[\![\, B[P := P^k] \,]\!] \\
\Rightarrow & \{ \text{ induction hypothesis } \} \\
 & \sigma \in \mathcal{T}_U[\![\, B[P := \hat{P}^k] \,]\!] \\
\Leftrightarrow & \{ \text{ Definition 10.62 } \} \\
 & \sigma \in \mathcal{T}_U[\![\, \hat{P}^{k+1} \,]\!] \quad .
\end{array}
$$

□

If $B \xrightarrow{(e,a,t)}_* B'$ and B is guarded then this transition can be derived without applying one of the transition rules for recursive process behaviours (cf. Table 10.2) due to the guardedness of B. But then, the process instantiations occurring in B may be replaced by some arbitrary expression X without prohibiting this transition.

10.65. LEMMA. Let $B \in \mathsf{PA}_U$ such that B is guarded. Then for arbitrary $X \in \mathsf{PA}_U$ and process identifier P we have:

$$B \xrightarrow{(e,a,t)}_* B' \;\Rightarrow\; B[P := X] \xrightarrow{(e,a,t)}_* B'[P := X] \quad .$$

PROOF. Straightforward by induction on B. □

The following lemma is based on the intuition that traces of length at most n can involve at most n unfoldings of process instantiations. More precisely, it states that if σ is a timed trace of B' where all occurrences of P are replaced by its n-th unfolding $\hat{P}^n$ and $|\,\sigma\,| \leqslant n$, then P may be replaced in the resulting term by an arbitrary term X while preserving that σ is a timed trace.

10.66. LEMMA. Let $B' \in \mathsf{PA}_U$ possibly containing unguarded occurrences of P, for $P := B$ and B guarded. Then for arbitrary term $X \in \mathsf{PA}_U$:

$$\forall\, n \geqslant 0 : \sigma \in \mathcal{T}_U[\![\, B'[P := \hat{P}^n]\,]\!] \;\wedge\; |\,\sigma\,| \leqslant n \;\Rightarrow\; \sigma \in \mathcal{T}_U[\![\, B'[P := \hat{P}^n[P := X]]\,]\!].$$

PROOF. By induction on n.
Base: for n=0 the lemma trivially holds since ε is a trace of each behaviour.
Induction step: Assume the lemma holds for n=k; consider k+1. First we derive

$$\begin{array}{ll} & B'[P := \hat{P}^{k+1}] \\ = & \{ \text{ Definition 10.61 } \} \\ & B'[P := B[P := \hat{P}^k]] \\ = & \{ \text{ substitution property } \} \\ & B'[P := B][P := \hat{P}^k] \quad . \end{array}$$

In a similar way we can derive that

$$B'[P := \hat{P}^{k+1}[P := X]] = B'[P := B][P := \hat{P}^k[P := X]] \quad . \tag{10.1}$$

Now assume $\sigma \in \mathcal{T}_U[\![\, B'[P := B][P := \hat{P}^k]\,]\!]$ with $\sigma = (e,a,t)\,\sigma'$ and $|\,\sigma'\,| = k$. Then:

$$\begin{array}{ll} & \sigma \in \mathcal{T}_U[\![\, B'[P := \hat{P}^{k+1}[P := X]]\,]\!] \\ \Leftrightarrow & \{ \text{ (10.1) } \} \\ & \sigma \in \mathcal{T}_U[\![\, B'[P := B][P := \hat{P}^k[P := X]]\,]\!] \\ \Leftrightarrow & \{ \ \sigma = (e,a,t)\,\sigma';\ B \text{ is guarded } \} \\ & \quad B'[P := B][P := \hat{P}^k[P := X]] \xrightarrow{(e,a,t)}_* B''[P := \hat{P}^k[P := X]] \\ & \ \wedge\ \sigma' \in \mathcal{T}_U[\![\, B''[P := \hat{P}^k[P := X]]\,]\!] \\ \Leftarrow & \{ \ B'[P := B] \text{ is guarded (since } B \text{ is); Lemma 10.65 } \} \\ & B'[P := B][P := \hat{P}^k] \xrightarrow{(e,a,t)}_* B''[P := \hat{P}^k] \ \wedge\ \sigma' \in \mathcal{T}_U[\![\, B''[P := \hat{P}^k[P := X]]\,]\!] \\ \Leftarrow & \{ \text{ induction hypothesis } \} \\ & B'[P := B][P := \hat{P}^k] \xrightarrow{(e,a,t)}_* B''[P := \hat{P}^k] \ \wedge\ \sigma' \in \mathcal{T}_U[\![\, B''[P := \hat{P}^k]\,]\!] \\ \Leftrightarrow & \{ \ \sigma = (e,a,t)\,\sigma' \ \} \\ & \sigma \in \mathcal{T}_U[\![\, B'[P := B][P := \hat{P}^k]\,]\!] \\ \Leftrightarrow & \{ \text{ assumption } \} \\ & \text{true} \quad . \end{array}$$

□

The set of timed event traces of of P is equal to the union of the sets of i-persistent timed event traces for all i.

10.67. THEOREM. For $P := B$ we have $\mathcal{T}_U[\![\, P\,]\!] = \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}_U[\![\, P^j\,]\!]$.

PROOF. '$\subseteq$':

$$\begin{array}{ll} & \sigma \in \mathcal{T}_U[\![\, P\,]\!] \ \wedge\ |\,\sigma\,| \leqslant n \\ \Leftrightarrow & \{ \text{ Definition 10.60 } \} \end{array}$$

$$\begin{array}{ll} & \sigma \in \mathcal{T}_U[\![\, P[P := P] \,]\!] \;\wedge\; |\,\sigma\,| \leqslant n \\ \Leftrightarrow & \{ \text{ Lemma 10.63 } \} \\ & \sigma \in \mathcal{T}_U[\![\, P[P := \hat{P}^n] \,]\!] \;\wedge\; |\,\sigma\,| \leqslant n \\ \Rightarrow & \{ \text{ Lemma 10.66 } \} \\ & \sigma \in \mathcal{T}_U[\![\, P[P := \hat{P}^n[P := \mathbf{0}]] \,]\!] \\ \Leftrightarrow & \{ \text{ Definition 10.62 } \} \\ & \sigma \in \mathcal{T}_U[\![\, P[P := P^n] \,]\!] \\ \Leftrightarrow & \{ \text{ Definition 10.60 } \} \\ & \sigma \in \mathcal{T}_U[\![\, P^n \,]\!] \quad . \end{array}$$

Since this holds for all n it immediately follows that $\sigma \in \mathcal{T}_U[\![\, P \,]\!] \;\Rightarrow\; \sigma \in \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}_U[\![\, P^j \,]\!]$.

'$\supseteq$':

$$\begin{array}{ll} & \sigma \in \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}_U[\![\, P^j \,]\!] \\ \Leftrightarrow & \{ \text{ calculus } \} \\ & \forall\, j \geqslant i : \sigma \in \mathcal{T}_U[\![\, P^j \,]\!] \\ \Rightarrow & \{ \text{ Lemma 10.64 } \} \\ & \forall\, j \geqslant i : \sigma \in \mathcal{T}_U[\![\, \hat{P}^j \,]\!] \\ \Leftrightarrow & \{ \text{ Lemma 10.63 } \} \\ & \sigma \in \mathcal{T}_U[\![\, P \,]\!] \quad . \end{array}$$

□

Then we have the following consistency result between the denotational semantics in terms of urgent event structures and the event-based operational semantics.

10.68. THEOREM. For $P := B$ we have $T_U(\mathcal{E}_U[\![\, P \,]\!]) = \mathcal{T}_U[\![\, P \,]\!]$.

PROOF.

$$\begin{array}{ll} & T_U(\mathcal{E}_U[\![\, P \,]\!]) \\ = & \{ \text{ Definition 10.58 } \} \\ & T_U(\bigsqcup_i \mathcal{H}^i_B(\bot_u)) \\ = & \{ \text{ Theorem 10.54 } \} \\ & \bigcup_i \bigcap_{j \geqslant i} T_U(\mathcal{H}^j_B(\bot_u)) \\ = & \{ \; \} \\ & \bigcup_i \bigcap_{j \geqslant i} T_U(P^j) \\ = & \{ \text{ Theorem 6.34 } \} \\ & \bigcup_i \bigcap_{j \geqslant i} \mathcal{T}_U[\![\, P^j \,]\!] \\ = & \{ \text{ Theorem 10.67 } \} \\ & \mathcal{T}_U[\![\, P \,]\!] \quad . \end{array}$$

□

10.5 Real-time event structures

In this section we extend the results of Section 10.3 for real-time event structures. The definitions in this section are all relative to real-time event structure $\Lambda_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$ for $i{=}1,2$.

10.69. Definition. (*Partial order on real-time event structures*)

$\Lambda_1 \trianglelefteq_r \Lambda_2$ iff

1. $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$
2. $\mathcal{D}_1 = \mathcal{D}_2 \restriction E_1$
3. $\forall\, e \in E_1 : \mathcal{T}_1((X \cap E_1, e)) = \mathcal{T}_2((X, e))$
4. $\mathcal{U}_1 = \mathcal{U}_2 \restriction E_1$.

□

This ordering is identical to the ordering of urgent event structures (except for the fact that $\mathcal{D}$ and $\mathcal{T}$ are dealing with sets of time instants rather than time instants). It follows in the same way as in Section 10.4 that $\langle \mathsf{EBES}_R, \trianglelefteq_r \rangle$ is a pointed complete c.p.o.. Also characterizations of l.u.b., timed event traces of $\bigsqcup_i \Lambda_i$, and so on, are identical to the urgent case. It remains to check continuity on events of $\triangleright$ and $\blacktriangleright$.

10.70. Theorem. $\overline{(T)\ a_\xi\ ;}$, $\overline{+}$, $\overline{\triangleright_\xi}$, $\overline{\blacktriangleright}$, … and $\phi()$ are continuous on $\langle \mathsf{EBES}_R, \trianglelefteq_r \rangle$.

Proof. For all operators, except for the new operators $\blacktriangleright$ and $\triangleright$, the proof is identical to that of continuity in the urgent case. We prove the theorem for $\blacktriangleright$. For $\triangleright$ the theorem follows immediately since $\overline{B_1 \overset{t}{\triangleright}_\xi}\ B_2$ is modelled as $B_1 + ([t,t])\ \tau_\xi\ ;\ B_2$ where τ is urgent, the fact that $\trianglelefteq_r$ equals $\trianglelefteq_u$, and that $\overline{(t)\, a_\xi\ ;}$, $\overline{+}$, and $\overline{\mathcal{U}_U()}$ are continuous on $\langle \mathsf{EBES}_U, \trianglelefteq_u \rangle$.

In this proof let $\Lambda_i = \langle \mathcal{E}_i, \mathcal{D}_i, \mathcal{T}_i, \mathcal{U}_i \rangle$ with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for $i{=}1,2$. Similarly Λ_i' is defined.

Suppose $\Lambda_1 \trianglelefteq_r \Lambda_2$ and let $\Lambda_1' = \Lambda \overline{\overset{t}{\blacktriangleright}} \Lambda_1$ and $\Lambda_2' = \Lambda \overline{\overset{t}{\blacktriangleright}} \Lambda_2$. We prove that $\Lambda_1' \trianglelefteq_r \Lambda_2'$ by systematically checking the constraints of $\trianglelefteq_r$.

1. $\mathcal{E}_1' \trianglelefteq \mathcal{E}_2'$ follows directly from the fact that the 'plain' event structure of Λ_i', for $i{=}1,2$, is identical to that of $\Lambda\ \overline{[\!>}\ \Lambda_i$, and the fact that $\overline{[\!>}$ is continuous.

2. $$\begin{aligned}
& \mathcal{D}_2' \restriction E_1' \\
= \quad & \{ \text{ definition } \mathcal{E}_R[\![\ ]\!] \ \} \\
& \mathcal{D}_2' \restriction (E \cup E_1) \\
= \quad & \{ \text{ definition } \mathcal{E}_R[\![\ ]\!] \ \} \\
& \{ (e, \mathcal{D}(e) \cap [0,t]) \mid e \in E \} \cup (\{ (e, t{+}\mathcal{D}_2(e)) \mid e \in E_2 \} \restriction E_1) \\
= \quad & \{ \ \Lambda_1 \trianglelefteq_r \Lambda_2 \ \Rightarrow\ E_1 \subseteq E_2 \ \wedge\ \mathcal{D}_2 \restriction E_1 = \mathcal{D}_1 \ \} \\
& \{ (e, \mathcal{D}(e) \cap [0,t]) \mid e \in E \} \cup \{ (e, t{+}\mathcal{D}_1(e)) \mid e \in E_1 \} \\
= \quad & \{ \text{ definition } \mathcal{E}_R[\![\ ]\!] \ \} \\
& \mathcal{D}_1' \quad .
\end{aligned}$$

3. $\mathcal{T}_2'((X_2', e)) = (\mathcal{T} \cup \mathcal{T}_2)((X_2', e)) = (\mathcal{T} \cup \mathcal{T}_1)((X_2' \cap E_1, e)) = \mathcal{T}_1'((X_2' \cap E_1, e))$.
4. $\mathcal{U}_2' \upharpoonright E_1' = (\mathcal{U} \cup \mathcal{U}_2) \upharpoonright (E \cup E_1) = \mathcal{U} \cup (\mathcal{U}_2 \upharpoonright E_1) = \mathcal{U} \cup \mathcal{U}_1 = \mathcal{U}_1'$.

This proves that $\overline{\blacktriangleright}^{t}$ is monotonic in the right argument. The proof for monotonicity in the left argument is obtained in a similar way. In addition we have

$$E\left((\bigsqcup_i \Lambda_i)\,\overline{\blacktriangleright}\,\Lambda\right) = E\left((\bigsqcup_i \Lambda_i)\,\overline{[>}\,\Lambda\right) = E\left(\bigsqcup_i(\Lambda_i\,\overline{[>}\,\Lambda)\right) = E\left(\bigsqcup_i(\Lambda_i\,\overline{\blacktriangleright}\,\Lambda)\right) \quad .$$

This proves that $\overline{\blacktriangleright}$ is continuous on events. ☐

The event-based operational semantics of PA_R can be extended in the same way as for PA_T, that is, by incorporating the inference rules:

$$\frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{P_\phi \xrightarrow{(\phi\xi,a,t)}\!\!\!\!\rightarrow \phi(B')}\ (P := B) \qquad \frac{B \xrightarrow{(\xi,a,t)}\!\!\!\!\rightarrow B'}{\phi(B) \xrightarrow{(\phi\xi,a,t)}\!\!\!\!\rightarrow \phi(B')}$$

The function ut is extended for process instantiation P such that $\mathsf{ut}(P) \triangleq \mathsf{ut}(B)$ for $P := B$. In order to let ut be well-defined we require $P := B$ to be weakly guarded, i.e., B should become guarded by substituting for a finite number of times the bodies for the process instantiations occurring in B.

10.6 Stochastic event structures

The definitions in this section are all relative to stochastic event structures $\Sigma_i = \langle \mathcal{E}_i, \mathcal{F}_i, \mathcal{G}_i \rangle$ with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2. For this case we only provide definitions of the partial order ($\unlhd_s$) and the l.u.b.. From these definitions it will be clear that the results from the deterministic timed case can be carried over to the stochastic setting in an easy way.

10.71. DEFINITION. (*Partial order on stochastic event structures*)
Let $X_i \subseteq E_i$ for i=1, 2. Then $\Sigma_1 \unlhd_s \Sigma_2$ iff

1. $\mathcal{E}_1 \unlhd \mathcal{E}_2$
2. $\mathcal{F}_2 \upharpoonright E_1 = \mathcal{F}_1$
3. $\forall\, e \in E_1 : \mathcal{G}_2((X_2, e)) = \mathcal{G}_1((X_2 \cap E_1, e))$.

☐

10.72. LEMMA. $\langle \mathsf{EBES}_S, \unlhd_s \rangle$ is a pointed c.p.o..

PROOF. Routine and omitted ☐

It is easy to verify that $\perp_s = \langle \perp, \varnothing, \varnothing \rangle$, the empty stochastic event structure, is the least element under $\unlhd_s$.

10.73. DEFINITION. (*Least upper bound (under $\trianglelefteq_s$)*)

Let $\Sigma_1 \trianglelefteq \Sigma_2 \trianglelefteq \ldots$ be a chain, then $\bigsqcup_i \Sigma_i \triangleq \langle \bigsqcup_i \mathcal{E}_i, \bigcup_i \mathcal{F}_i, \mathcal{G} \rangle$ with

$$\mathcal{G} = \{ ((\bigcup_k X_k, e), F) \mid \exists\, j : (\forall\, k \geqslant j : X_k \overset{F}{\mapsto}_k e \;\wedge\; X_{k+1} \cap E_k = X_k) \}.$$

□

10.74. LEMMA. $\bigsqcup_i \Sigma_i$ is the least upper bound of chain $\Sigma_1 \trianglelefteq_s \Sigma_2 \trianglelefteq_s \ldots$.

PROOF. Similar as the proof of Lemma 10.21. □

Given the definitions of $\trianglelefteq_s$ and $\bigsqcup_i \Sigma_i$ it is now straightforward to define a continuous function $\mathcal{F}_B$ in a similar way as for the deterministic timed case. The semantics $\mathcal{E}_S[\![\, P \,]\!]$ is then defined as the l.u.b. of the sequence $\perp_s, \mathcal{F}_B(\perp_s), \ldots$. We will not bother the reader with the details here.

10.7 Probabilistic event structures

In this section we will consider recursion in the probabilistic setting (as introduced in Chapter 9). Section 10.7.1 defines a c.p.o. $\trianglelefteq_p$ on probabilistic event structures and characterizes a l.u.b. of chains under this ordering. $\trianglelefteq_p$ is shown to satisfy the nice properties, such as preservation of trace sets. Section 10.7.2 proves all operators, including $+_p$, to be continuous on $\trianglelefteq_p$ and provides a denotational semantics of $P := B$ for weakly guarded B. Section 10.7.3 presents an event-based operational semantics for $P := B$.

10.7.1 A pointed complete partial order

The definitions and results in this section are all relative to probabilistic event structures $\Pi_i = \langle \mathcal{E}_i, \pi_i \rangle$ with $\mathcal{E}_i = (E_i, \leadsto_i, \mapsto_i, l_i)$ for i=1, 2.

10.75. DEFINITION. (*Partial order on probabilistic event structures*)

Let $\Pi_1 \trianglelefteq_p \Pi_2$ iff $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$ and $\pi_1 = \pi_2 \restriction E_1$. □

Π_1 is 'smaller than' Π_2 iff their event structures are smaller (i.e., $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$) and events in Π_1 are only assigned a probability in Π_2 if this was done in Π_1 and this probability does not change.

10.76. LEMMA. $\langle \mathsf{EBES}_P, \trianglelefteq_p \rangle$ is a pointed c.p.o..

PROOF. Routine and omitted. □

It is easy to show that $\perp_p = \langle \perp, \varnothing \rangle$, the empty probabilistic event structure, is the least element of $\langle \mathsf{EBES}_P, \trianglelefteq_p \rangle$.

10.77. EXAMPLE. Consider the probabilistic event structures of Figure 10.6, referred to as (a) Π_1, (b) Π_2, and (c) Π_3, and assume equally labelled events in different structures to be

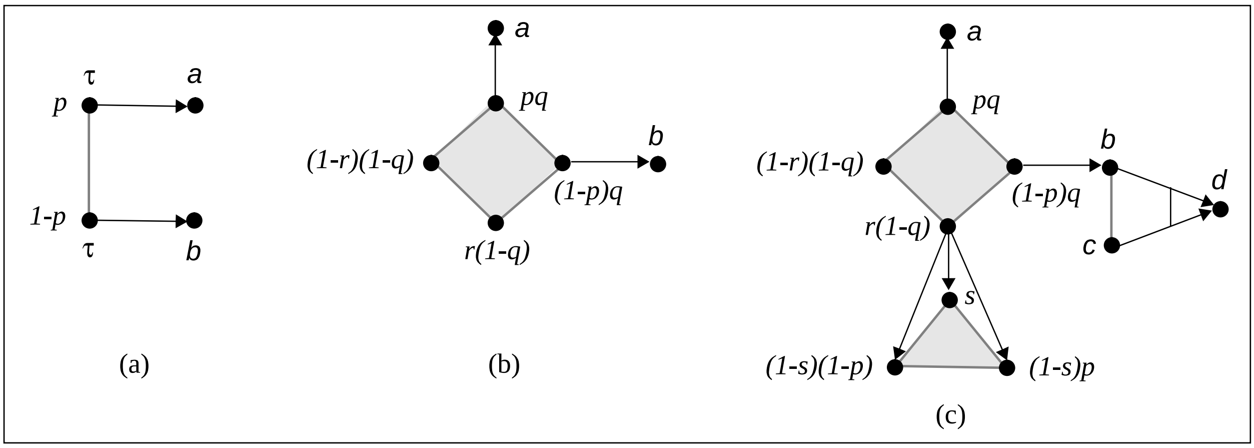


Figure 10.6: Probabilistic event structures with (a) $\not\trianglelefteq_p$ (b) and (b) $\trianglelefteq_p$ (c).

the same. We have $\mathcal{E}_1 \not\trianglelefteq \mathcal{E}_2$, since $\pi_1 \neq \pi_2 \restriction E_1$. The reader should be able to verify that $\Pi_2 \trianglelefteq_p \Pi_3$ without great difficulty. □

For chain $\Pi_1 \trianglelefteq_p \Pi_2 \trianglelefteq_p \ldots$ let $\bigsqcup_i \Pi_i$ be defined as follows. The probability function π is the union of the probability functions of the elements in the chain.

10.78. DEFINITION. (*Least upper bound (under $\trianglelefteq_p$)*)

Let $\Pi_1 \trianglelefteq_p \Pi_2 \trianglelefteq_p \ldots$ be a chain, then $\bigsqcup_i \Pi_i \triangleq \langle \bigsqcup_i \mathcal{E}_i, \bigcup_i \pi_i \rangle$. □

10.79. LEMMA. $\bigsqcup_i \Pi_i$ is the least upper bound of chain $\Pi_1 \trianglelefteq_p \Pi_2 \trianglelefteq_p \ldots$.

PROOF. Routine and omitted. □

The following theorem lists some properties of $\trianglelefteq_p$.

10.80. THEOREM. We have:

1. $\Pi_1 \trianglelefteq_p \Pi_2 \;\Rightarrow\; T_P(\Pi_1) \subseteq T_P(\Pi_2)$.
2. $(\Pi_1 \trianglelefteq_p \Pi_2 \;\wedge\; E_1 = E_2) \;\Rightarrow\; \Pi_1 = \Pi_2$.
3. $T_P(\bigsqcup_i \Pi_i) = \bigcup_i T_P(\Pi_i)$.
4. $\Pi_1 \trianglelefteq_p \Pi_2 \;\Rightarrow\; \mathsf{cl}(\Pi_1) \subseteq \mathsf{cl}(\Pi_2)$.

PROOF. 1. and 3. follow directly from Theorem 10.6 and the fact that $T_P(\Pi_i) = T(\mathcal{E}_i)$, for i=1,2. 4. follows directly from the definition of $\trianglelefteq_p$. For 2. suppose $\Pi_1 \trianglelefteq_p \Pi_2$ and $E_1 = E_2$. Then $\mathcal{E}_1 \trianglelefteq \mathcal{E}_2$, and by Theorem 10.6, $\mathcal{E}_1 = \mathcal{E}_2$. Since $\Pi_1 \trianglelefteq_p \Pi_2$ we have $\pi_1 = \pi_2 \restriction E_1 = \pi_2 \restriction E_2 = \pi_2$. So, $\Pi_1 = \Pi_2$. □

10.7.2 A fixed point semantics

In this section we consider the semantics of $P := B$ where $B \in \mathsf{PA}_P$. Again, we first have to prove that the operators $\overline{a_\xi\,;}, \overline{+}, \overline{+_p}, \ldots$ are continuous w.r.t. $\trianglelefteq_p$. This is similar to the timed and urgent case discussed before, due to:

10.81. LEMMA. For $\langle \mathsf{EBES}_P, \trianglelefteq_p \rangle$ and $F : \mathsf{EBES}_P \longrightarrow \mathsf{EBES}_P$ we have: F is continuous iff F is continuous on events.

PROOF. Similar to the proof of Lemma 10.11. $\square$

The renaming operator on event structures is extended to probabilistic ones as follows.

10.82. DEFINITION. For $\Pi = \langle \mathcal{E}, \pi \rangle$ and ϕ an occurrence identifier let $\phi(\Pi) \triangleq \langle \phi(\mathcal{E}), \pi' \rangle$ with $\pi'(\phi e) = \pi(e)$ for $\phi e \in \phi(dom(\pi))$. $\square$

10.83. THEOREM. $\overline{a_\xi\,;}, \overline{+}, \overline{+_p}, \ldots$ and $\phi()$ are continuous on $\langle \mathsf{EBES}_P, \trianglelefteq_p \rangle$.

PROOF. We prove that the operators are continuous on events, which—by Lemma 10.81—proves the case. For the renaming operators $\phi()$ these proofs are trivial and omitted. We prove the theorem for $\overline{a_\xi\,;}, \overline{+}, \overline{+_p}$ and $\overline{||_G}$. The proofs for the other cases are similar and omitted here. For the treated constructs it suffices to only consider the constraints from Definition 10.75 concerning the probabilistic parts. Apart from $+_p$ it suffices to only prove monotonicity since $\mathcal{E}_P[\![\ ]\!]$ is a conservative extension of $\mathcal{E}[\![\ ]\!]$. In this proof let $\Pi_i = \langle \mathcal{E}_i, \pi_i \rangle$ with $\mathcal{E}_i = (E_i, \rightsquigarrow_i, \mapsto_i, l_i)$ for i=1, 2. Similarly Π'_i is defined.

1. Action-prefix. Suppose $\Pi_1 \trianglelefteq_p \Pi_2$, let $\Pi'_1 = \overline{a_\xi\,;}\,\Pi_1$ and $\Pi'_2 = \overline{a_\xi\,;}\,\Pi_2$. We infer: $\pi'_2 \restriction E_1 = \pi_2 \restriction E_1 = \pi_1 = \pi'_1$. This proves that $\overline{a_\xi\,;}$ is monotonic.

2. Choice. Suppose $\Pi_1 \trianglelefteq_p \Pi_2$, let $\Pi'_1 = \Pi_1 \overline{+} \Pi$ and $\Pi'_2 = \Pi_2 \overline{+} \Pi$. We infer:

$$
\begin{array}{ll}
 & \pi'_2 \restriction E'_1 \\
= & \{\ \text{definition } \mathcal{E}_P[\![\ ]\!]\ \} \\
 & (\pi_2 \cup \pi) \restriction (E_1 \cup E) \\
= & \{\ \text{calculus}\ \} \\
 & \pi_2 \restriction (E_1 \cup E) \cup \pi \restriction (E_1 \cup E) \\
= & \{\ E \cap E_i = \varnothing \text{ for } i\text{=}1,2\ \} \\
 & \pi_2 \restriction E_1 \cup \pi \\
= & \{\ \Pi_1 \trianglelefteq_p \Pi_2\ \} \\
 & \pi_1 \cup \pi \\
= & \{\ \text{definition } \mathcal{E}_P[\![\ ]\!]\ \} \\
 & \pi'_1\ \ .
\end{array}
$$

3. Probabilistic choice. Suppose $\Pi_1 \trianglelefteq_p \Pi_2$, let $\Pi'_1 = \Pi_1 \overline{+_p} \Pi$ and $\Pi'_2 = \Pi_2 \overline{+_p} \Pi$. Since the causality-based semantics of $+_p$ is equal to that of $+$ (cf. Chapter 9), except for the treatment of π, we only have to consider the probabilistic part. So we check:

$$
\begin{array}{ll}
 & \forall e \in dom(\pi'_1) : \pi'_2(e) = \pi'_1(e) \\
\Leftrightarrow & \{\ \text{definition } \mathcal{E}_P[\![\ ]\!]\ \} \\
 & \forall e \in dom(\pi_1) \cup init(\Pi_1) \cup dom(\pi) \cup init(\Pi) : \pi'_2(e) = \pi'_1(e) \\
\Leftrightarrow & \{\ \forall e \in dom(\pi) \cup init(\Pi) : \pi'_2(e) = \pi'_1(e) \text{ (cf. definition } \mathcal{E}_P[\![\ ]\!])\ \} \\
 & \forall e \in dom(\pi_1) \cup init(\Pi_1) : \pi'_2(e) = \pi'_1(e)
\end{array}
$$

$$\begin{array}{ll}
\Leftrightarrow & \{\ A \cup B = A \setminus B \cup B \setminus A \cup (A \cap B)\ \} \\
& \forall\, e \in \mathit{dom}(\pi_1) \setminus \mathit{init}(\Pi_1) \cup \mathit{init}(\Pi_1) \setminus \mathit{dom}(\pi_1) \cup (\mathit{dom}(\pi_1) \cap \mathit{init}(\Pi_1)) : \pi_2'(e) = \pi_1'(e) \\
\Leftrightarrow & \{\ \text{definition of } \mathcal{E}_P[\![\ ]\!]\ \} \\
& \quad (\forall\, e \in \mathit{dom}(\pi_1) \setminus \mathit{init}(\Pi_1) : \pi_2'(e) = \pi_1(e)) \\
& \wedge\ (\forall\, e \in \mathit{init}(\Pi_1) \setminus \mathit{dom}(\pi_1) : \pi_2'(e) = p) \\
& \wedge\ (\forall\, e \in \mathit{dom}(\pi_1) \cap \mathit{init}(\Pi_1) : \pi_2'(e) = p \cdot \pi_1(e)) \\
\Leftarrow & \{\ \text{Lemma 10.7; } \mathit{dom}(\pi_2) \cap E_1 = \mathit{dom}(\pi_1)\text{; definition of } \mathcal{E}_P[\![\ ]\!]\ \} \\
& \quad (\forall\, e \in \mathit{dom}(\pi_1) \setminus \mathit{init}(\Pi_1) : \pi_2(e) = \pi_1(e)) \\
& \wedge\ (\forall\, e \in \mathit{init}(\Pi_1) \setminus \mathit{dom}(\pi_1) : p = p) \\
& \wedge\ (\forall\, e \in \mathit{dom}(\pi_1) \cap \mathit{init}(\Pi_1) : p \cdot \pi_2(e) = p \cdot \pi_1(e)) \\
\Leftrightarrow & \{\ A \cup B = A \setminus B \cup B \setminus A \cup (A \cap B)\ \} \\
& \forall\, e \in \mathit{dom}(\pi_1) : \pi_2(e) = \pi_1(e) \\
\Leftarrow & \{\ \Pi_1 \trianglelefteq_p \Pi_2\ \} \\
& \text{true} \quad .
\end{array}$$

The proof of monotonicity in the second argument (that is, $\Pi_1 \trianglelefteq_p \Pi_2 \ \Rightarrow\ \Pi\overline{+_p}\Pi_1 \trianglelefteq_p \Pi\overline{+_p}\Pi_2$) is obtained by reversing the arguments in the above proof. In addition we have

$$E(\bigsqcup_i \Pi_i\overline{+_p}\Pi) = E(\bigsqcup_i \Pi_i\overline{+}\Pi) = E(\bigsqcup_i(\Pi_i\overline{+}\Pi)) = E(\bigsqcup_i(\Pi_i\overline{+_p}\Pi)).$$

This proves that $\overline{+_p}$ is continuous on events.

4. Parallel composition. Suppose $\Pi_1 \trianglelefteq_p \Pi_2$, let $\Pi_1' = \Pi_1\overline{||_G}\Pi$ and $\Pi_2' = \Pi_2\overline{||_G}\Pi$. The proof that $\Pi_1' \trianglelefteq_p \Pi_2'$ is as follows:

$$\begin{array}{ll}
& \forall\, e \in \mathit{dom}(\pi_1') : \pi_2'(e) = \pi_1'(e) \\
\Leftrightarrow & \{\ \text{definition of } \mathcal{E}_P[\![\ ]\!]\ \} \\
& \forall\, e \in (\mathit{dom}(\pi_1) \times \{\,*\,\}) \cup (\{\,*\,\} \times \mathit{dom}(\pi)) : \pi_2'(e) = \pi_1'(e) \\
\Leftrightarrow & \{\ \ \} \\
& \quad (\forall\, e \in \mathit{dom}(\pi_1) : \pi_2'((e,*)) = \pi_1'((e,*))) \\
& \wedge\ (\forall\, e \in \mathit{dom}(\pi) : \pi_2'((*,e)) = \pi_1'((*,e))) \\
\Leftrightarrow & \{\ \text{definition of } \mathcal{E}_P[\![\ ]\!]\ \} \\
& (\forall\, e \in \mathit{dom}(\pi_1) : \pi_2(e) = \pi_1(e))\ \wedge\ (\forall\, e \in \mathit{dom}(\pi) : \pi(e) = \pi(e)) \\
\Leftrightarrow & \{\ \Pi_1 \trianglelefteq_p \Pi_2\ \} \\
& \text{true} \quad .
\end{array}$$

This proves that $\overline{||_G}$ is monotonic in the first argument; like for $+_p$ the proof for monotonicity in the second argument can be obtained by reversing the arguments in the above proof.

□

Recall that the syntax of PA_P is defined using the predicated pc, ppc, and ppa. For $P := B$ we extend the definitions of these predicates as follows: $\mathsf{pc}(P) \triangleq \mathsf{pc}(B)$, $\mathsf{ppc}(P) \triangleq \mathsf{ppc}(B)$ and $\mathsf{ppa}(P) \triangleq \mathsf{ppa}(B)$. In order to have these predicates well-defined we require $P := B$ to be weakly guarded, that is, B should become guarded by substituting for a finite number of times the bodies of processes for the process instantiations occurring in B. The event structure semantics of $P := B$ is now defined as the l.u.b. of the sequence $\perp_p, \mathcal{P}_B(\perp_p), \mathcal{P}_B(\mathcal{P}_B(\perp_p)), \ldots$, where $\mathcal{P}_B$ is the probabilistic variant of $\mathcal{F}_B$.

10.84. DEFINITION. For $P := B$ a process definition let $\mathcal{E}_P[\![P]\!] \triangleq \bigsqcup_i \mathcal{P}^i_B(\perp_p)$. □

10.85. THEOREM. $\forall P \in \mathsf{PA}_P : L(\mathcal{E}_P[\![P]\!]) = L(\mathcal{E}[\![\Phi_P(P)]\!])$.

PROOF. Straightforward and omitted. □

10.7.3 Event-based operational semantics

This section extends the event-based operational semantics of PA_P of Chapter 9 with recursion. We take the same approach as in Section 10.3.3. So, each process instantiation of P is uniquely identified, as well as all occurrences of action-prefix and $\surd$. The additional inference rules are presented in Table 10.3.

$$\frac{B \xrightarrow{(\xi,a)} B'}{P_\phi \xrightarrow{(\phi\xi,a)} \phi(B')} \quad (P := B) \qquad \frac{B \xrightarrow{(\xi,a)} B'}{\phi(B) \xrightarrow{(\phi\xi,a)} \phi(B')}$$

$$\frac{B \xRightarrow{(\xi,\tau,p)} B'}{P_\phi \xRightarrow{(\phi\xi,\tau,p)} \phi(B')} \quad (P := B) \qquad \frac{B \xRightarrow{(\xi,\tau,p)} B'}{\phi(B) \xRightarrow{(\phi\xi,\tau,p)} \phi(B')}$$

Table 10.3: Additional transition rules for PA_P.

In the same way as in Section 10.3.3 it can be proven that for $P := B$ the set of event traces generated by the operational semantics coincides with the set of event traces from the denotational semantics. We will not further elaborate on this here.

10.8 Conclusions

In this chapter we have proposed a denotational semantics for recursively defined processes. This was done by applying standard fixed point theory. For each type of event structure defined in Chapters 4 through 9 of this thesis a pointed c.p.o. (or: domain) was defined and a characterization of the least upper bound of a chain under this order was provided. Except for the urgent and real-time event structures the ordering was shown to correspond to an intuitive semantical notion, viz. trace set inclusion. Besides, for each case it was shown that continuity w.r.t. the ordering boils down to continuity on events; a notion which is—as shown by Winskel [155]—technically more convenient to handle.

All operators in the process algebras PA, ..., PA_P were shown to be continuous w.r.t. the appropriate ordering. This enabled us to define the denotational semantics of $P := B$ as the least fixed point of a function on event structures. For all cases (except the urgent and real-time case) it was shown that this semantics is a conservative extension of the denotational semantics of recursive process definitions in PA—when eliminating the time, stochastic, or probabilistic information from the lposets of the event structure at hand we obtain the lposets of the event structure that are obtained by eliminating the quantitative information from the event structure at hand.

For the extended process algebras PA_T, PA_R, PA_U and PA_P we provided an event-based operational semantics for the derivation of timed (or probabilistic) event transitions of recursively defined processes. For all these cases we have shown that this operational semantics is consistent with the denotational fixed point semantics in the sense that identical sets of timed event traces are generated.

We defined the meaning of a recursive process definition by defining a pointed c.p.o. and by taking the limits of the meaning of its approximants. For event structure models this approach is quite common, see Winskel [155], Langerak [89] and Degano *et al.* [42]. An alternative approach is taken by, for instance, Loogen & Goltz [95] and Baier & Majster-Cederbaum [10] by defining a *complete metric space* on event structures. The relationship between the use of pointed c.p.o.'s and complete metric spaces in the context of event structures has been addressed by Baier & Majster-Cederbaum [11]. For all cases we used the structure of the event structure as a means to define a pointed c.p.o.; for the interval event structures of Murphy [108], a timed variant of event structures, the structure of time is used instead to define a pointed c.p.o..

11 Conclusion

This chapter contains a retrospective view on the work presented in this dissertation, summarizes the main technical results and provides some overall conclusions. In addition, some thoughts on future work are presented.

11.1 Introduction

This dissertation concerns extensions of (a variant of) event structures, a partial-order model for concurrent systems. The original incentives of our work were to study the expressiveness of event structures to effectively support the specification of distributed systems and to facilitate the formal representation of performance and reliability aspects in these models. A secondary aim was to (formally) relate the quantitative extensions of event structures to interleaving models for concurrency such that partial-order and interleaving models can be used coherently in the system design process and can be compared in a perspicuous way.

To achieve this we have widened in several ways the notion of extended bundle event structures, a model developed by Langerak [89] for providing a noninterleaving semantics to the standardized process algebra LOTOS. Basically these event structures consist of labelled events modelling occurrences of actions, a bundle relation indicating the causal dependencies among events, and an (asymmetric) conflict relation modelling the branching structure of events. The bundle relation relates a set of events, the bundle set, to an event. Bundles have to satisfy the stability constraint that requires events in a bundle set to be mutually in conflict such that only one event in a bundle set can happen.

11.2 Originality

This dissertation introduced *dual event structures*, a model obtained from extended bundle event structures by dropping the stability constraint, and several quantitative extensions of extended bundle event structures that treat real-time (both of a deterministic and stochastic nature), urgency, and probability.

Dual event structures support the specification of *disjunctive causality*, a type of causality that has received only scant attention in the literature. Rensink's [126, 127] families of labelled partial orders (lposets) were used as an underlying semantical model for dual event structures. Other models that support disjunctive causality among events are the event automata of Pinna & Poigné [118], $\{\mathsf{AND}, \mathsf{OR}\}$ automata of Gunawardena [60], and local event structures of Hoogers *et al.* [75, 76]. These models are all based on a kind of event automaton, where states keep information about the events that have happened so far, and transitions correspond to occurrences of events. Neither of these models, however, keeps track of the causal dependencies between events. Recently, Pinna & Poigné equipped their event automata with a means to mimic causal dependencies [117], but they do not address the problem of how to deduce causal dependencies in case of disjunctive causality as we did in this dissertation.

Although quantitative extensions of interleaving models have been (and still are) in vogue, noninterleaving models have been scarcely enriched with notions like time and probability. This dissertation addressed a series of such extensions of extended bundle event structures. A few partial-order models are known to us that are equipped with real-time; extensions with urgent and non-urgent events, probabilities, or time constraints defined by distribution functions, as treated in this dissertation, are unknown to us. Our real-time model, referred to as *real-time event structures*, associates a set of time instants to events, modelling absolute time constraints, and to bundles, modelling relative time constraints between causally dependent events. This model resembles the real-time extension of causal trees by Fidge [47], although he only associates time to events, does not incorporate a timeout and watchdog operator, and bases his approach on a linear-time model. Other work in this direction has been reported by Casley *et al.* [32], Maggiolo-Schettini & Winkowski [99], Murphy [106, 108], Gunawardena [61, 62], and Janssen *et al.* [78]. A more detailed description of these approaches and their relation with real-time event structures is given in Chapter 7.

11.3 Main technical achievements

This dissertation proposed a series of novel types of event structures: dual, timed, real-time, urgent, stochastic, and probabilistic event structures. Except for dual event structures that are more expressive than currently available process algebras, we considered the appropriateness of all these models to provide a noninterleaving semantics for quantitative extensions of a process algebra PA akin to LOTOS. For each variant of PA we could obtain a *denotational semantics* using the appropriate type of event structures, while retaining the noninterleaving semantics of PA to a maximal extent. A corresponding event-based *operational semantics* for most process algebras was given. This operational semantics keeps track of the occurrence of actions, rather than the actions themselves (as usual).

Below we list for each type of event structure (and related process algebra) the main technical achievements.

Dual event structures

- Characterization of lposets both in an intensional way, i.e., using the structure of the dual event structure at hand, and in an operational way, i.e., starting from event traces (but without equipping them with causality information). As an interesting result these characterizations do not coincide like for extended bundle event structures.

- Event traces are not sufficiently expressive as an underlying semantical model for dual event structures.

- Dual event structures are (on the level of lposets) strictly more expressive than Winskel's stable event structures [153, 154], and as a result, do not respect a fixed cause-and-effect relation between events.

Urgent event structures

- Due to the global impact of urgency (roughly speaking, timeouts), event traces are required to be time-consistent.
- The denotational semantics of PA_U, the urgent timed variant of PA, is not a conservative extension of the semantics of PA, since urgent events may prevent (conflicting) events to occur.
- The corresponding event-based operational semantics of PA_U, based on a separation between action- and time-transitions, closely resembles a proposal of Bolognesi *et al.* [19].

Real-time event structures

- Appropriate to provide a novel noninterleaving semantics to a real-time process algebra that includes *timeout* and *watchdog* operators.
- Absence of any mechanism to explicitly force the passage of time; time is included as a parameter in extended bundle event structures.
- Restrict the global impact of urgency such that event traces of a real-time event structure do respect causality, but not necessarily time. For each *ill-timed* trace, however, there is a corresponding time-consistent trace with the same timed events.
- The event-based operational semantics of PA_R, PA with time, timeout and watchdog operator, is a minimal and (in our opinion) elegant extension of the standard (interleaving) operational semantics of PA, and is strong bisimulation equivalent with the (noninterleaving) denotational semantics of PA_R.

Stochastic event structures

- When time constraints are determined by exponential distributions it suffices to associate rates—a rate uniquely defines an exponential distribution—only to events; the resulting model is well-suited to provide a noninterleaving semantics to a stochastic process algebra. The corresponding operational semantics coincides with several proposals from the literature, if rates are combined in the appropriate way at synchronization.
- Non-memoryless distributions can be supported if the class of distribution functions at hand is closed under product and has a unit element for this operation. Phase-type distributions fit well these requirements and are useful from a practical perspective.

Probabilistic event structures

- Probabilistic behaviour can be represented by decorating events with probabilities.
- The denotational semantics of PA_P, PA + an internal probabilistic choice operator, is a conservative extension of the denotational semantics of PA.
- The event-based operational semantics of PA_P is testing-equivalent with the denotational semantics using probabilistic event structures.

11.4 Epilogue and further work

We conclude this section by comparing the achievements of this dissertation with quantitative extensions of labelled transition systems, one of the most prominent interleaving models, in the literature. We believe that this dissertation has proven that most quantitative extensions of event structures are intuitively appealing and conceptually simpler than their interleaved counterparts. In the real-time model we benefit from the absence of actions that explicitly force time to pass; in the probabilistic model we do not have to distinguish between probabilistic and nonprobabilistic transitions (or the like) and simply attach probabilities to events; and, in the stochastic model we can exploit the notion of causal independence such that non-memoryless distribution functions can be incorporated, a problem that has not (yet) been solved satisfactory in labelled transition systems. We admit, however, that in the urgent model the advantages of event structures diminish due to the global impact of urgent events. The fact that we are 'forced' in this framework to work in a time-consistent manner, in particular that all urgent events (including causally independent ones) must be executed in that way, thwarts one of the main benefits of event structures, i.e., the locality aspect (or, the absence of a global state).

Another interesting result of this dissertation is that most of the event-based operational semantics for the various quantitative extensions of PA are relatively simple (and conservative) extensions of the standard interleaving semantics of PA. The inference rules for the real-time extension are significantly less complex than most existing interleaving proposals, while in the probabilistic case the rules simplify those of Hansson & Jonsson [65]. For the urgent case we do not 'gain' something compared to the interleaving case; the rules for this case are almost identical to those of Bolognesi *et al.* [19].

To our opinion these results justify a further exploration of the models introduced in this dissertation in order to make them suitable to effectively support the design and performance analysis of concurrent systems. Some topics that need to be addressed to reach these goals are the notions of equivalences (congruences) and preorders (precongruences) on event structures that reflect natural notions of transformation and implementation, the incorporation of data (like value passing), and the development of tool support (for instance, based on earlier work of Botma & Langerak [22]). In addition, the mapping of event structures to performance models in a systematic way needs to be addressed. There it would be interesting to consider performance models that are not based on global states (like Markov chains), but that are more 'truly concurrent'.

Appendix A Stochastic processes

In this appendix we briefly recall some results and definitions from basic probability theory as far as they are needed to understand the stochastic material in this thesis (mainly Chapters 8 and 9). For a more through treatment we refer to Kobayashi [87] and Kant [80]. We assume the reader to be familiar with the notion of stochastic variables.

A.1 Basic notions

A.1. DEFINITION. A *stochastic variable* U is characterised by a *distribution function* F_U such that $F_U(x) \triangleq Pr\{ U \leqslant x \}$. □

A stochastic variable is continuous if its distribution function is everywhere continuous. In this appendix we mainly deal with continuous distribution functions. Distribution functions satisfy the following properties:

1. $x < y \;\Rightarrow\; F_U(x) \leqslant F_U(y)$
2. $\lim_{x \longrightarrow -\infty} F_U(x) = 0$ and $\lim_{x \longrightarrow \infty} F_U(x) = 1$
3. $F_U(x) \geqslant 0$ for $-\infty < x < \infty$.

The first and last property are self-explanatory. The first part of the second property states that the event $U \leqslant x$ for $x \longrightarrow -\infty$ converges towards the impossible event and that the probability of this event is 0. The second part of this property states that for $x \longrightarrow \infty$ the event $U \leqslant x$ converges towards the certain event which occurs with probability 1. As $F_U(x)$ corresponds to a probability we have that $0 \leqslant F_U(x) \leqslant 1$ for all x.

A.2. DEFINITION. Whenever it exists, the derivative of distribution function F_U is called the *probability density function* of U, and is denoted F'_U. Therefore

$$F_U(x) \triangleq \int_{-\infty}^{x} F'_U(y)\, dy \quad .$$

□

A.3. DEFINITION. The i-th *moment* (i=1, 2, ...), denoted μ_i, of stochastic variable U is defined as the expectation of U^i. That is,

$$\mu_i \triangleq E[U^i] = \int_{-\infty}^{\infty} y^i F'_U(y) dy \quad .$$

The *expectation* of U equals the first moment μ_1 and the *variance* of U equals $\mu_2 - \mu_1^2$. $\square$

In order to be able to consider combinations of stochastic variables the joint distribution is used.

A.4. DEFINITION. Let $U_1, \ldots, U_n$ $(n \geqslant 1)$ be stochastic variables where U_i has distribution F_{U_i}, and $\overline{U} = (U_1, \ldots, U_n)$. $F_{\overline{U}}$ is called a *joint distribution function* and is defined for $\overline{x} = (x_1, \ldots, x_n)$ as

$$F_{\overline{U}}(\overline{x}) \triangleq \int_{-\infty}^{x_1} \cdots \int_{-\infty}^{x_n} F'_{\overline{U}}(y_1, \ldots, y_n)\, dy_n \ldots dy_1.$$

$U_1, \ldots, U_n$ are called *statistically independent* iff

$$F_{\overline{U}}(\overline{x}) = \prod_{i=1}^{n} F_{U_i}(x_i) = \int_{-\infty}^{x_1} F'_{U_1}(y_1)\, dy_1 \cdot \ldots \cdot \int_{-\infty}^{x_n} F'_{U_n}(y_n)\, dy_n.$$

$\square$

Note that $F_{\overline{U}}(\overline{x}) = Pr\{ U_1 \leqslant x_1, \ldots, U_n \leqslant x_n \}$.

Stochastic variables can be defined as functions from other stochastic variables. For instance, if U and V are stochastic variables, then $U{+}c$, where c is some constant, $U{+}V$ and $max(U, V)$ are stochastic variables.

A.5. LEMMA. For stochastic variables U, V with $U = V + c$ for some constant c we have $F_U(x) = F_V(x{-}c)$ and $F'_U(x) = F'_V(x{-}c)$.

PROOF. $F_U(x) = Pr\{ U \leqslant x \} = Pr\{ V + c \leqslant x \} = Pr\{ V \leqslant x{-}c \} = F_V(x{-}c)$. $\square$

Basically, a *stochastic process* is a collection of stochastic variables $\{ U(t) \mid t \in \mathsf{Time} \}$ where usually $U(t)$ denotes the value, or *state*, of U at time t. (We assume the state space to be discrete.) If Time is a denumerable set then the stochastic process is called *discrete-time*, if it is continuous the stochastic process is called *continuous-time*. If the next state of a stochastic process only depends on the current state, and not on earlier states, it is called a Markov process.

A.6. DEFINITION. (*Markov process*)

A stochastic process $\{ U(t) \mid t \in \mathsf{Time} \}$ is a *Markov process* iff for any i $(i > 0)$ the distribution of $U(t_{i+1})$ only depends on $U(t_i)$. That is,

$$Pr\{ U(t_{i+1}) \leqslant x \mid U(t_1) = x_1, \ldots, U(t_n) = x_n \} = Pr\{ U(t_{i+1}) \leqslant x \mid U(t_n) = x_n \}.$$

$\square$

A similar definition can be given for the discrete-time case. In Chapter 8 we consider Markov processes that are invariant under time shifts.

A.7. DEFINITION. Markov process $\{ U(t) \mid t \in \mathsf{Time} \}$ is called *time-homogeneous* iff for any t, t' such that $t' < t$ and x, x' we have

$$Pr\{ U(t) \leqslant x \mid U(t') = x' \} = Pr\{ U(t - \Delta) \leqslant x \mid U(t' - \Delta) = x' \}.$$

□

For Markov processes the next state only depends on the current state, and not the amount of time already spent in that state. This means that the distribution function that determines the *residence time* in a state should satisfy the *memoryless* property (see also Chapter 8). As a result, state residence times are exponentially distributed in the continuous-time case, and geometrically distributed in the discrete-time case. An extension of Markov processes, referred to as *semi-Markov* processes, allows arbitrary state residence times. These processes will be further dealt with in Chapter 9.

In this thesis we only consider Markov processes with a discrete state space. Such processes are called Markov *chains*. In the sequel we consider how continuous-time and discrete-time Markov chains (CTMCs and DTMCs) are described and confine ourselves to time-homogeneous chains. We first consider the discrete case.

A.2 Discrete-time Markov chains

We start by a classification of DTMCs which is of importance when calculating performance results. The terminology used here is adopted from Kemeny & Snell [85]. An *ergodic chain* (or irreducible chain) is a chain in which it is possible to go from every state to every other state[1].

A DTMC is often represented by a transition probability matrix $\mathbf{P}$, where $\mathbf{P}(i, j)$ can be interpreted as the probability of going from state i to j in a single transition. In general, for $n > 0$, $\mathbf{P}^n(i, j)$ denotes the probability of going from state i to j in n transitions.

A.8. DEFINITION. (*Transition probability matrix*)

$\mathbf{P}$ is a *transition probability matrix* (or stochastic matrix) iff for all i, $\sum_j \mathbf{P}(i, j) = 1$ (that is, each row sums up to 1) and $0 \leqslant \mathbf{P}(i, j) \leqslant 1$, for all i, j. □

An important notion is periodicity.

A.9. DEFINITION. The period $d(i)$ of state i is: $d(i) \triangleq gcd\{ n \mid n > 0 \wedge \mathbf{P}^n(i, i) > 0 \}$, where $gcd(\varnothing) \triangleq 0$. If $d(i) > 1$, i is called *periodic*, if $d(i) = 1$, i is called *aperiodic*. □

gcd denotes the greatest common divisor of a set of positive naturals. When state i in an ergodic chain is periodic with period $d(i)$, then all states in this chain are periodic with period $d(i)$, so we can simply speak about the period d of an ergodic chain.

[1] We restrict ourselves to *finite* Markov chains. By definition, finite ergodic chains are so-called *positive recurrent* [80]. Positive recurrence means that the expected number of transitions to return to a state is smaller than ∞, and is a necessary precondition for a general Markov chain—possibly infinite—to be ergodic. Since we only consider finite chains, the notion of positive recurrence does not have to be dealt with.

A.10. DEFINITION. A *periodic chain* is an ergodic chain with a periodic state. A *regular chain* is an ergodic chain without a periodic state. □

(It should be noticed that sometimes regular chains are called ergodic, while ergodic chains are called irreducible.)

An important part of the analysis of Markov chains is the calculation of stationary distributions and so-called steady state (or limiting) distributions. Intuitively, once a system starts in a stationary distribution it remains there forever. The limiting distribution is the distribution the system will have when time $t \longrightarrow \infty$, given some initial distribution.

Let $\pi(n)$ be the distribution of a chain at the n-th step ($n \geqslant 0$). The elements of $\pi(n)$ define the probability, $\pi_j(n)$, of being in state j at the n-th step. A chain is completely characterized by its transition probability matrix $\mathbf{P}$ and its initial distribution $\pi(0)$.

A.11. DEFINITION. (*Stationary distribution*)

π is a *stationary distribution* of a chain iff: $\pi(0) = \pi \;\Rightarrow\; (\forall\, n : \pi(n) = \pi)$. □

For stationary distribution π it holds that if the system is started with π as the initial distribution, it will retain this distribution forever. Thus the system does not move and is called stationary.

A.12. DEFINITION. (*Limiting distribution*)

π is the *limiting distribution* of a chain if, for all initial distributions $\pi(0)$, we have $\pi = \lim_{n \longrightarrow \infty} \pi(n)$, provided this limit exists. □

It is a well-known fact that for regular chains a limiting and stationary distribution always exist and that these distributions are identical.

$\pi(n)$ can be calculated from $\pi(n-1)$ as follows:

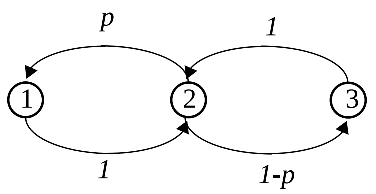
$$\pi(n) = \pi(n-1) \cdot \mathbf{P}, \text{for } n > 1.$$

This recursive equation can be rewritten into

$$\pi(n) = \pi(0) \cdot \mathbf{P}^n. \tag{A.1}$$

Thus, the limiting distribution of a chain is equal to $\lim_{n \longrightarrow \infty} \pi(0) \cdot \mathbf{P}^n$, provided this limit exists. This limit exists for regular chains, but not for periodic ones. So, a regular chain has a unique limiting distribution, but a periodic one does not. Intuitively this is clear as, although a periodic chain 'on the long run' reaches some 'stationary behaviour', it remains cycling in a fixed way. The limiting distribution of a DTMC can—if it exists—be computed by solving the following system of linear equations

$$\pi \cdot \mathbf{P} = \pi, \quad \sum_j \pi_j = 1 \quad .$$



$$\mathbf{P} = \begin{bmatrix} 0 & 1 & 0 \\ p & 0 & 1-p \\ 0 & 1 & 0 \end{bmatrix}$$

Figure A.1: Periodic discrete-time Markov chain.

A.13. EXAMPLE. Consider the periodic chain (d=2) of Figure A.1, and assume the chain is initially in state 1, that is, $\pi(0) = [1, 0, 0]$. Using equation (A.1) we get:

$$\pi(n) = \begin{cases} [0, 1, 0] & \text{if } n \text{ is odd} \\ [p, 0, 1-p] & \text{if } n \text{ is even.} \end{cases}$$

Therefore, $\lim_{n \longrightarrow \infty} \pi(n)$ does not exist for $\pi(0)$ and—by definition—the chain has no limiting distribution. However, for $\pi'(0) = [\frac{1}{2}\, p, \frac{1}{2}, \frac{1}{2}\,(1-p)]$ we get $\pi(n) = \pi'(0)$, for all n. This is a stationary distribution of the chain. So, although the periodic chain has a stationary distribution, it has no limiting distribution. □

A.3 Continuous-time Markov chains

A CTMC is determined by its (infinitesimal) generator matrix (or rate matrix) and its initial distribution.

A.14. DEFINITION. (*Generator matrix*)

> $\mathbf{Q}$ is a *generator matrix* iff, for all i, $\mathbf{Q}(i,j) \geqslant 0$ $(i \neq j)$, $\sum_j \mathbf{Q}(i,j) = 0$ (that is, each row sums up to 0), and $\mathbf{Q}(i,i) = -\sum_{j \neq i} \mathbf{Q}(i,j)$. □

For obtaining the limiting distribution π of a CTMC (which, in the continuous case, always exists) the following system of linear equations has to be solved

$$\pi \cdot \mathbf{Q} = 0, \quad \sum_j \pi_j = 1 \quad .$$

Appendix B Domain theory

In this appendix we briefly recall some results and definitions from basic domain theory as far as they are needed to understand Chapter 10. For a more thorough treatment we refer to Schmidt [132] and Gunther & Scott [63]. A more informal treatment is given in Tennent [139] and Manna *et al.* [100].

B.1. Definition. (*Partial order*)

A binary relation $\unlhd$ on set D is a *partial order* iff, for all $d, d', d'' \in D$:

1. $d \unlhd d$ (reflexivity)
2. $(d \unlhd d' \ \wedge\ d' \unlhd d) \ \Rightarrow\ d = d'$ (anti-symmetry)
3. $(d \unlhd d' \ \wedge\ d' \unlhd d'') \ \Rightarrow\ d \unlhd d''$ (transitivity).

□

The pair $\langle D, \unlhd \rangle$ is a partially ordered set, or shortly, *poset*. If $d \not\unlhd d'$ and $d' \not\unlhd d$ then d and d' are incomparable.

B.2. Definition. Let $\langle D, \unlhd \rangle$ a poset and $D' \subseteq D$.

1. $d \in D$ is an *upper bound* of D' if $\forall\, d' \in D' : d' \unlhd d$.
2. $d \in D$ is a *least* upper bound (l.u.b.) of D', denoted $\bigsqcup D'$, if d is an upper bound of D' and $(\forall\, d'' \in D : d''$ is an upper bound of $D' \ \Rightarrow\ d \unlhd d'')$.

□

B.3. Lemma. Let $\langle D, \unlhd \rangle$ a poset and $D' \subseteq D$. If D' has a l.u.b., this l.u.b. is unique.

Proof. Routine and omitted. □

B.4. Definition. Let $\langle D, \unlhd \rangle$ a poset and $D' \subseteq D$. D' is a *chain* if $D' \neq \varnothing$ and $(\forall\, d, d' \in D' : d \unlhd d' \ \vee\ d' \unlhd d)$. ($D'$ is totally ordered.) □

The l.u.b. of chain $d_1 \unlhd d_2 \unlhd \ldots$ is denoted $\bigsqcup_i D$ where $D = \{\, d_1, d_2, \ldots \}$, or simply by $\bigsqcup_i d_i$.

B.5. Definition. Let $\langle D, \unlhd \rangle$ a poset.

1. $\langle D, \unlhd \rangle$ is *complete* (c.p.o.) if each chain in D has a l.u.b..

2. $\langle D, \trianglelefteq \rangle$ is *pointed complete* if it is complete and there exists a *least element* in D, denoted $\bot$, such that $\forall\, d \in D : \bot \trianglelefteq d$.

□

(Note: different terminology in the literature exists. Sometimes a pointed c.p.o. is called a Scott domain, or simply domain, and sometimes the existence of a least element is incorporated in the definition of c.p.o.. Here, we follow Schmidt [132].)

B.6. Definition. Let $\langle D, \trianglelefteq \rangle$ and $\langle D', \trianglelefteq' \rangle$ posets and $F : D \to D'$.

1. F is *monotonic* iff $\forall\, d_1, d_2 \in D : d_1 \trianglelefteq d_2 \;\Rightarrow\; F(d_1) \trianglelefteq' F(d_2)$.
2. If D and D' are complete, then F is *continuous* iff for each chain E in D we have $F(\bigsqcup_D E) = \bigsqcup_{D'} F(E)$.

□

That is, F is continuous if and only if it preserves l.u.b.'s.

B.7. Corollary. Let $\langle D, \trianglelefteq \rangle$ and $\langle D', \trianglelefteq' \rangle$ be c.p.o.'s and $F : D \to D'$. Then:

$$F \text{ is continuous} \quad \Rightarrow \quad F \text{ is monotonic} .$$

Proof. Consider w.l.o.g. $D = \{\, d, d' \,\}$. Then we derive:

$$
\begin{array}{ll}
 & d \trianglelefteq d' \\
= & \{ \text{ Definition B.2 } \} \\
 & \bigsqcup_D \{\, d, d' \,\} = d' \\
\Rightarrow & \{ \text{ Leibniz's rule } \} \\
 & F(\bigsqcup_D \{\, d, d' \,\}) = F(d') \\
= & \{ \; F \text{ is continuous } \} \\
 & \bigsqcup_{D'} \{\, F(d), F(d') \,\} = F(d') \\
= & \{ \text{ Definition B.2 } \} \\
 & F(d) \trianglelefteq' F(d') \quad .
\end{array}
$$

□

For function F, let F^0 be the identity function, and $F^{n+1} = F \circ F^n$, for $n \geqslant 0$, where $\circ$ denotes usual function composition.

B.8. Theorem. *Kleene's first recursion theorem*

Let $\langle D, \trianglelefteq \rangle$ a pointed c.p.o. and $F : D \to D$ continuous. Then:

1. $\{\, d \in D \mid F(d) = d \,\}$ has a least element, denoted $\mathsf{fix}\, F$.
2. $\mathsf{fix}\, F$ is *unique* and $\mathsf{fix}\, F = \bigsqcup_i F^i(\bot)$, for $i \geqslant 0$.

PROOF. See, for instance, [132, Theorem 6.11]. □

$\mathsf{fix}\, F$ is called the *least fixed point* of F.

B.9. THEOREM. Let $\langle D, \trianglelefteq \rangle$, $\langle D', \trianglelefteq' \rangle$ and $\langle D'', \trianglelefteq'' \rangle$ c.p.o.'s, $F : D \to D'$ and $G : D' \to D''$ be continuous functions. Then $G \circ F$ is continuous.

PROOF. Routine and omitted. □

B.10. DEFINITION. Let $\langle D_1, \trianglelefteq_1 \rangle, \ldots, \langle D_n, \trianglelefteq_n \rangle$ pointed c.p.o.'s. Then define $\langle D, \trianglelefteq \rangle$ with $D = D_1 \times \ldots \times D_n$ and $(d_1, \ldots, d_n) \trianglelefteq (d'_1, \ldots, d'_n)$ iff $d_i \trianglelefteq_i d'_i$, for all $0 < i \leqslant n$. □

B.11. LEMMA. $\langle D, \trianglelefteq \rangle$, the product of pointed c.p.o.'s $\langle D_1, \trianglelefteq_1 \rangle, \ldots, \langle D_n, \trianglelefteq_n \rangle$, is a pointed c.p.o..

PROOF. See, for instance, [132, Proposition 6.17]. □

B.12. LEMMA. A function $F : D_1 \times \ldots \times D_n \to E$ is continuous iff it is continuous on every D_i, for $0 < i \leqslant n$.

PROOF. See, for instance, [132, Proposition 6.18]. □

Bibliography

[1] L. ACETO AND D. MURPHY. On the ill-timed but well-caused. In Best [15], pages 97–111.

[2] L. ACETO AND D. MURPHY. Timing and causality in process algebra. *Acta Informatica*, 1996. (to appear).

[3] M. AJMONE MARSAN AND A. BIANCO AND L. CIMINIERA AND R. SISTO AND A. VALENZANO. A LOTOS extension for the performance analysis of distributed systems. *IEEE/ACM Transactions on Networking*, **2**(2):151–164, 1994.

[4] M. AJMONE MARSAN AND G. CONTE AND G. BALBO. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Programming Languages and Systems*, **2**(2):93–122, 1984.

[5] R. ALUR AND D.L. DILL. A theory of timed automata. *Theoretical Computer Science*, **126**:183–235, 1994.

[6] J.C.M. BAETEN. The total order assumption. In S. Purushothaman and A. Zwarico, editors, *Proceedings First North American Process Algebra Workshop*, Workshops in Computing, pages 231–240. Springer-Verlag, 1993. (also in Proceedings of the Workshop "What good are partial orders?", E. Best (editor), Hildesheimer Informatik-Berichte 13/92, pages 1-11, 1992).

[7] J.C.M. BAETEN AND J.A. BERGSTRA. Real time process algebra. *Formal Aspects of Computing*, **3**(2):142–188, 1991.

[8] J.C.M. BAETEN AND J.A. BERGSTRA AND S.A. SMOLKA. Axiomatizing probabilistic processes: ACP with generative probabilities. *Information and Computation*, **121**:234–255, 1995. (preliminary version appeared in W.R. Cleaveland, editor, *Concur '92*, LNCS 630, pages 472–485. Springer-Verlag, 1992).

[9] J.C.M. BAETEN AND J.W. KLOP, EDITORS. *Concur '90: Theories of Concurrency – Unification and Extension*, volume 458 of *Lecture Notes in Computer Science*. Springer-Verlag, 1990.

[10] C. BAIER AND M.E. MAJSTER-CEDERBAUM. The connection between an event structure semantics and an operational semantics for TCSP. *Acta Informatica*, **31**:81–104, 1994.

[11] C. BAIER AND M.E. MAJSTER-CEDERBAUM. Denotational semantics in the cpo and metric approach. *Theoretical Computer Science*, **135**:171–220, 1994.

[12] Y. BEN-ASHER AND E. FARCHI. Using true concurrency to model execution of parallel programs. *International Journal of Parallel Programming*, **22**(4):375–407, 1994.

[13] J.A. Bergstra and J.W. Klop. Algebra of communicating processes with abstraction. *Theoretical Computer Science*, **37**(1):77–121, 1985.

[14] M. Bernardo and L. Donatiello and R. Gorrieri. Modeling and analyzing concurrent systems with MPA. In Herzog and Rettelbach [69], pages 175–189.

[15] E. Best, editor. *Concur '93: Concurrency Theory*, volume 715 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[16] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, **14**:25–59, 1987.

[17] T. Bolognesi and G. Ciaccio. Cumulating constraints on the 'when' and the 'what'. In Tenney et al. [140], pages 433–448.

[18] T. Bolognesi and F. Lucidi. Timed process algebras with urgent interactions and a unique powerful binary operator. In de Bakker et al. [40], pages 124–148.

[19] T. Bolognesi and F. Lucidi and S. Trigila. Converging towards a timed LOTOS standard. *Computer Standards & Interfaces*, **16**:87–118, 1994.

[20] T. Bolognesi and S. Schneider. Unpublished manuscript, 1994.

[21] T. Bolognesi, J. van de Lagemaat, and C.A. Vissers, editors. *LOTOSphere: Software Development with LOTOS*. Kluwer Academic Publishers, 1995.

[22] B. Botma and R. Langerak. Simulator for LOTOS to study the independence and causality of events. In D. Hogrefe and S. Leue, editors, *Formal Description Techniques VII*, Participants proceedings, pages 201–203, 1994.

[23] G. Boudol and I. Castellani. A non-interleaving semantics for CCS based on proved transitions. *Fundamenta Informaticae*, **11**(4):433–452, 1988.

[24] G. Boudol and I. Castellani. Permutations of transitions: An event structure semantics for CCS and SCCS. In de Bakker et al. [39], pages 411–427.

[25] G. Boudol and I. Castellani. Flow models of distributed computations: Event structures and nets. Rapports de Recherche 1482, INRIA, 1991.

[26] G. Boudol and I. Castellani. Flow models of distributed computations: three equivalent semantics for CCS. *Information and Computation*, **114**:247–314, 1994. (preliminary version appeared in I. Guessarian, editor, *Semantics of Systems of Concurrent Processes*, LNCS 469, pages 96–141. Springer-Verlag, 1990).

[27] E. Brinksma. Performance and formal design–a process algebraic perspective. (oral presentation at 6th Int. Workshop on Petri Nets and Performance Models), 1995.

[28] E Brinksma and J.-P. Katoen and R. Langerak and D. Latella. Performance analysis and true concurrency semantics. In T. Rus and C. Rattray, editors, *Theories and Experiences for Real-Time System Development*, volume 2 of *AMAST Series in Computing*, chapter 12, pages 309–337. World Scientific, 1994.

[29] E. Brinksma and J.-P. Katoen and R. Langerak and D. Latella. A stochastic causality-based process algebra. *The Computer Journal*, **38**(7):552–565, 1995.

[30] P. Buchholz. Markovian process algebra: Composition and equivalence. In Herzog and Rettelbach [69], pages 11–30.

[31] R.T. Casley. *On the Specification of Concurrent Systems.* PhD thesis, Stanford University, 1991.

[32] R.T. Casley and R.F. Crew and J. Meseguer and V.R. Pratt. Temporal structures. *Mathematical Structures in Computer Science*, **1**(2):179–213, 1991.

[33] C.-T. Chou. Mechanical verification of distributed algorithms in higher-order logic. *The Computer Journal*, **38**(2):152–162, 1995.

[34] I. Christoff. Testing equivalences and fully abstract models for probabilistic processes. In Baeten and Klop [9], pages 126–140.

[35] L. Christoff. *Specification and Verification Models for Probabilistic Processes.* PhD thesis, Uppsala University, 1993. (also available as Technical Report 93/37).

[36] P. Darondeau and P. Degano. Causal trees. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *Automata, Languages and Programming*, volume 372 of *Lecture Notes in Computer Science*, pages 234–248. Springer-Verlag, 1989.

[37] P. Darondeau and P. Degano. Event structures, causal trees, and refinement. In B. Rovan, editor, *Mathematical Foundations of Computer Science 1990*, volume 452 of *Lecture Notes in Computer Science*, pages 239–245. Springer-Verlag, 1990.

[38] M. Davio. Kronecker products and shuffle algebra. *IEEE Transactions on Computers*, **C-30**(2):116–125, 1981.

[39] J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors. *Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency*, volume 354 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[40] J.W. de Bakker, C. Huizing, W.-P. de Roever, and G. Rozenberg, editors. *Real-time: Theory in Practice*, volume 600 of *Lecture Notes in Computer Science*. Springer-Verlag, 1992.

[41] M.K. de Weger and H. Franken and C.A. Vissers. A development model for distributed information systems. In *Proceedings of the 1st Int. Distributed Conference on High Performance Networking for Teleteaching (IDC'95)*, 1995.

[42] P. Degano and R. De Nicola and U. Montanari. On the consistency of 'truly concurrent' operational and denotational semantics (extended abstract). In *Third Annual Symposium on Logic in Computer Science*, pages 133–141. IEEE Computer Society Press, 1988.

[43] P. Degano and R. Gorrieri and S. Vigna. On relating some models for concurrency. In M.-C. Gaudel and J.-P. Jouannaud, editors, *Theory and Practice of Software Technology*, volume 668 of *Lecture Notes in Computer Science*, pages 15–30. Springer-Verlag, 1993. (revised version as Technical Report UBLCS-93-35, University of Bologna).

[44] M. Diaz and R. Groz, editors. *Formal Description Techniques V*, volume C–10 of *IFIP Transactions*. North-Holland, 1993.

[45] M. Fang and C.J. Ho-Stuart and H.S.M. Zedan. Specification of real-time probabilistic behaviour. In A. Danthine, G. Leduc, and P. Wolper, editors, *Protocol Specification, Testing, and Verification, XIII*, volume C–16 of *IFIP Transactions*, pages 143–157. North-Holland, 1993.

[46] L. Ferreira Pires. *Architectural Notes: A Framework for Distributed Systems Development*. PhD thesis, University of Twente, 1994.

[47] C. Fidge. A constraint-oriented real-time process calculus. In Diaz and Groz [44], pages 363–378.

[48] A. Giacalone and C.-C. Jou and S.A. Smolka. Algebraic reasoning for probabilistic concurrent systems. In M. Broy and C.B. Jones, editors, *Proceedings of the Working Conference on Programming Concepts and Methods*, pages 443–458. North-Holland, 1990.

[49] R.J. van Glabbeek. The linear time – branching time spectrum. In Baeten and Klop [9], pages 278–297.

[50] R.J. van Glabbeek. The linear time – branching time spectrum II. In Best [15], pages 66 – 81.

[51] R.J. van Glabbeek. What is branching time semantics and why to use it? *Bull. Eur. Ass. Theoret. Comput. Sci.*, **53**:190–198, 1994.

[52] R.J. van Glabbeek and G.D. Plotkin. Configuration structures (extended abstract). In D. Kozen, editor, *Proceedings 10th Annual Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1995.

[53] R.J. van Glabbeek and S.A. Smolka and B. Steffen. Reactive, generative, and stratified models of probabilistic processes. *Information and Computation*, **121**:59–80, 1995. (earlier version, together with C. Tofts, in *Proceedings 5th Annual IEEE Symposium on Logic in Computer Science*, pages 130–141, IEEE Computer Society Press, 1990).

[54] R.J. van Glabbeek and F.W. Vaandrager. Petri Net models for algebraic theories of concurrency. In J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, editors, *PARLE — Parallel Architectures and Languages Europe*, volume 259 of *Lecture Notes in Computer Science*, pages 224–242. Springer-Verlag, 1987.

[55] P. Godefroid. *Partial-Order Models for the Verification of Concurrent Systems–An Approach to the State-Explosion Problem.* PhD thesis, Université de Liege, 1994. (a revised version appeared as volume 1032 of *Lecture Notes in Computer Science*, 1996).

[56] R. Gorrieri and M. Roccetti and E. Stancampiano. A theory of processes with durational actions. *Theoretical Computer Science*, **140**:73–94, 1995.

[57] N. Götz. *Stochastische Prozessalgebren–Integration von Funktionalem Entwurf und Leistungsbewertung Verteilter Systeme.* PhD thesis, Universität Erlangen-Nürnberg, 1994. (in German).

[58] N. Götz and U. Herzog and M. Rettelbach. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In L. Donatiello and R. Nelson, editors, *Performance Evaluation of Computer and Communication Systems*, volume 729 of *Lecture Notes in Computer Science*, pages 121–146. Springer-Verlag, 1993.

[59] N. Götz and U. Herzog and M. Rettelbach. TIPP – introduction and application to protocol performance analysis. In H. König, editor, *Formale Beschreibungstechniken für verteilte Systeme*, FOKUS series. Saur Verlag, 1993.

[60] J. Gunawardena. Causal automata. *Theoretical Computer Science*, **101**:265–288, 1992.

[61] J. Gunawardena. Periodic behaviour in timed systems with $\{\mathsf{AND}, \mathsf{OR}\}$ causality — part I: Systems of dimensions 1 and 2. Technical Report STAN-CS-93-1462, Stanford University, 1993.

[62] J. Gunawardena. A dynamic approach to timed behaviour. In B. Jonsson and J. Parrow, editors, *Concur' 94: Concurrency Theory*, volume 836 of *Lecture Notes in Computer Science*, pages 178–193. Springer-Verlag, 1994.

[63] C.A. Gunther and D.A. Scott. Semantic domains. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science (Vol. B: Formal Models)*, chapter 12, pages 633–674. Elsevier Science Publishers B.V., 1990.

[64] H. Hansson. *Time and Probability in Formal Design of Distributed Systems.* PhD thesis, Uppsala University, 1991. (revised version appeared in the series *Real-Time Safety Critical Systems*, vol. 1, Elsevier, 1994).

[65] H. Hansson and B. Jonsson. A calculus for communicating systems with time and probabilities. In *Proceedings of 11th IEEE Real-Time Systems Symposium*, pages 278–287. IEEE Computer Society Press, 1990.

[66] C. Harvey. Performance engineering as an integral part of system design. *British Telecom Technology Journal*, **4**:142–147, 1986.

[67] M. Hennessy and T. Regan. A temporal process algebra. *Information and Computation*, **117**:221–239, 1995.

[68] H. Herrmanns and M. Rettelbach. Syntax, semantics, equivalences, and axioms for MTIPP. In Herzog and Rettelbach [69], pages 71–87.

[69] U. Herzog and M. Rettelbach, editors. *Proceedings of the 2nd Workshop on Process Algebras and Performance Modelling*, Erlangen, 1994. Universität Erlangen-Nürnberg.

[70] D.P. Heyman and M.J. Sobel. *Stochastic Models in Operations Research*, volume 1 - Stochastic Processes and Operating Characteristics. McGraw-Hill, New York, 1982.

[71] J. Hillston. PEPA: Performance Enhanced Process Algebra. Technical Report CSR-24-93, University of Edinburgh, 1993.

[72] J. Hillston. *A Compositional Approach to Performance Modelling.* PhD thesis, University of Edinburgh, 1994. (also available as Technical Report CST-107-94).

[73] J. Hillston. The nature of synchronisation. In Herzog and Rettelbach [69], pages 51–70.

[74] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[75] P.W. Hoogers. *Behavioural Aspects of Petri Nets.* PhD thesis, Leiden University, 1994.

[76] P.W. Hoogers and H.C.M. Kleijn and P.S. Thiagarajan. An event structure semantics for general Petri nets. *Theoretical Computer Science*, **153**(1/2):129–170, 1996. (preliminary version appeared in E. Best, editor, *Concur '93*, LNCS 715, pages 462–476. Springer-Verlag, 1993).

[77] W. Janssen. *Layered Design of Parallel Systems.* PhD thesis, University of Twente, 1994.

[78] W. Janssen and M. Poel and Q. Wu and J. Zwiers. Layering of real-time distributed processes. In Langmaack et al. [92], pages 393–417.

[79] A.S.A. Jeffrey and S. Schneider and F.W. Vaandrager. A comparison of additivity axioms in timed transition systems. Technical Report CS-R9366, Centre for Mathematics and Computer Science, 1993.

[80] K. Kant. *Introduction to Computer System Performance Evaluation.* Computer Science Series. McGraw-Hill, Inc., 1992.

[81] J.-P. Katoen. Causal behaviours and nets. In G. de Michelis and M. Diaz, editors, *Application and Theory of Petri Nets 1995*, volume 935 of *Lecture Notes in Computer Science*, pages 258–277. Springer-Verlag, 1995.

[82] J.-P. Katoen and R. Langerak and D. Latella. Modelling systems by probabilistic process algebra: An event structures approach. In Tenney et al. [140], pages 253–268.

[83] J.-P. Katoen and D. Latella and R. Langerak and E. Brinksma and T. Bolognesi. A consistent causality-based view on a timed process algebra. In A. Cornell and D. Ionescu, editors, *Proceedings 3rd Amast Workshop on Real-Time System Development*, 1996.

[84] R.M. Keller. Formal verification of parallel programs. *Communications of the ACM*, **19**(7):371–384, 1976.

[85] J.G. Kemeny and J.L. Snell. *Finite Markov Chains.* Van Nostrand, 1960.

[86] A.S. Klusener. *Models and axioms for a fragment of real time process algebra.* PhD thesis, Eindhoven University of Technology, 1993.

[87] H. Kobayashi. *Modeling and Analysis: An Introduction to System Performance Evaluation Methodology.* Addison-Wesley, 1978.

[88] L. Lamport. On interprocess communication, part I: Basic formalism. *Distributed Computing*, **1**:77–85, 1986.

[89] R. Langerak. *Transformations and Semantics for LOTOS.* PhD thesis, University of Twente, 1992.

[90] R. Langerak. Bundle event structures: a non-interleaving semantics for LOTOS. In Diaz and Groz [44], pages 331–346.

[91] R. Langerak and D. Latella. A language of finite probabilistic processes and its interleaving semantics. Memoranda Informatica 93-24, University of Twente, 1993.

[92] H. Langmaack, W.-P. de Roever, and J. Vytopil, editors. *Formal Techniques in Real-Time and Fault-Tolerant Systems*, volume 863 of *Lecture Notes in Computer Science.* Springer-Verlag, 1994.

[93] D. Latella. Recursive bundle event structures. Memoranda Informatica 93–27, University of Twente, 1993.

[94] L. Logrippo and M. Faci and M. Haj-Hussein. An introduction to LOTOS: learning by examples. *Computer Networks and ISDN Systems*, **23**:325–342, 1992.

[95] R. Loogen and U. Goltz. Modelling nondeterministic concurrent processes with event structures. *Fundamenta Informaticae*, **14**:39–74, 1991.

[96] G. Lowe. Representing nondeterminism and probabilistic behaviour in reactive processes. Technical Report PRG-TR-12-93, Oxford University Computing Laboratory, 1993.

[97] G. Lowe. Probabilistic and prioritized models of timed CSP. *Theoretical Computer Science*, **138**:315–352, 1995.

[98] N.A. Lynch and F.W. Vaandrager. Action transducers and timed automata. *Formal Aspects of Computing*, 1996. (preliminary version appeared in W.R. Cleaveland, editor, *Concur'92*, LNCS 630, pages 436–455. Springer-Verlag, 1992).

[99] A. Maggiolo-Schettini and J. Winkowski. Towards an algebra for timed behaviours. *Theoretical Computer Science*, **103**:335–363, 1992.

[100] Z. Manna and S. Ness and J. Vuillemin. Inductive methods for proving properties of programs. *Communications of the ACM*, **16**(8):491–502, 1973.

[101] A. Mazurkiewicz. Basic notions of trace theory. In de Bakker et al. [39], pages 285–363.

[102] C. Miguel and A. Fernández and L. Vidaller. LOTOS extended with probabilistic behaviours. *Formal Aspects of Computing*, **5**:253–281, 1993.

[103] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[104] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[105] F. Moller and C. Tofts. A temporal calculus of communicating systems. In Baeten and Klop [9].

[106] D.V.J. Murphy. *Time, Causality, and Concurrency*. PhD thesis, University of Surrey, 1990. (also available as Technical Report CSC 90/R32, University of Glasgow).

[107] D.V.J. Murphy. Timed process algebra, Petri nets, and event refinement. In J.M. Morris and R.C. Shaw, editors, *4th Refinement Workshop*, Workshops in Computing, pages 456–478. Springer-Verlag, 1991.

[108] D.V.J. Murphy. Time and duration in noninterleaving concurrency. *Fundamenta Informaticae*, **19**:403–416, 1993.

[109] M.F. Neuts. *Matrix-geometric Solutions in Stochastic Models–An Algorithmic Approach*. The Johns Hopkins University Press, 1981.

[110] M.F. Neuts. *Structured Stochastic Matrices of M/G/1 Type and Their Applications*. Marcel Dekker, Inc., 1989.

[111] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, **34**:83–133, 1984.

[112] X. Nicollin and J. Sifakis. An overview and synthesis on timed process algebras. In de Bakker et al. [40], pages 526–548.

[113] X. Nicollin and J. Sifakis and S. Yovine. From ATP to timed graphs and hybrid systems. In de Bakker et al. [40], pages 549–572.

[114] M. Nielsen and G.D. Plotkin and G. Winskel. Petri nets, event structures and domains, part 1. *Theoretical Computer Science*, **13**(1):85–108, 1981.

[115] M. Núñez and D. de Frutos. Testing semantics for probabilistic LOTOS. In G. von Bochmann, R. Dssouli, and O. Rafiq, editors, *Formal Description Techniques VIII*, pages 365–380, 1995.

[116] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Proceedings 5th GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[117] G.M. Pinna and A. Poigné. The mathematics of event automata. In *Proceedings Int. Conf. on Category Theory and Computer Science*, 1993.

[118] G.M. Pinna and A. Poigné. On the nature of events: another perspective in concurrency. *Theoretical Computer Science*, **138**(2):425–454, 1995. (preliminary version appeared in I.H. Havel and V. Koubek, editors, *Mathematical Foundations of Computer Science' 92*, LNCS 629, pages 430–441. Springer-Verlag, 1992).

[119] B. Plateau and J.-M. Fourneau. A methodology for solving Markov models of parallel systems. *Journal of Parallel and Distributed Computing*, **12**:370–387, 1991.

[120] G.D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[121] V.R. Pratt. Modeling concurrency with partial orders. *International Journal of Parallel Programming*, **15**(1):33–71, 1986.

[122] S. Purushothaman and P.A. Subrahmanyam. Reasoning about probabilistic behaviour in concurrent systems. *IEEE Transactions on Software Engineering*, **SE-13**(6):740–745, 1987.

[123] J. Quemada and D. de Frutos and A. Azcorra. TIC: A TImed Calculus. *Formal Aspects of Computing*, **5**:224–252, 1993.

[124] G.M. Reed and A.W. Roscoe. A timed model for Communicating Sequential Processes. *Theoretical Computer Science*, **58**:249–261, 1988. (preliminary version appeared in L. Kott, editor, *Proceedings 13th Int. Colloquium on Automata, Languages and Programming (ICALP)*, LNCS 226, pages 314–323. Springer-Verlag, 1986).

[125] W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.

[126] A. Rensink. Posets for configurations! In W.R. Cleaveland, editor, *Concur '92*, volume 630 of *Lecture Notes in Computer Science*, pages 269–285. Springer-Verlag, 1992.

[127] A. Rensink. *Models and Methods for Action Refinement*. PhD thesis, University of Twente, 1993.

[128] M. Rettelbach. *Stochastische Prozessalgebren mit zeitlosen Aktivitäten und probabilistischen Verzweigungen.* PhD thesis, Universität Erlangen-Nürnberg, 1996. (in German).

[129] N. Rico and G. von Bochmann. Performance description and analysis for distributed systems using a variant of LOTOS. In B. Jonsson et. al., editor, *Protocol Specification, Testing, and Verification IX*, pages 199–213. North-Holland, 1991.

[130] S.M. Ross. *Stochastic Processes.* John Wiley & Sons, New York, 1983.

[131] R.A. Sahner and K.S. Trivedi. Performance and reliability analysis using directed acyclic graphs. *IEEE Transactions on Software Engineering*, **SE-13**(10):1105–1114, 1987.

[132] D.A. Schmidt. *Denotational Semantics: a methodology for language development.* Allyn and Bacon, 1986.

[133] S. Schneider. An operational semantics for timed CSP. *Information and Computation*, **116**:193–213, 1995.

[134] R. Schwarz and F. Mattern. Detecting causal relationships in distributed computations: in search of the holy grail. *Distributed Computing*, **7**:149–174, 1994.

[135] K. Seidel. Probabilistic communicating processes. *Theoretical Computer Science*, **152**:219–249, 1995.

[136] R. Sharp. *Principles of Protocol Design.* Prentice-Hall, 1994.

[137] M.W. Shields. Concurrent machines. *The Computer Journal*, **28**(5):449–465, 1985.

[138] R. Sisto and L. Ciminiera and A. Valenzano. Probabilistic characterization of algebraic protocol specifications. In *Proceedings 12th Int. Conf. on Distributed Computing Systems*, pages 260–268. IEEE Computer Society Press, 1992.

[139] R.D. Tennent. The denotational semantics of programming languages. *Communications of the ACM*, **19**:437–453, 1976.

[140] R.L. Tenney, P.D. Amer, and M.Ü. Uyar, editors. *Formal Description Techniques VI*, volume C–22 of *IFIP Transactions.* North-Holland, 1994.

[141] C.M.N. Tofts. A synchronous calculus of relative frequency. In Baeten and Klop [9], pages 467–480.

[142] J. Tretmans. *A Formal Approach to Conformance Testing.* PhD thesis, University of Twente, 1992.

[143] K.S. Trivedi and A. Bobbio and G. Ciardo and R. German and A. Puliafito and M. Telek. Non-Markovian Petri nets. *Performance Evaluation Review*, **23**:263–264, 1995.

[144] F.W. VAANDRAGER. A simple definition for parallel composition of prime event structures. Report CS-R8903, Centre for Mathematics and Computer Science, 1989.

[145] M. VAN SINDEREN AND L. FERREIRA PIRES AND C.A. VISSERS AND J.-P. KATOEN. A design model for open distributed processing systems. *Computer Networks and ISDN Systems*, **27**:1263–1285, 1995.

[146] T. VERHOEFF. *A Theory of Delay-Insensitive Circuits*. PhD thesis, Eindhoven University of Technology, 1994.

[147] C.A. VISSERS. FDTs for open distributed systems, a retrospective and a prospective view. In L. Logrippo, R.L. Probert, and H. Ural, editors, *Protocol Specification, Testing and Verification X*, pages 341–362. North-Holland, 1990.

[148] C.A. VISSERS AND G. SCOLLO AND M. VAN SINDEREN AND E. BRINKSMA. On the use of specification styles in the design of distributed systems. *Theoretical Computer Science*, **89**(1):179–206, 1991.

[149] Y. WANG. Real-time behaviour of asynchronous agents. In Baeten and Klop [9], pages 502–520.

[150] Y. WANG. Algebraic reasoning for real-time probabilistic processes with uncertain information. In Langmaack et al. [92], pages 680–693.

[151] H. WEHRHEIM. Parametric action refinement. In E.-R. Olderog, editor, *Programming Concepts, Methods, and Calculi*, volume A–56 of *IFIP Transactions*, pages 247–266. North-Holland, 1994.

[152] G. WINSKEL. *Events in Computation*. PhD thesis, University of Edinburgh, 1980. (also available as Technical Report CST-10-80).

[153] G. WINSKEL. Event structure semantics for CCS and related languages. In M. Nielsen and E.M. Schmidt, editors, *Automata, Languages and Programming*, volume 140 of *Lecture Notes in Computer Science*, pages 561–576. Springer-Verlag, 1982.

[154] G. WINSKEL. Event structures. In W. Brauer, W. Reisig, and G. Rozenberg, editors, *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer-Verlag, 1987.

[155] G. WINSKEL. An introduction to event structures. In de Bakker et al. [39], pages 364–397.

[156] G. WINSKEL AND M. NIELSEN. Models for concurrency. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4: Semantic Modelling, pages 2–148. Oxford University Press, 1995.

[157] A. YAKOVLEV AND M. KISHINEVSKY AND A. KONDRATYEV AND L. LAVAGNO. On the models for asynchronous circuit behaviour with OR causality. Technical Report 463, University of Newcastle upon Tyne, 1993. (extended abstract in R. Valette, editor,

Application and Theory of Petri Nets 1994, LNCS 851, pages 568–587. Springer-Verlag, 1994).

[158] J.J. Žic. Time-constrained buffer specifications in CSP+T and timed CSP. *ACM Transactions on Programming Languages and Systems*, **16**(6):1661–1674, 1994.

[159] J. Zwiers and W. Janssen. Partial order based design of concurrent systems. In W.-P. de Roever J.W. de Bakker and G. Rozenberg, editors, *A decade of concurrency–reflections and perspectives*, volume 803 of *Lecture Notes in Computer Science*, pages 622–684. Springer-Verlag, 1994.

Glossary of notation

General notations

$\varnothing$	empty set, empty function, empty relation
$X \longrightarrow Y$	total function from X to Y
$X \longrightarrow_p Y$	partial function from X to Y
$dom(f)$	domain of function f
S^a	$S \cup \{ a \}$
$S^{a,b}$	$S \cup \{ a, b \}$
$=_{\text{iso}}$	isomorphism between labelled transition systems
$\approx_{te}$	testing equivalence
$\approx$	weak bisimulation equivalence
$\sim$	configuration equivalence, strong bisimulation equivalence
ε	empty trace, empty lposet
$\triangleq$	is defined by
$[x]_{\mathcal{R}}$	equivalence class of x under relation $\mathcal{R}$
$\upharpoonright$	projection
$\circ$	function composition
$\mathcal{P}(S)$	powerset of set S
$\mathcal{R}^*$	reflexive and transitive closure of relation $\mathcal{R}$
S^*	set of finite sequences of elements in set S
$[\![\ ]\!]$	semantic mapping
$S_1 \bowtie_G S_2$	set of synchronized (on G) timed events in S_1 and S_2

Classes

Act	universe of observable actions
$\mathcal{A}$	universe of actions
DF	class of distribution functions
EBES	class of extended bundle event structures
BES	class of bundle event structures
DES	class of dual event structures
EDES	class of extended dual event structures
EBES_T	class of timed event structures
EBES_R	class of real-time event structures
EBES_S	class of stochastic event structures
EBES_U	class of urgent event structures
EBES_P	class of probabilistic event structures

E_U	universe of events
LTS	class of labelled transition systems
$\mathbb{R}$	set of real numbers
Time	time domain

Behaviour expressions

$\mathbf{0}$	inaction
$\surd$	successful termination
$a\,;\,B$	action-prefix
$(T)\ a\,;\,B$	timed action-prefix
$(F)\ a\,;\,B$	stochastic action-prefix
$B+B$	choice
$B\ +_p\ B$	probabilistic choice
$B >> B$	enabling
$B\,[> B$	disrupt
$B\,\|\|_G\,B$	parallel composition
$B\,\|\|\,B$	full synchronization
$B\,\|\|\|\,B$	no synchronization
$B \setminus G$	hiding
$B[H]$	relabelling
$B \rhd B$	timeout
$B \blacktriangleright B$	watchdog
$\mathcal{U}_U(B)$	urgency operator
${}^t[\,B\,]$	time-shift of behaviour B with t time units
${}^t\{\,B\,\}$	behaviour B that can only perform events later than t
τ	silent action
δ	successful termination action
$\mathsf{Act}(B)$	set of observable actions in behaviour B
H	relabelling function from $\mathsf{Act}^{\tau,\delta} \longrightarrow \mathsf{Act}^{\tau,\delta}$
G	set of observable actions, $G \subseteq \mathsf{Act}$

Event structures

$init(\mathcal{E})$	initial events of event structure $\mathcal{E}$
$exit(\mathcal{E})$	termination events of event structure $\mathcal{E}$
$E(\mathcal{E})$	set of events of event structure $\mathcal{E}$
$T(\mathcal{E})$	set of event traces of event structure $\mathcal{E}$
$C(\mathcal{E})$	set of configurations of event structure $\mathcal{E}$
$L(\mathcal{E})$	set of lposets of event structure $\mathcal{E}$
$pos(\Gamma)$	set of events with a nonzero delay in Γ

$\#$	symmetric conflict relation
$\leadsto$	asymmetric conflict relation, time passing transition relation
$X \mapsto e$	bundle relation
$\prec$	flow relation
$\vdash$	enabling relation
$\rightleftharpoons$	interleaving relation
$\prec_C$	precedence relation on configuration C
$X \overset{t}{\mapsto} e$	timed bundle relation
l	event labelling function
$\mathcal{D}$	event delay function
$\mathcal{T}$	bundle delay function
$\mathcal{U}$	urgency predicate
π	probability function, limiting distribution of DTMC and CTMC
$\mathcal{E}[\sigma]$	event structure $\mathcal{E}$ after event trace σ
E^s	set of synchronizing events
E^f	set of non-synchronizing events
L°	intensional characterization of lposets
$L^\bullet$	operational characterization of lposets
$\trianglelefteq$	partial order on event structures
$\overline{\mathsf{op}}$	operator $\mathtt{op}$ on behaviours interpreted on event structures

Event traces

$\overline{\sigma}$	set of elements in σ
σ_i	prefix of σ upto the i-th element
$\sim_T$	timed configuration equivalence
$<_\sigma$	precedence relation on trace σ
$\mid \sigma \mid$	the number of elements in σ
$[\sigma]$	set of events in sequence σ of timed events
$\preccurlyeq$	faster than relation on timed traces
$\sigma \setminus G$	σ with actions in G hidden
$\sigma[H]$	σ relabelled by H
${}^t[\sigma]$	sequence σ of timed events shifted by t time units
$\mathsf{mx}(\sigma)$	maximal timing of event in σ

Time and stochastic related notions

T	set of time instants $T \subseteq \mathsf{Time}^\infty$
$[x, y]$	interval $\{ t \mid x \leqslant t \leqslant y \}$
$(x, y]$	interval $\{ t \mid x < t \leqslant y \}$
(x, y)	interval $\{ t \mid x < t < y \}$

$[x, y)$	interval $\{ t \mid x \leqslant t < y \}$
$x \ominus y$	$max(x-y, 0)$
$Pr\{ A \}$	probability of event A
F_U	distribution function of stochastic variable U
$E[U]$	expectation of stochastic variable U
$\circledast$	rate composition operator
$\mathbf{u}$	identity of product on distribution functions
$\mathbf{P}$	transition probability matrix
$\mathbf{Q}$	generator matrix
$(\underline{\alpha}, \mathbf{T})$	representation of phase-type distribution
$\oplus$	tensor sum
$\otimes$	tensor product
ϕ	limiting distribution of DTSMC
r_i	average residence time in state i

Miscellaneous

$\longrightarrow$	event transition relation
$\Longrightarrow$	observable action transition relation, probabilistic transition relation
$\longrightarrow\!\!\!\!\!\rightarrow$	timed event transition relation
$\longrightarrow_*$	combined time passing and event transition relation
$\bigsqcup_i d_i$	least fixed point of chain $d_1 \trianglelefteq d_2 \trianglelefteq \ldots$
$\perp$	least element of partial order

Index

Samenvatting

Het specificeren, ontwerpen, en analyseren van functionele aspekten van (gedistribueerde) systemen is een belangrijke toepassing van formele methoden. Recentelijk is er meer belangstelling ontstaan voor het bestuderen van kwantitatieve aspekten van dergelijke systemen gebaseerd op formele methoden. Diverse uitbreidingen van formele methoden zijn bekend uit de literatuur waarbij het optreden van een aktie een bepaalde kans kan worden toegekend en/of waarbij het tijdstip van optreden van een aktie kan worden aangegeven.

Een belangrijke reden voor het verrijken van formele methoden met kwantitatieve informatie is het mogelijk maken van de analyse van prestatiekenmerken van een systeemontwerp. Hierdoor kan de efficiëntie van verschillende ontwerpalternatieven worden bepaald zodat al in een vroeg stadium van het ontwerpproces kan worden afgezien van een bepaald ontwerp, omdat deze in onvoldoende mate aan de gewenste prestatiekenmerken voldoet. Dit voorkomt kostbaar herontwerp in latere ontwerpfasen. Een formele specificatie die kwantitatieve informatie bevat is ook bruikbaar voor het ontwikkelen van prestatiemodellen, zoals Markov ketens en wachtrijsystemen, op een begrijpbare en effectieve wijze vanuit systeemspecificaties.

De formele methoden waarvan kwantitatieve uitbreidingen bekend zijn, zijn veelal gebaseerd op de *interleaving* (of: verweving) van causaal onafhankelijke akties. Interleaving modellen abstraheren van het feit dat systemen feitelijk bestaan uit een aantal (deels) onafhankelijke deelsystemen. De globale toestand van het systeem wordt als uitgangspunt genomen, zonder daarbij het distributie-aspekt te vertegenwoordigen. Het systeemgedrag wordt gemodelleerd door het beschouwen van totaal geordende sequenties van akties waarin akties van het ene onafhankelijke deelsysteem worden verweven met akties van andere deelsystemen.

Dit proefschrift behandelt kwantitatieve en kwalitatieve uitbreidingen van *eventstrukturen*, een belangrijke representant van partiële order, of zogenaamde *noninterleaving* modellen voor concurrente systemen. Uitbreidingen die aan de orde komen zijn bijvoorbeeld de behandeling van tijdsaspekten, zowel in de normale als stochastische zin, urgentie van optreden, en probabiliteitsaspekten. Tot op heden heeft de behandeling van deze noties in de kontekst van noninterleaving modellen nauwelijks de aandacht gekregen.

Noninterleaving modellen abstraheren niet van het feit dat systemen bestaan uit een aantal (deels) onafhankelijke deelsystemen en het begrip 'globale toestand' speelt geen voorname rol in deze modellen. Het systeemgedrag wordt gemodelleerd door het beschouwen van geordende sequenties van akties die niet totaal geordend behoeven te zijn, maar partieel geordend. De causale afhankelijkheden worden weergegeven door deze partiële ordening.

Interleaving en noninterleaving modellen zijn complementair ten op zichte van elkaar in het systeemontwerpproces. Hoewel we in dit proefschrift voor het merendeel noninterleaving modellen beschouwen, zullen we ook de ingrediënten presenteren voor het verkrijgen van overeenkomende interleaving modellen. Hierdoor kunnen beide type modellen op een coherente wijze worden toegepast en is een vergelijking mogelijk tussen onze modellen en die uit de literatuur.

Uitgangspunten voor dit proefschrift zijn

- *extended bundle event structures*, een aangepaste versie van de traditionele eventstrukturen van Winskel die tegemoet komt aan de specifieke eisen van synchronisatie met meerdere partijen en disruptie, en
- *procesalgebra's*, abstracte beschrijvingsformalismen voor gedistribueerde systemen die bestaan uit een aantal krachtige operatoren om systeemspecificaties samen te stellen.

Extended bundle event structures bestaan uit *gelabelde events* die gebeurtenissen van akties (aangegeven door het label) modelleren, een *bundle* relatie die causale afhankelijkheden tussen events aangeeft, en een (asymmetrische) *conflict* relatie die uitsluitingen tussen events aangeeft. Eventstrukturen, in het bijzonder extended bundle event structures, worden behandeld in Hoofdstuk 2.

De bundle relatie brengt een verzameling events, de bundle verzameling, in verband met een event. De interpretatie is dat één event in de bundle verzameling moet zijn opgetreden om het optreden van het event waarmee het in relatie staat te doen optreden (dat is, te veroorzaken). Alle events in een bundle verzameling staan onderling met elkaar in conflict zodat slechts één event in zo'n verzameling kan optreden. Wanneer deze eis wordt losgelaten kunnen meerdere events in een bundleverzameling optreden en wordt de uitdrukkingskracht vergroot, dat wil zeggen, zogenaamde *disjunktieve causaliteit* wordt ondersteund. In Hoofdstuk 3 wordt onderzocht hoe *gelabelde partiële ordeningen* (lposets), die in dit proefschrift worden gebruikt als onderliggend semantisch model van eventstrukturen, kunnen worden gegenereerd als deze eis vervalt. In dit hoofdstuk worden ook een aantal bruikbare transformatieregels bepaald voor het resulterende model die gelijkheid in termen van lposets bewaren, en beschouwd verder nog een symmetrische irreflexieve *interleaving relatie* tussen events.

Eventstrukturen beschrijven systeemgedrag met behulp van causale ordeningen (bundles) tussen events en hun onderlinge uitsluitingen (conflicten). Om het beschrijven van tijdsafhankelijke systemen, zoals communicatieprotocollen, mogelijk te maken beschouwen we het concept *tijd*. Hoofdstukken 4, 6 en 7 behandelen uitvoerig de toevoeging van tijd aan extended bundle event structures. Real-time event structures kennen een verzameling tijdstippen toe aan bundles, die de relatieve tijdseisen tussen causaal afhankelijke events weergeven, en aan events, om absolute tijdseisen weer te kunnen geven (Hoofdstukken 4 en 7). Urgente event structures staan alleen de specificatie van minimale tijdseisen toe, maar bevatten *urgente* events, events die moeten optreden zodra ze mogelijk zijn (Hoofdstuk 6). Timeouts zijn een typisch fenomeen die door urgent events kunnen worden gemodelleerd. De veralgemenisering richting de notie van tijd van een meer stochastische aard wordt behandeld in Hoofdstuk 8. Stochastische event structures kennen *verdelingsfunkties* toe aan events en bundles, in plaats van verzamelingen tijdstippen. Uiteindelijk behandelen we in Hoofdstuk 9 de toevoeging van *probabiliteit* aan extended bundle event structures. Een probabiliteit kan worden toegekend aan een event die aangeeft wat de kans is dat dat event daadwerkelijk optreedt gegeven dat het kan optreden.

Eventstrukturen zijn zeer geschikt voor het geven van een noninterleaving semantiek van procesalgebra's op een *compositionele* wijze. Dit houdt in dat de interpretatie van een samengestelde procesalgebraische expressie gedefinieerd wordt als een funktie van de interpretaties van haar componenten. In dit proefschrift onderzoeken we of de kwantitatieve uitbreidingen van eventstrukturen kunnen worden gebruikt om een noninterleaving semantiek te geven van procesalgebra's met kwantitatieve informatie. Hiertoe gebruiken we de procesalgebra PA als basis, in feite de internationaal gestandaardiseerde procesalgebra LOTOS met een wat beknoptere syntax. De gehanteerde principes zijn echter ook bruikbaar voor gerelateerde procesalgebra's zoals CCS van Milner en CSP van Hoare. Voor iedere kwantitatieve variant van PA hebben we geprobeerd de noninterleaving semantiek van PA zoveel mogelijk te behouden, zodat *maximale compatibiliteit* wordt gegarandeerd.

De kwantitatieve uitbreidingen van procesalgebra's die we beschouwen zijn real-time varianten die *timeout*, *watchdog* en *urgency* operatoren bevatten, stochastische varianten waarin het tijdstip van voorkomen van akties wordt bepaald door *exponentiële*, of de algemenere en praktisch meer bruikbare, *fase type* verdelingsfunkties, en een probabilistische variant die een (interne) *probabilistische keuze* operator bevat. Voor iedere variant wordt een denotationele semantiek gegeven in termen van de overeenkomende kwantitatieve uitbreiding van eventstrukturen. Dit wordt gedaan op een *modulaire wijze* zodat combinaties (zoals tijd en probabiliteit) op een eenvoudige wijze kunnen worden verkregen.

Bovendien wordt voor de meeste genoemde procesalgebra's een *operationele semantiek* gepresenteerd die gebaseerd is op events, dus voorkomens van akties, in plaats van de akties zelf (zoals te doen gebruikelijk in operationele semantiek). Zo'n operationele semantiek schept een basis voor de vergelijking van ons werk met bestaande kwantitatieve uitbreidingen van interleaving modellen. De operationele regels voor het real-time geval zijn een nieuwe (en minimale) uitbreiding van het ongetimede geval; voor het urgente geval verkrijgen we regels die sterk overeenkomen met een voorstel van Bolognesi, Lucidi en Trigila; voor het stochastische geval met exponentiële verdelingen vormen de verkregen regels een basis voor verschillende bestaande stochastische procesalgebra's en voor het probabilistische geval verkrijgen we regels die gerelateerd (doch iets eenvoudiger) zijn aan het werk van Hansson en Jonsson. De relatie tussen de verschillende operationele semantieken en denotationele semantiek wordt uitgebreid onderzocht.

Hoofdstuk 10 behandelt recursie in alle varianten van procesalgebra's uit dit proefschrift. Gebruik makende van standaard domeintheorie wordt de denotationele semantiek van recursief gedefinieerde processen voor de kwantitatieve uitbreidingen van PA bepaald. Ook wordt de operationele semantiek gebaseerd op events uitgebreid met recursie. Aangetoond wordt dat de relatie tussen denotationele en operationele semantiek ook geldt voor het recursieve geval.

Hoofdstuk 11 bevat een terugkijkende blik op het werk van dit proefschrift, vat de belangrijkste technische resultaten samen, en presenteert een aantal algemene conclusies.

Curriculum Vitae

6 october 1964	geboren te Krimpen aan den IJssel
1977 – 1983	Athenaeum B Carolus Clusius College Zwolle
september 1983 – december 1987	studie Informatica Universiteit Twente met lof afgestudeerd bij de vakgroep Systeem Programmatuur en Apparatuur
februari 1988 – februari 1990	tweede-fase opleiding Informatie- en Communicatietechniek Technische Universiteit Eindhoven afdeling Wiskunde & Informatica vakgroep Parallellisme en Architectuur
februari 1990 – april 1992	wetenschappelijk medewerker Philips Natuurkundig Laboratorium afdeling Information and Software Technology
april 1992 – april 1996	medewerker onderzoek Universiteit Twente faculteit Informatica vakgroep Tele-Informatica en Open Systemen