Compositional Modelling of Stochastic Hybrid Systems

The research for this thesis was funded by the European Commission project HYBRIDGE, IST-2001-32460.

Title: Compositional Modelling of Stochastic Hybrid Systems
Author: S.N. Strubbe
ISBN: 90-365-2290-0

Printed by Wöhrmann Print Service, The Netherlands.

COMPOSITIONAL MODELLING OF STOCHASTIC HYBRID SYSTEMS

PROEFSCHRIFT

ter verkrijging van
de graad van doctor aan de Universiteit Twente,
op gezag van de rector magnificus,
prof. dr. W.H.M. Zijm,
volgens besluit van het College voor Promoties,
in het openbaar te verdedigen
op donderdag 8 december 2005 om 16.45 uur

door

Stefan Nicolaas Strubbe
geboren op 24 Augustus 1977
te Woerden, Nederland

Dit proefschrift is goedgekeurd door de promotor
Prof. dr. A.J. van der Schaft

Contents

1

Introduction

1.1 Hybrid systems

In this thesis we present a modelling framework for compositional modelling of stochastic hybrid systems. Hybrid systems consist of a combination of continuous and discrete dynamics. The state space of a hybrid system is hybrid in the sense that it consists of a continuous component and a discrete component. The continuous state takes values in a continuum (normally an Euclidean space) and evolves continuously through this continuum. The discrete state takes values in a discrete set and evolves in the sense that it jumps to different values in this set at certain time instances. The continuous and discrete dynamics are in general connected to each other: for different discrete states the continuous dynamics will be different and a change of the discrete state may depend on the continuous variables. A typical example of a hybrid system is a thermostat controlling the temperature in a living room. The continuous state is the temperature which changes continuously through time and the discrete state is the state of the heater that can have only two values, *on* and *off*. The connection between discrete and continuous dynamics is clear here: if the discrete state is *on*, then the continuous state evolves as 'temperature increases', if the discrete state is *off*, then the continuous state evolves as 'temperature decreases'. The change of discrete state, controlled by the thermostat, depends on the continuous state as: switch heater *off* when the temperature is too high, switch heater *on* when the temperature is too low.

There have been many approaches for modelling hybrid systems. In the hybrid automata approach [vdSS00, Hen96, ACH$^+$95], the discrete state automata is extended to a hybrid state automata by assigning continuous variables to each location. Automata approaches making distinction between active (or output) actions and passive (or input) actions are [JSvdS03] and [LSV01]. In the process algebraic approach [BKU05, Cui04], the processes are represented by algebraic formulas, which can be combined by using operators (like parallel composition operators) to form new algebraic formulas. In the Petri net approach [DA98], discrete state Petri nets are extended with continuous variables to form a hybrid Petri net.

Where the approaches to hybrid systems are many, the approaches to stochastic hybrid systems are few. There are several forms of stochasticity that can be brought into a deterministic hybrid systems framework. The time instances of transitions between locations can be made probabilistic; selecting the target state of a transition can be made probabilistic; diffusion terms can be added to the differential equations (which determine the continuous dynamics). The models SHS (Stochastic Hybrid System) [HLS00], SDP (Switching Diffusion Process) [GAM93, GAM97] and GSHS (Generalized Stochastic Hybrid System) [BL04, Buj04] include all above forms of stochasticity. They are all stochastic analysis approaches to hybrid systems and they are all based on the PDP (Piecewise Deterministic Markov Process) model [Dav84, Dav93]. The models in [Blo03] and [GB04] are based on stochastic differential equations and include all above forms of stochasticity. For an overview on the above stochastic hybrid models we refer to [PBL03]. The model DCPN (Dynamically Colored Petri Net) [EB03], which is also based on the PDP model, has a Petri Net approach. DCPN, which does not include diffusion terms, is, as far as we know, the only model that aims at compositional modelling of stochastic hybrid systems.

Our approach to stochastic hybrid systems will be an automata approach with distinction between active and passive actions. The idea of establishing communication through active and passive actions (presented in Chapter 4) was inspired by the communication mechanisms from DCPN.

1.2 Compositional modelling

Researchers in the field of of hybrid systems come from both the systems and control community, where mainly continuous systems are studied, and from the computer science community, where mainly discrete event systems are studied. Both communities have developed notions of 'compositional modelling' for continuous systems and discrete event systems respectively.

With compositional modelling we mean that a complex system, which consists of a network of interacting subsystems, is modelled by first modelling the individual subsystems and second by modelling the interaction, i.e., by modelling how these subsystems can be connected to form the complex composite system. Since nowadays (hybrid) systems like software systems, telecommunication systems or air traffic management systems, often have a very complex structure, modelling these systems without compositional modelling methods is nearly impossible.

An example of composition for continuous systems is the behavioral approach [PW98, Wil97]. Here, composition is defined on a semantical level, where the semantics of a (continuous) system is formed by all possible trajectories of the system. This forms an open systems view: the open system can be influenced by another system, by letting this other system determine which of its trajectories is forbidden to be executed. Composition of two systems is then, on the semantical level, determined as the intersection of the trajectories of both systems. This notion of composition is for example used in [JSvdS03, Jul05] to connect the continuous dynamics in the composition of hybrid systems.

In the process algebra literature, composition of discrete event processes has its origins in [Mil89] and [Hoa85]. Here, interaction between processes is expressed via synchronized transitions. This process algebra concept of composition via synchronized transitions is also used for transition systems and automata. For connecting the discrete dynamics, it is also used for timed automata [AD94, D'A97] and hybrid automata [Hen96].

Compositional frameworks for hybrid systems are based on compositional frameworks for continuous systems and discrete event systems. See [ACH$^+$95] for the compositional model *hybrid systems*, see [Hen96] for the compositional model *hybrid automata*, see [LSV03] for the compositional model *Hybrid Input Output Automata* and see [EB03] for the compositional Petri net model DCPN. Based on such compositional models, tools have been developed for the modelling and analysis of hybrid systems. See for example [BSA04] for the tool CHARON in the context of stochastic hybrid systems.

In the approach for stochastic hybrid systems that we present in this thesis, we will not consider composition of continuous dynamics. It will be explained later why we make this choice. For the discrete behavior we will use the concept of synchronization of transitions.

1.3 Bisimulation

When two systems are composed to form a new composite system, the state space of the composite system equals the product space of the state spaces of the two component systems. This leads to the well-known state space explosion problem for composition: the state space of the composite system grows exponentially with the state spaces of its components. Sometimes, when there are many components involved, the total (discrete) state space is so large that it can not even be stored in memory. One technique that proved to be very effective in many cases for discrete event systems, is bisimulation. Bisimulation is an equivalence notion. For a system with a large state space it is often possible to find a system that is bisimulation equivalent to it, but has a smaller state space.

Bisimulation was first developed by Milner in the context of discrete event systems. See for example [Mil89] for this notion of bisimulation. Since then it has also successfully been applied to timed automata [AD94], probabilistic automata [LS91], IMCs (Interactive Markov Chains) [Her02], (continuous) dynamical systems [vdS04a, vdS04b] and hybrid systems [vdS04c, LPS00]. Bisimulation for IMCs, will turn out to be very interesting for us, because we can generalize this bisimulation notion to PDP-type systems, which are stochastic hybrid systems of the type that we are interested in. In [Her02] it is shown that IMC-bisimulation for Markov Chains (which is a subclass of IMCs) coincides with the much older notion of lumpability for Markov Chains (see [KS60]).

In all these cases of bisimulation, the so called substitutivity property holds, which means that components in a composition can be substituted by bisimulation equivalent components, while bisimulation is preserved for the composite system. This allows compositional state reduction through bisimulation.

1.4 Our contribution

We briefly describe for each chapter of the thesis what the contents and contributions of that chapter are.

Chapter 2 In this chapter we introduce some technical notions and results that we use in the other chapters of the thesis.

Chapter 3 In this chapter semantical models are defined and their behavior is explained. Each semantical model captures a specific part of the behavior of the syntactical models, which are IMCs, PDPs and CPDPs, that we describe in this thesis. The semantical models are TMS (Transition Mechanism Structure), CFSJS (Continuous Flow Spontaneous Jump System), FTS (Forced Transition Structure) and NTS (Non-deterministic Transition Structure). In the chapters hereafter we can then describe/define the behavior of the syntactical models in terms of these semantical models and because all syntactical models of this thesis use the same semantical models, we can formally compare their behavior by means of their semantics.

Chapter 4 In this chapter we introduce transition systems whose transitions are either active or passive. We explain and illustrate what types of interaction can be expressed through synchronization of active and/or passive transitions. For these active/passive transition systems, we define a composition operator, denoted by $|^P_A|$, that can express all types of interaction that we discussed. This chapter also forms a basis for Chapter 7, where we define active/passive composition for CPDPs. This chapter is based on [SL05].

Chapter 5 The compositional modelling framework CPDP that we present in this thesis has its origins in two other frameworks: IMCs and PDPs. In this chapter we introduce the syntactical model IMC (Interactive Markov Chains). IMCs have two types of transitions. The Markovian transitions of an IMC form the Markov chain structure of the IMC. The interactive transitions form the communication/interaction structure of the IMC. We give CFSJS/NTS semantics for IMCs. Through interactive transitions, IMCs can communicate/interact. This interaction is expressed by the composition operator that we treat in this chapter. The chapter finishes with the treatment of the equivalence notions of strong and weak bisimulation for IMCs. Under certain conditions, an IMC can be reduced by bisimulation to a Markov chain. The material of this chapter comes from [Her02].

Chapter 6 In this chapter we define the PDP (Piecewise Deterministic Markov Process) framework. A PDP is a stochastic process for which the strong Markov property holds. We give CFSJS/FTS and TMS semantics for PDPs and we show how the stochastic process of a PDP can be realized on the Hilbert cube. We finish the chapter by giving a PDP example where composition of different components is involved. Because PDPs cannot be modelled compositionally, the model has to be given in a monolithic way. In Chapter

7 we show how this system can be modelled in a compositional way in the CPDP framework. The material of this chapter comes from [Dav93].

Chapter 7 This chapter is the first of the three main chapters about the CPDP framework. We define the CPDP (Communicating PDP) automaton and give its semantics in terms of CFSJS/NTS. CPDPs have both active and passive transitions, and we define composition of CPDPs by using a generalized version of the $|^P_A|$ composition operator of Chapter 4. We prove that the class of CPDPs is closed under this composition operation. With this framework we can model PDP-type systems in a compositional way. We extend the CPDP model with a more powerful interaction structure. The result is called value passing CPDP, and in this extended framework CPDP components can communicate information concerning their continuous variables to each other. CFSJS/NTS semantics of value passing CPDP is given. Composition with the extended $|^P_A|$ operator is defined and it is shown that the class of value passing CPDPs is closed under composition. This chapter is based on [SJvdS03, SvdS05e].

Chapter 8 Non-determinism is present in the CPDP model. To resolve this non-determinism, a scheduler for CPDPs is introduced. CFSJS/NTS semantics of scheduled CPDPs is given. We show that by using the maximal progress property for scheduled CPDPs, the semantic domain changes from CFSJS/NTS to CFSJS/FTS, which is the same semantic domain as the one of PDP processes. This means that we can formally compare the behavior of scheduled CPDPs under maximal progress and the behavior of PDPs. We give conditions under which CPDPs can be transformed into PDPs (without changing the CFSJS/NTS semantics) and we give an algorithm which determines this transformation in a stepwise way. For situations where CPDP components have their own schedulers, it might be desirable to have a composition operation for scheduled CPDPs. We investigate for which scheduled CPDPs such a composition 'makes sense', and for those scheduled CPDPs we define the composition such that the class of scheduled CPDPs is closed under this composition. This chapter is based on [SvdS05d, SvdS05a]

Chapter 9 In this chapter we define bisimulation for CPDPs and for scheduled CPDPs. We show that the substitutivity property holds and we show that under maximal progress, bisimilar scheduled CPDPs have the same stochastic behavior. This means that bisimulation can be used as a compositional reduction technique for analyzing the stochastic behavior of CPDPs. For a restricted class of CPDPs we give an algorithm that automatically computes bisimulation relations for CPDPs and scheduled CPDPs. This chapter is based on [SvdS05c, SvdS05a].

Chapter 10 In this chapter we model a part of a complex ATM (Air Traffic Management) system as a composite CPDP. This example illustrates the use of value passing and the different types of CPDP interaction that can be expressed through the $|^P_A|$ operator. This ATM system was originally modelled as a DCPN (Dynamically Colored Petri Net). We briefly describe the general DCPN model, we give the DCPN of this ATM system and we compare it to the CPDP model. This chapter is based on [EB03, EKBO04, SvdS05b]

Chapter 11 In this chapter we draw conclusions and we point out directions for future research.

2 Preliminaries

The contents of this chapter are mainly taken from [Dav93], [Wil91]. It contains in very compact form the technical notions and some of the technical results that we use in the rest of the thesis.

2.1 Topological spaces and measure spaces

The state spaces of the models that we use in this thesis are all (isomorphic to) subsets of $\mathbb{R}^n$. To define probability measures on these spaces, we first need to define the concepts of topological space and measure space.

Definition 2.1. A *topological space* is a set X together with a collection $\mathcal{T}$ of subsets, called *open sets*, that have the following properties:

1. $X \in \mathcal{T}$ and $\emptyset \in \mathcal{T}$,
2. if $A, B \in \mathcal{T}$ then $A \cap B \in \mathcal{T}$,
3. if $A_\alpha \in \mathcal{T}$ for all $\alpha \in J$, where J is an arbitrary index set, then $\cup_{\alpha \in J} A_\alpha \in \mathcal{T}$.

$\mathcal{T}$ is then called a *topology* on X.

We use the standard topology on $\mathbb{R}^n$, in which a set $A \subset \mathbb{R}^n$ is open if for all $x \in A$ there exists an $\epsilon > 0$ such that $\{y \mid d(x,y) < \epsilon\} \in A$, where d is the standard metric, defined as $d(x,y) = (\sum_{i=1\cdots n}(x_i - y_i)^2)^{\frac{1}{2}}$.

Definition 2.2. A *σ-algebra* $\mathcal{F}$ on Ω is a class of subsets of Ω such that:

1. $F \in \mathcal{F}$ implies $\Omega \backslash F \in \mathcal{F}$ and
2. if $F_i \in \mathcal{T}$ for $i \in I$, where I is any countable index set, then $\cup_{i \in I} F_i \in \mathcal{F}$.

We call $(\Omega, \mathcal{F})$ a *measurable space*.

Let $\mathcal{S}$ be a collection of subsets of Ω. $\sigma(\mathcal{S})$, which we call the σ-algebra generated by $\mathcal{S}$, is defined as the smallest σ-algebra in Ω that contains all elements from $\mathcal{S}$. Let E be a topological space, then we define $\mathcal{B}(E)$ as the σ-algebra generated by the open sets of E. We also call this the *Borel σ-algebra* of E and we call its elements the *Borel sets* of E. The Borel σ-algebra is the most natural and most commonly used measurable space for defining *measures* and *probability measures*.

Definition 2.3. A *measure* m on a measurable space $(\Omega, \mathcal{F})$ is a function from $\mathcal{F}$ to $\mathbb{R}_+$ such that if F_i, for $i \in I$, where I is any countable index set, are disjoint, then $m(\cup_{i \in I} F_i) = \sum_{i \in I} m(F_i)$. If for a measure m we have $m(\Omega) = 1$, then we call m a *probability measure*.

A commonly used measure on $\mathbb{R}^n$ is the Lebesgue measure.

Definition 2.4. Consider the space $\mathbb{R}^n$. For all sets A of the form $A = [a_1, b_1] \times [a_2, b_2] \cdots \times [a_n, b_n]$, with $b_i > a_i$ for all $i = 1 \cdots n$, we define $\tilde{l}(A) = (b_1 - a_1)(b_2 - a_2) \cdots (b_n - a_n)$. There exists a unique measure l on $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n))$ such that $l(A) = \tilde{l}(A)$ for all A of the above form. This measure is called the *Lebesgue measure*.

We call $(\Omega, \mathcal{F}, m)$, where $(\Omega, \mathcal{F})$ is a measurable space and m is a probability measure, a *probability space*. All measurable spaces that we use are *Borel spaces*, which notion is defined as follows.

Definition 2.5. A bijective mapping ι from a measurable space $(X, \mathcal{B}(X))$ to a measurable space $(Y, \mathcal{B}(Y))$ is a *Borel isomorphism* if for each $B \in \mathcal{B}(X)$, $\iota(B) \in \mathcal{B}(Y)$ and for each $B \in \mathcal{B}(Y)$, $\iota^{-1}(B) \in \mathcal{B}(X)$. In this case we call $(X, \mathcal{B}(X))$ and $(Y, \mathcal{B}(Y))$ *Borel isomorphic*. A measurable space $(X, \mathcal{F})$ is a *Borel space* if it is Borel isomorphic to a Borel subset of a complete separable metric space.

2.2 Product spaces

Product spaces play an important role in our theory of composition of stochastic hybrid systems.

Definition 2.6. The product space of measurable spaces $(X, \mathcal{F}_X)$ and $(Y, \mathcal{F}_Y)$ is defined as $(X \times Y, \mathcal{F})$, where $\mathcal{F}$ is the σ-algebra generated by the sets $A \times B$, where $A \in \mathcal{F}_X$ and $B \in \mathcal{F}_Y$.

It can be seen that for Borel spaces E_1 and E_2 we have that

$$(E_1 \times E_2, \sigma(\mathcal{B}(E_1) \times \mathcal{B}(E_2))) = (E_1 \times E_2, \mathcal{B}(E_1 \times E_2)).$$

In other words, the σ-algebra of the product space is exactly the Borel σ-algebra. If m_1 is a probability measure on Borel space $(E_1, \mathcal{F}_1)$ and m_2 is a probability measure on Borel space $(E_2, \mathcal{F}_2)$, then there exists a unique probability measure m on the product Borel space $E_1 \times E_2$ satisfying $m(A \times B) = m_1(A) m_2(B)$ for all $A \in \mathcal{F}_1$ and $B \in \mathcal{F}_2$. We call this probability measure m the *product probability measure* of m_1 and m_2, and we write $m_1 \times m_2$ for m.

2.3 Stochastic processes

Definition 2.7. Let $(S, \mathcal{F})$ be a measurable space. A function h from S to $\mathbb{R}$ is *measurable* if for all $A \in \mathcal{B}(\mathbb{R})$ we have $h^{-1}(A) \in \mathcal{F}$, where $h^{-1}(A) := \{\xi \in E | h(\xi) \in A\}$.

A *random variable* is a measurable function from some measurable space to $\mathbb{R}$. Let T be a random variable defined on the measurable space $(E, \mathcal{F})$ and let m be a probability measure on $(E, \mathcal{F})$. Then, for $A \in \mathcal{B}(\mathbb{R})$, we define the probability that T takes its value in A as $\mathrm{P}(T \in A) := m(T^{-1}(A))$.

A *stochastic process* is a family of random variables $\{X_t\}$, with $t \in J$, where J is some index set, where for all $t \in J$, X_t is defined on the same probability space. If $J = \mathbb{R}_+$, then we call the stochastic process a *continuous time process.*

For computer simulation of stochastic processes, it is often needed to draw samples from various random variables. By drawing these samples, we determine a trajectory for the state of the process. We call such a trajectory that corresponds to a set of drawn samples an *execution path.* With the following result, we can create an execution path if we have a random generator for the interval $[0, 1]$ with uniform distribution.

Proposition 2.8. *If m is a probability measure on a Borel space E, then there exists a measurable function $\psi : [0, 1] \to E$ such that $l \circ \psi^{-1} = m$, where l is the Lebesgue measure.*

For drawing a sample from the random variable X, we can then draw a sample u from $U[0, 1]$, i.e., the uniform distribution on the interval $[0, 1]$, which then provides the sample $X(\psi(u))$ for random variable X.

2.4 Integration

In Chapter 8 we use *integration* over a subset of a measurable space. We define *integration* in a two steps. In the first step integration is defined for *simple functions.*

Let $(E, \mathcal{F}, m)$ be a probability space. A measurable function $f : E \to \mathbb{R}$ is called *simple* if it can be written as

$$f = \sum_{i=1}^{m} a_k I_{A_i},$$

where for $i = 1 \cdots m$ $A_i \in \mathcal{F}$ and I_A is the indicator function, i.e., $I_A(\xi)$ equals 1 if $\xi \in A$ and equals 0 otherwise. We define m_0 as a mapping from simple functions to $\mathbb{R}$ as

$$m_0(f) := \sum_{i=1}^{m} a_k m(A_i).$$

Let SF^+ be the set of all simple positive functions. Then we define for measurable function f

$$m(f) := \sup\{m_0(h) | h \in SF^+, h \leq f\} \leq \infty,$$

where $h \leq f$ means that $h(\xi) \leq f(\xi)$ for all $\xi \in E$. That $m(f)$ is uniquely defined in this way is guaranteed by the monotone convergence theorem from integration theory. Now we can define integrability and integration.

Definition 2.9. Let $f : E \to \mathbb{R}$ be a measurable function. We write $f = f^+ - f^-$, where

$$f^+(\xi) := \max\{f(\xi), 0\}, \quad f^-(\xi) := \max\{-f(\xi), 0\}.$$

f is called *integrable* if $m(f^+) < \infty$ and $m(f^-) < \infty$, and then we define

$$\int f \mathrm{d}m = m(f^+) - m(f^-).$$

We also use the notations

$$\int_{\xi \in A} f(\xi) \mathrm{d}m = \int_{\xi \in A} f(\xi) m(\mathrm{d}\xi) = m(g^+) - m(g^-),$$

where $g := I_A f$.

2.5 Ordinary differential equations

The continuous dynamics that we encounter in this thesis, can all be expressed by ordinary differential equations (ODEs). In order to guarantee a unique solution, we only consider vector fields that are *locally Lipschitz*, which notion is defined as follows.

Definition 2.10. A vector field $f : \mathbb{R}^n \to \mathbb{R}^n$ is called *Lipschitz continuous* on $A \subset \mathbb{R}^n$ if there exists a constant L such that

$$||f(t) - f(s)|| \leq L||t - s||, \quad \text{for all } t, s \in A,$$

where $||\cdot||$ is the *Euclidean norm*, which is defined for $x \in \mathbb{R}^n$ as $||x|| := (\sum_{i=1}^n x_i^2)^{\frac{1}{2}}$. f is called *locally Lipschitz* if it is Lipschitz continuous for any compact set A.

We will use the following result.

Proposition 2.11. *If $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^m \to \mathbb{R}^m$ are locally Lipschitz, then $f \times g : \mathbb{R}^{n+m} \to \mathbb{R}^{n+m}$, which is defined as $f \times g(x_1, x_2) = (f(x_1), g(x_2))$, is also locally Lipschitz.*

We have the following useful result concerning ODEs with locally Lipschitz vector fields.

Theorem 2.12. *Let $f : \mathbb{R}^n \to \mathbb{R}^n$ be a locally Lipschitz vector field. Then, the ordinary differential equation*

$$\dot{x}(t) = f(x(t)), \quad x(0) = x_0 \in \mathbb{R}^n,$$

has a unique solution for $t \in [0, t_e[$, where $t_e \leq \infty$ is the explosion time*, i.e., the first time such that $\lim_{t \to t_e} ||x(t)|| = \infty$.*

For the vector fields we use in this thesis, we assume that there are no explosions, i.e., $t_e = \infty$. Then the solution of the differential equation is uniquely determined for all $t \geq 0$. We will normally denote the solution of an ODE by a *flow map* $\phi : \mathbb{R}_+ \times \mathbb{R}^n \to \mathbb{R}^n$, where $\phi(t, x)$ equals the solution at time t when the ODE is initiated at state x at time zero.

3

Semantical models

All models used in this thesis fall into either the class of syntactical models or into the class of semantical models. Syntactical models are the models or modelling languages/paradigms that are used to model the systems that we want to analyse/control. Semantical models are used to capture/express the behavior of a syntactical model. Semantical models are also used to compare syntactical models with each other. For example, if two syntactical objects have the same semantics, they can be regarded equivalent.

In this thesis we consider three syntactical and four semantical models. The syntactical models are: Interactive Markov Chains (IMCs), Piecewise Deterministic Markov Processes (PDPs) and Communicating Piecewise Deterministic Markov Processes (CPDPs). The semantical models are: Transition Mechanism Structures (TMSs), Non-deterministic Transition Systems (NTSs), Continuous Flow Spontaneous Jump Systems (CFSJSs) and Forced Transition Systems (FTSs). We can distinguish different levels for the semantical and syntactical models that we use. If the behavior of a semantical model M_1 is expressed within semantical model M_2, then M_1 is a higher level semantical model than M_2. Then, from high to low we consider the following levels.

- Syntactical level: IMC, PDP, CPDP
- High semantical level: CFSJS, NTS
- Intermediate semantical level: FTS
- Low semantical level: TMS

In this chapter we define all semantical models and we show how these models are related to each other, i.e., how lower semantical objects express the behavior of higher semantical objects.

The main syntactical models of this thesis are the PDP and CPDP models. All semantical models in this chapter will be used to capture a certain part of the behavior of PDP or CPDP type systems. PDPs and CPDPs are stochastic systems, but they do not contain diffusions. Therefore all semantical models do also not contain diffusions.

3.1 Transition Mechanism Structure

A transition mechanism structure gives us the random variables and the flow maps that are necessary to determine execution paths of a stochastic (hybrid) system. Once we know the TMS of a system, we can directly determine the stochastic process and the stochastic execution paths of the system. The stochastic process or the execution paths can be used to analyze the systems (stochastic) behavior.

A TMS consists of two parts: 1. a transition mechanism which determines the time of a transition and the target state of the transition, 2. a flow map which determines the continuous flow between two transitions. The semantical model TMS is formally defined as follows.

Definition 3.1. A *transition mechanism structure* (TMS) is a tuple (E, ξ_0, ϕ, TM). E is a Borel state space, ξ_0 is the initial state. ϕ is a flow map, i.e., the process evolves from state ξ_0 at time zero to state $\phi(t, \xi)$ at time t if no transitions occur in the interval $[0, t]$, etc. TM is a *transition mechanism* on E. A *transition mechanism* on a Borel space E is a pair (T, Q) with $T : E \to RV(\bar{\mathbb{R}}_+, \mathcal{B}(\bar{\mathbb{R}}_+))$, where $RV(\Omega, \mathcal{F})$ denotes the set of all random variables (defined on any probability space) taking values in the measurable space $(\Omega, \mathcal{F})$ and where $\bar{\mathbb{R}}^+ := \mathbb{R}_+ \cup \{\infty\}$, and with $Q : E \to Prob(E)$. Here $Prob(E)$ denotes the set of all probability measures on the measurable space $(E, \mathcal{B}(E))$.

Given a state $\xi \in E$, the transition mechanism $TM = (T, Q)$ determines a transition-time t and a transition target state ξ' by drawing a sample t from the random variable $T(\xi)$ followed by drawing a sample ξ' from the probability measure $Q(\phi(t, \xi))$. We also say that with this procedure we have drawn the sample (t, ξ') from the transition mechanism $TM(\xi)$. If the sample ∞ is drawn from $T(\xi)$, then this is not followed by drawing a sample from Q. We then say that the sample $(\infty, \emptyset)$ is drawn from $TM(\xi)$.

Determining the stochastic process of a TMS is treated in Chapter 6. An execution path of a TMS (E, ξ_0, ϕ, TM) is generated as follows. Draw a sample (t_1, ξ_1) from $TM(\xi_0)$. For $t \in [0, t_1[$ the execution path has value $\phi(t, \xi_0)$. Draw a sample (t_2, ξ_2) from $TM(\xi_1)$. For $t \in [t_1, t_1 + t_2[$, the execution path has value $\phi(t - t_1, \xi_1)$. Draw a sample (t_3, ξ_3) from $TM(\xi_2)$, etc.

TMS forms the lowest semantical level that we use. The TMS of a system can be derived from the CFSJS and the FTS semantics of the system. This derivation is done in section 3.4.

3.2 Continuous Flow Spontaneous Jump System (CFSJS)

For both the syntactical models PDP and CPDP, we have that transitions, where the state instantaneously jumps to another state, can happen in two ways: 1. Spontaneously (with some probability distribution), 2. Forced (when the state reaches some 'forbidden' area and is forced to jump to a another state). In between transitions, the state of a PDP/CPDP evolves continuously. The part of the PDP/CPDP system behavior concerning the continuous evolution and the spontaneous transitions is captured by a CFSJS. The part concerning the forced transitions is captured by an FTS.

Definition 3.2. A CFSJS is a tuple $(E, \xi_0, \phi, \lambda, Q)$. The state space E is a Borel space. ξ_0 is the initial state, $\phi : \mathbb{R}_+ \times E \to E$ is the flow map, $\lambda : E \to \mathbb{R}_+$ is the jump rate and $Q : E \to Prob(E)$ is the transition measure.

The jump rate $\lambda(\xi)$ of a CFSJS at state ξ determines the probability of a spontaneous transition 'near' state ξ as follows: if the system is at state ξ at time t, then the probability that a spontaneous transition occurs in the interval $[t, t + \Delta t]$ equals $\lambda(\xi)\Delta t + o(\Delta t)$, where $o(\Delta t)$ denotes a function such that $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$. In other words, for Δt small enough, the probability that a spontaneous transition occurs in the interval $[t, t + \Delta t]$ equals approximately $\lambda(\xi)\Delta t$. (This means that if the process is at state ξ at time $\hat{t}$ and the next jump happens at time $\hat{t} + t$, then t is determined by a Poisson Process with intensity $\lambda(\phi(t, \xi))$, see [Kin93]).

If a systems behavior is completely captured by a CFSJS, i.e., there are no forced transitions, then the CFSJS completely determines the stochastic executions of the system. By determining the TMS of a CFSJS, we indirectly determine the stochastic process/executions of the CFSJS.

Definition 3.3. The TMS (transition mechanism structure) of a CFSJS $(E, \xi_0, \phi, \lambda, Q)$ is defined as $(E, \xi_0, \phi, (T, Q))$, where, for $\xi \in E$, the *survivor function* $\Psi_{T(\xi)}(t)$ of $T(\xi)$, is defined as

$$\Psi_{T(\xi)}(t) = \mathrm{e}^{-\int_0^t \lambda(\phi(s,\xi))\mathrm{d}s}. \tag{3.1}$$

The *survivor function* $\Psi_{T(\xi)}(t)$ is by definition equal to $\mathrm{P(T(\xi) > t)}$ and thus expresses the probability that $T(\xi)$ 'survives' the time instant t, or, in other words, expresses the probability that a transition does not occur until time t.

We show that (3.1) indeed expresses that if at time zero, i.e., the time of the previous transition, the process is at state $\hat{\xi}$ and at time t the process is at state ξ, then, given that no jump occurred in the interval $[0, t]$, the probability that a spontaneous transition occurs in the interval $[t, t + \Delta t]$ equals $\lambda(\xi)\Delta t + o(\Delta t)$.

$$\mathrm{P}(T(\hat{\xi}) \in [t, t + \Delta t] \mid T(\hat{\xi}) > t) = \frac{\psi_{T(\hat{\xi})}(t) - \psi_{T(\hat{\xi})}(t + \Delta t)}{\psi_{T(\hat{\xi})}(t)} =$$

$$1 - \mathrm{e}^{-\int_0^{t+\Delta t} \lambda(\phi(s,\hat{\xi}))\mathrm{d}s + \int_0^t \lambda(\phi(s,\hat{\xi}))\mathrm{d}s} = 1 - \mathrm{e}^{-\int_0^{\Delta t} \lambda(\phi(s+t,\hat{\xi}))\mathrm{d}s},$$

which, after Taylor expansion, equals $\lambda(\xi)\Delta t + o(\Delta t)$.

3.2.1 Reassessing the jump time

Let X be a TMS with transition mechanism (T, Q) and flow map ϕ. We can execute X as described in Section 3.1. Suppose that during such an execution we lose at some time $\hat{t}$, while the process is at state $\xi_{\hat{t}}$, the information of the last drawn sample from T. Can we now continue the execution path from $\xi_{\hat{t}}$ in a correct way, or do we have to start a new execution path from ξ_0? Let t_l denote the time of the previous transition (before $\hat{t}$) and let ξ_{t_l} be the state of the execution path at time this time t_l, which is the target state of the transition at time t_l. If t_l and ξ_{t_l} are known, then it is correct to continue the stochastic execution from $\hat{t}$ as follows: draw a sample $\tilde{t}$ from $\tilde{T}$, where $\tilde{T}$ is a random variable such that

$$\mathrm{P}(\tilde{T} > t) = \mathrm{P}(T(\xi_{t_l}) > \hat{t} - t_l + t | T(\xi_{t_l}) > \hat{t} - t_l).$$

Now let the execution path flow from state $\xi_{\hat{t}}$ to state $\phi(\tilde{t}, \xi_{\hat{t}})$ and switch at state $\phi(\tilde{t}, \xi_{\hat{t}})$ according to the measure $Q(\phi(\tilde{t}, \xi_{\hat{t}}))$. From the new state we again draw a new sample from the transition mechanism, etc.

Now we show that we can determine $\mathrm{P}(\tilde{T} > t)$ without knowing t_l and ξ_{t_l}, and consequently we can conclude that we can correctly continue the execution path from state $\hat{\xi}$ without having any information except that the process is at state $\hat{\xi}$.

The transition mechanism (T, Q) of a CFSJS has a special structure: the following property holds.

$$\mathrm{P}(T(\xi) > \hat{t} + t | T(\xi) > \hat{t}) = \mathrm{P}(T(\phi(\hat{t}, \xi)) > t). \tag{3.2}$$

This property is called the Markov property . Because of this property we have

$$\mathrm{P}(\tilde{T} > t) = P(T(\hat{\xi}) > t).$$

Thus, if during the execution of a CFSJS, we loose the information of the last drawn sample before time $\hat{t}$ and state $\xi_{\hat{t}}$, then because of property (3.2), we can continue the execution by considering $\xi_{\hat{t}}$ as a state right after some switch. This means that we draw a sample $\tilde{t}$ from $T(\xi_{\hat{t}})$, followed by drawing a sample from $Q(\phi(\tilde{t}, \xi_{\hat{t}}))$, etc. As we will see in the next section, this observation makes it possible that the behavior of a system that consists of two CFSJSs executed at the same time, can be expressed as a single CFSJS.

3.2.2 Representing two parallel CFSJSs as a single CFSJS

Let $X = (E_X, \xi_{X,0}, \phi_X, \lambda_X, Q_X)$ and $Y = (E_Y, \xi_{Y,0}, \phi_Y, \lambda_Y, Q_Y)$ be two CFSJSs. Assume that at time t_0 both the processes X and Y are started. Let $\xi_X : \mathbb{R}_+ \to E_X$ be an execution path generated by the TMS of X and let $\xi_Y : \mathbb{R}_+ \to E_Y$ be an execution path generated by the TMS of Y. Then we call $\xi : \mathbb{R}_+ \to E_X \times E_Y$, where $\xi(t) = (\xi_X(t), \xi_Y(t))$, an execution path of the simultaneous execution of X and Y on the combined state space $E_X \times E_Y$. We show that these combined execution paths can be generated by the TMS of a single CFSJS denoted as $X|Y$. In Chapter 7 we need this result when two components (i.e., two CPDPs) that are executed in parallel, need to be represented as a single component (i.e., as a single CPDP). The state space of $X|Y$ is the product space $E_X \times E_Y$. As we will see, a jump for X with measure m_X is then represented in $X|Y$ by a jump with measure $m_X \times Id$. Here Id represents the identity reset measure that resets the Y part of the state of $X|Y$ such that it does not change with probability one. For Y this can be interpreted as drawing a new sample because the information of the last drawn sample has been lost, which does not influence the stochastic behavior of Y as we saw in the previous section.

Suppose that after the start at t_0, X switches for the first time at t_1 at state $\xi_{X,1}$ and Y does not switch before t_1. Then at t_1, the state of X is reset by measure $Q(\xi_{X,1})$. Let $\xi_{Y,1}$ be the state of Y at time t_1. The state of Y is not reset at time t_1, but from Section 3.2.1 we know that the stochastic behavior of Y will not change if we reset the state of Y at time t_1 with the identity reset measure $Id(\xi_{Y,1})$. ($Id(\xi_{Y,1})$ resets the state of Y to state $\xi_{Y,1}$ with probability one). Then for the execution of Y, the state does not change at time t_1, but a new sample is drawn from the transition mechanism at state $\xi_{Y,1}$, which does not influence the stochastic execution according to Section 3.2.1.

We define a transition mechanism (T, Q) on the state space $E_X \times E_Y$ such that generating an execution path of (T, Q) is equal to generating a combined execution path for X and Y as described above.

Let for all $(\xi_X, \xi_Y) \in E_X \times E_Y$ the random variable $T(\xi_X, \xi_Y)$ be equal to $\min\{T_X(\xi_X), T_Y(\xi_Y)\}$. Then T determines the jump time of 'either X or Y'. It can be seen that the survivor function of $T(\xi_X, \xi_Y)$ equals

$$\Psi_{T(\xi_X,\xi_Y)}(t) = \mathrm{e}^{-\int_0^t \lambda_X(\phi(s,\xi_X)+\lambda_Y(\phi(s,\xi_Y))\mathrm{d}s}. \tag{3.3}$$

If a switch happens at combined state (ξ_X, ξ_Y), then it can be seen that the probability that this switch is a switch of X is equal to $\frac{\lambda_X(\xi_X)}{\lambda_X(\xi_X)+\lambda_Y(\xi_Y)}$ and the probability that this switch is a switch of Y is equal to $\frac{\lambda_Y(\xi_Y)}{\lambda_X(\xi_X)+\lambda_Y(\xi_Y)}$.

If X switches at state (ξ_X, ξ_Y), the reset measure $Q_X(\xi_X) \times Id(\xi_Y)$ is used and if Y switches at state (ξ_X, ξ_Y), the reset measure $Id(\xi_X) \times Q_Y(\xi_Y)$ is used. Then we get for Q

$$\begin{aligned} Q(\xi_X, \xi_Y) = {} & \frac{\lambda_X(\xi_X)}{\lambda_X(\xi_X) + \lambda_Y(\xi_Y)} Q_X(\xi_X) \times Id(\xi_Y) + \\ & \frac{\lambda_Y(\xi_Y)}{\lambda_X(\xi_X) + \lambda_Y(\xi_Y)} Id(\xi_X) \times Q_Y(\xi_Y). \end{aligned} \tag{3.4}$$

Define CFSJS $X|Y$ as $(E_X \times E_Y, (\xi_{X,0}, \xi_{Y,0}), (\phi_X, \phi_Y), \lambda, Q)$, where

$$\lambda(\xi_X, \xi_Y) = \lambda_X(\xi_X) + \lambda_Y(\xi_Y).$$

The TMS of $X|Y$ equals (T, Q) and therefore CFSJS $X|Y$ generates the same execution paths (with the same probabilities) as the combination of execution paths of X and Y.

If $|$ denotes the operator that maps two CFSJSs to the combined CFSJS, then it can be seen that $|$ is associative and the combination of X,Y and Z can be expressed as either $(X|Y)|Z$ or $X|(Y|Z)$.

3.3 Forced Transition Structure (FTS)

If a system has forced transitions, then the behavior of the system concerning these forced transitions can be captured as an FTS.

Definition 3.4. An FTS is a tuple $(E, \mathcal{T})$. The state space E is a Borel space. $\mathcal{T} \subset E \times \mathit{Prob}(E)$ is the transition relation. For each $\xi \in E$, there exists at most one measure m such that $(\xi, m) \in \mathcal{T}$. If a state ξ is such that there exists an m such that $(\xi, m) \in \mathcal{T}$, then we call ξ an *enabled state* of the FTS.

If a system X has corresponding FTS $(E, \mathcal{T})$, then if $(\xi, m) \in \mathcal{T}$ means that if X reaches state ξ at some time t, then X is forced to switch at this state and the target state of the switch is determined by measure m.

3.4 CFSJS combined with FTS

The behavior of a PDP and, under certain conditions, the behavior of a CPDP can be captured as a combination of a CFSJS and an FTS. In fact, this combination means that the process runs as the CFSJS until an enabled state of the FTS is reached. Then the forced transition is executed, and the CFSJS execution continues from the state right after the forced transition. We now show how this combination of CFSJS and FTS behaves in terms of TMS.

Let (X_C, X_F), where $X_C = (E, \xi_0, \phi, \lambda, Q)$ is an CFSJS and $X_F = (E, \mathcal{T})$ is an FTS, be the combined semantics of a system X with state space E. For each $\xi \in E$ we define $t_*(\xi)$ as

$$t_*(\xi) := \begin{cases} \inf\{t \geq 0 | \phi(t, \xi) \text{ is an enabled state of } X_F\} \\ \infty \text{ if no such time exists.} \end{cases}$$

Thus, $t_*(\xi)$ is the maximum time before a jump surely occurs from the moment that the process is in state ξ. Either a jump occurs before time $t_*(\xi)$ because of the CFSJS part or a forced jump happens at time $t_*(\xi)$.

The transition mechanism structure of (X_C, X_F) is then equal to $(E, \xi_0, \phi, (T, \tilde{Q}))$, where $\tilde{Q}(\xi)$ equals $Q(\xi)$ if ξ is not an enabled state of X_F and $\tilde{Q}(\xi)$ equals m if ξ is an enabled state of X_F, where m is such that $(\xi, m) \in \mathcal{T}$. The survivor function of T (which definition we take from [Dav93]), equals

$$\Psi_{T(\xi)}(t) = I_{(t<t_*(\xi))} \mathrm{e}^{-\int_0^t \lambda(\phi(s,\xi))\mathrm{d}s}. \tag{3.5}$$

3.5 Non-deterministic Transition System (NTS)

Besides spontaneous and forced transitions, we can also distinguish *non-deterministic transitions*. We call a ξ-enabled transition *non-deterministic* if, when a process reaches state ξ, the process has the potential to execute the transition but it is not forced to execute the transition. In other words, it is not determined whether the process should execute the transition. Forced transitions are clearly not non-deterministic since the process has no choice but is forced to execute a ξ-enabled forced transition when its state reaches ξ. Execution of spontaneous transitions is determined by random variables, spontaneous transitions are not non-deterministic therefore. We now define the semantical model NTS (Non-deterministic Transition Structure), whose transitions are non-deterministic.

Definition 3.5. An NTS is a tuple $(E, \Sigma, \mathcal{T})$. The state space E is a Borel space. Σ is a set of labels. $\mathcal{T} \subset E \times \Sigma \times Prob(E)$ is the transition relation.

We write '$\xi \xrightarrow{\sigma} m$' for '$(\xi, \sigma, m) \in \mathcal{T}$'. If a system X has corresponding NTS $(E, \Sigma, \mathcal{T})$, then $\xi \xrightarrow{\sigma} m$ has the meaning that if X reaches state ξ at some time t, then X has the possibility/potential to jump at this point on action σ and the target state of the jump is determined by measure m. Whether the transition is really executed at state ξ is not-determined. If there are multiple ξ-enabled transitions, then the process has, at state ξ, the potential to execute either one of them or to execute none of them. In the concurrent processes literature non-determinism is often used in a stricter sense than we do here. A ξ-enabled transition with label σ is then called non-deterministic if there is another ξ-enabled transition with the same label σ.

Actions σ are used for interaction between systems. In the next chapter we show how this interaction is established.

If a system has non-deterministic transitions, then the system is 'open' in the sense that its behavior can be influenced by other systems. Therefore, a system with non-deterministic transitions can, if the non-determinism is not resolved by for example a scheduler, not be stochastically executed. This means that we cannot determine the TMS of such a system.

We now give two simple examples that show how the behavior of a stochastic system can be captured in terms of CFSJS, FTS and NTS. In the first example the behavior is captured as a CFSJS together with an FTS, in the second example the behavior is captured as a CFSJS together with an NTS.

Example 3.6. The state x of system X takes value in $\mathbb{R}$ and evolves continuously as described by the ordinary differential equation $\dot{x} = 1$. The initial state $x_0 = 0$. A spontaneous transition may happen and the time of such a transition is exponentially distributed with parameter λ. The target state of such a transition is chosen with uniform distribution in $[0, 1]$. If x reaches the value 1, a transition is forced to happen. The target state is chosen with uniform distribution in $[0, 1]$.

The behavior of X can be captured as a CFSJS together with an FTS. The CFSJS equals $(\mathbb{R}, 0, \phi, \lambda, Q)$, where $\phi(t, x) = x + t$ for all states x and Q equals $U[0, 1]$, i.e., the uniform distribution on $[0, 1]$, for all states x. The FTS equals $(\mathbb{R}, \mathcal{T}_F)$, with $\mathcal{T}_F = \{(1, U[0, 1])\}$.

The TMS that corresponds to the combination of this CFSJS and FTS equals $(\mathbb{R}, 0, \phi, (T, Q))$, where for all states $x < 1$

$$\mathrm{P}(T(x) > t) = I_{(t<1-x)} \mathrm{e}^{-\lambda t}.$$

Example 3.7. The state x of system X takes value in $\mathbb{R}$ and evolves continuously as described by the ordinary differential equation $\dot{x} = 1$. The initial state $x_0 = 0$. A spontaneous transition may happen and the time of such a transition is exponentially distributed with parameter λ. The target state of such a transition is chosen with uniform distribution in $[0, 1]$. If $x \geq 1$, a transition is allowed but not forced to happen. The target state is chosen with uniform distribution in $[0, 1]$. The action of the transition is τ. The target state is chosen with uniform distribution in $[0, 1]$.

The behavior of X can be captured as a CFSJS together with an NTS. The CFSJS equals the CFSJS of Example 3.6. The NTS equals $(\mathbb{R}, \{\tau\}, \mathcal{T}_N)$, with $\mathcal{T}_N = \{(x, \tau, U[0, 1]) | x \geq 1\}$. Note that because of the presence of non-determinism, the behavior of the CFSJS together with the NTS can not be captured as a TMS.

3.6 Summary

In this section we summarize the semantical models that we defined in this chapter, and we show which models describe the behavior of which models. We also describe which semantical models are used to describe the behavior of the three syntactical models IMC, PDP and CPDP and based on that we show which type of transitions can be executed by these syntactical models. The formal semantics (in terms of these semantical models) of IMC, PDP and CPDP, will be given in chapters 5,6 and 7 respectively.

The behavior of an IMC is captured by a CFSJS and an NTS (as will be described in Chapter 5). A CFSJS has only spontaneous transitions and an NTS has only non-deterministic transitions. This means that an IMC has spontaneous transitions (captured by the CFSJS) and non-deterministic transitions (captured by the NTS). The latter transitions (when enabled) can occur at any time, but it is not determined (also not in a stochastic sense) when exactly. An IMC has no forced transitions. Because of the non-determinism, an IMC does not have a corresponding stochastic process and therefore its behavior cannot be captured as a TMS. (In [Her02], it is described how, by maximal progress, non-determinism might be resolved and consequently how the stochastic process of an IMC under maximal progress looks like).

The behavior of a PDP is captured by a CFSJS and an FTS (as will be described in Chapter 6). An FTS has only forced transitions. This means that a PDP has spontaneous transitions and forced transitions. Because there is no non-determinism, a PDP has a stochastic process and its behavior can be captured as a TMS. A TMS has both spontaneous and forced transitions.

The behavior of a CPDP is captured by a CFSJS an NTS (as will be described in Chapter 7). The same story above for the IMC applies to CPDP. In Chapter 8 we show how the non-determinism (captured by the NTS) can be resolved by using the *maximal progress* assumption and by using a *scheduler*. By resolving the non-determinism, the NTS is transformed into an FTS. In other words, the behavior of a scheduled CPDP under maximal progress is captured by a CFSJS and an FTS. Such a CPDP has a stochastic process and its behavior can be captured as a TMS.

4

The active/passive framework

One of the objectives of this thesis is to model complex systems in a compositional way. With a complex system, we mean a compound of subsystems that are running simultaneously and interact with each other. With this aim we introduced the NTS model. In this chapter we define how interaction between multiple NTSs can be established. Each NTS in the compound of NTSs is then a subsystem that can interact with the other subsystems in the compound. We will see in the last section of this chapter that the compound of NTSs can be expressed as another NTS. Before we do that, we first define interaction for normal labelled transition systems. Once we defined this, the definition can be easily generalized to NTSs.

Definition 4.1. A *labelled transition system* is a tuple $(X, \Sigma, \mathcal{T})$, where X is the set of states, Σ is the set of actions and $\mathcal{T} \subset X \times \Sigma \times X$ is the transition relation. $(x, \sigma, x') \in \mathcal{T}$ means that there is a transition from state x to state x' labelled with action σ.

Interaction means that two interacting systems can influence each others behavior. A common way in the process algebra literature of defining interaction for transition systems is by means of *synchronization* of transitions. Normally, a set of synchronization actions A is used and then composition of transition system X with transition system Y on the set A means that X and Y can execute their transitions independently except for the transitions with labels in A. These transitions in A should synchronize, which means that if $a \in A$ then X can execute an a-transition only when at the same time Y executes an a-transition, and vice versa. We call this kind of interaction *blocking interaction* because if Y has no a-transitions enabled at time t, and $a \in A$, then X can not execute its a-transitions at time t, in other words, Y blocks all a-transitions of X at time t. Vice versa, X can block the a-transitions of Y.

Let $|A|$ denote the composition operator that establishes blocking interaction for actions in A. Then, for transition systems, $|A|$ is normally formalized via *operational rules*. Operational rules are in Plotkin style [Plo81]: rule

$$\frac{A, B}{D}(C),$$

means 'if A and B are true, then D is true provided that C is true'. In other words, via this rule D is derived from A, B provided that C is true. Rules

$$\frac{A, B}{D}, \quad \frac{A}{D}(C) \quad \text{and} \quad \frac{A}{D}$$

mean 'if A and B are true, then D is true', 'if A is true, then D is true provided that C is true' and 'if A is true, then D is true', respectively.

$|A|$ maps two transition systems X and Y to a new transition system $X|A|Y$, which behaves as the composition of X and Y and the mapping is determined by the operational rules. Now we can formally define $|A|$.

Definition 4.2. Let $(X, \Sigma, \mathcal{T}_1)$ and $(Y, \Sigma, \mathcal{T}_2)$ be labelled transition systems and let $A \subset \Sigma$. Then $X|A|Y$ is defined as the labelled transition system $(X \times Y, \Sigma, \mathcal{T})$, where $\mathcal{T}$ is the least relation, i.e., the smallest subset of $(X \times Y) \times \Sigma \times (X \times Y)$, satisfying the following three rules

$$1. \frac{x \xrightarrow{a} x', y \xrightarrow{a} y'}{x|A|y \xrightarrow{a} x'|A|y'} (a \in A),$$

$$2. \frac{x \xrightarrow{a} x'}{x|A|y \xrightarrow{a} x'|A|y} (a \notin A), \quad 3. \frac{y \xrightarrow{a} y'}{x|A|y \xrightarrow{a} x|A|y'} (a \notin A),$$

where $x \xrightarrow{a} x'$ denotes $(x, a, x') \in \mathcal{T}_1$, $y \xrightarrow{a} y'$ denotes $(y, a, y') \in \mathcal{T}_2$ and $x|A|y \xrightarrow{a} x'|A|y'$ denotes $((x, y), a, (x', y')) \in \mathcal{T}$.

In this definition, rule 1 determines that transitions with label in A should synchronize and rules 2 and 3 determine that transitions with label not in A can be executed independently of the other system in the composition.

This chapter is structured as follows. The *trace semantics* of a labelled transition system consists of the set of all traces (i.e., strings of actions) that the labelled transition system can generate. Trace semantics is a commonly used semantics. Instead of defining a composition operator in an operational way as above, composition can also be defined as a mapping from the sets of traces of the two components to the set of traces of the composed system. We call this the trace semantics of the composition operator. In Section 4.2 we show that using the normal concept of traces, a compositional trace semantics of $|^P_A|$ can not be given. But, by defining a new concept of traces called *pie-traces*, a pie-trace semantics of $|^P_A|$ can be given and we give this semantics in this section. Pie-traces are enriched traces, i.e., they contain more information than ordinary traces. The concept of pie-trace is inspired by the concept of *event trace* defined in [Lan92].

4.1 The active/passive framework

We can distinguish two types of interactions between processes. Both are expressed through synchronization of transitions. First, *blocking*-interaction: both partners (for example the controller and the process) need to be able to do the action, otherwise the action will not take place. This is the type of interaction that we see in many process algebra models (e.g. [Hoa85, BB87]). Secondly, *broadcasting*- or *non-blocking*-interaction: one of the partners (the passive one) is not able to block the other (the active one). For example if a person (the passive partner) observes that a light (the active partner) is switched on. The person observes, but is not able to block the switching, i.e., the light could also be switched on without the person observing it.

Broadcasting-interaction can already be found in the literature, e.g. in broadcasting systems [Pra91] where several listening processes can receive (but not block) a signal, or in I/O automata [LSV03, LT88], where processes should be input-enabled, i.e., ready to receive an input in any state of the process, such that an output will never be blocked.

Most formalisms support only blocking interaction and some formalisms (broadcasting systems, I/O automata) support only broadcasting interaction. However, to our knowledge there exists no transition system based formalism that supports both blocking and broadcasting interaction. We will motivate that for certain applications it is desirable to have a formalism that supports both types of interaction. We use supervisory control systems as an example where both types of interaction are used.

Now we will introduce our framework, which is based on active and passive actions and which supports both blocking and broadcasting interaction. We introduce the framework in three steps,

1. By specifying *supervisory control systems*. We will see that we need both blocking and broadcasting interaction for this.

2. By specifying *modular supervisory control systems*, where a controller may consist of several modules. We will motivate that for this, in addition to blocking and broadcasting interaction, we need to control the scope of broadcasting interaction, which means that in a composition of more than two systems, we can specify which of the systems is involved in the broadcasting interaction and which of the systems are not.

3. By specifying *complex processes that consist of interacting subprocesses*. Here we will argue that we need the possibility to have multi-way synchronizations.

In each step we give motivating examples. The transition systems used in these examples all have a discrete state space. All results however also hold for transition systems with an infinite (continuous) state space. In the first step we explain how active and passive actions can be used to establish blocking and non-blocking interaction. In the second step we explain how to deal with observations in systems that consist of more than two components. In the third step we treat the issue of multi-way synchronization (can one active action synchronize with multiple passive actions or only with one passive action?).

Each of the three steps will contribute one or more operational rules for the composition operator.

4.1.1 Establishing blocking and non-blocking interaction

We consider two types of actions: *active* actions (denoted as a, b etc.) and *passive* actions (which are observing actions and are denoted as $\bar{a}$, $\bar{b}$, etc.). This means that we partition the set of actions of the transition system in two parts, Σ for

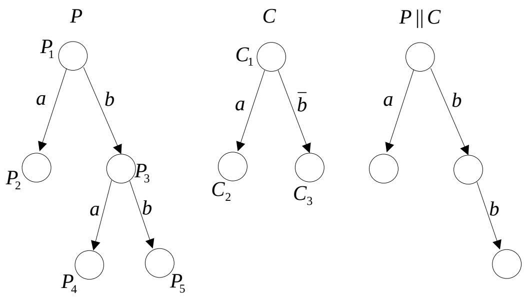

Figure 4.1: Process P controlled by C

the active transitions and $\bar{\Sigma}$ for the passive actions. Then the whole set of actions equals $\Sigma \cup \bar{\Sigma}$.

Because we want to have both blocking and non-blocking interactions, we have to make clear which interactions are blocking and which are non-blocking. Therefore, we introduce the set $A \subset \Sigma$ which contains all active actions that are involved in blocking interaction. The composition operator will first be denoted by $|A|$. We write $||$ to denote the composition operator $|A|$ in cases where A is not specified. Blocking-interaction is now expressed by the following operational rule ([Plo81]):

$$r1. \frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{a} L_2'}{L_1|A|L_2 \xrightarrow{a} L_1'|A|L_2'} (a \in A),$$

which says that a blocking-synchronization on a from joint location $L_1|A|L_2$ to $L_1'|A|L_2'$, can happen when both partners have the action a available from locations L_1 and L_2 to locations L_1' and L_2' respectively (In order to comply with the terminology used in timed and hybrid automata, we use the term *locations* to designate the *discrete states* in the transition system). Because for actions $a \in A$ rule r1 is the only rule that can bring forth an a-transition in the composed system, we get that an 'a-transition is present in the composed system' if and only if the premises of rule r1 hold. We call a blocking-synchronization also an active-active synchronization because it is a synchronization of two active transitions, i.e., of two transitions whose labels are active actions.

In Figure 4.1, where $a \in A$, we see that both the process P and the controller C have transitions labelled with a from their initial locations P_1 and C_1. This results in the (synchronized) a-transition in the composite system $P||C$. In location P_3, P can do an a-action, but because $a \in A$ and C does not have an active a-transition in location C_3, this transition is blocked by C and therefore the transition is not present in $P||C$.

Blocking as expressed in rule r1 can be used in supervisory control where a controller C blocks certain actions of the process P. Another situation where active actions must synchronize (i.e., are in A) is where two partners cooperate on a certain task. If two persons are chatting with each other, then they are cooperating on the chat-task so to say. In other words, both are chatting or both are not chatting, one cannot chat without the other (this situation will be described in the example in Section 4.1.4). For a situation where two persons P_1 and P_2 can both do a specific action (like hanging up the phone as described in Section 4.1.4) independently of the other person, then the action can be independently performed (i.e., should not be in A).

For non-blocking-interaction, we need an active a with $a \notin A$ in one partner and a passive $\bar{a}$ in the other partner. Non-blocking interaction is now expressed by

$$r2.\frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{\bar{a}} L_2'}{L_1|A|L_2 \xrightarrow{a} L_1'|A|L_2'}(a \notin A),$$

which means that L_1 executes an a, which is observed by a $\bar{a}$-transition outgoing from L_2. In Figure 4.1 we see for example that the b-transition from location P_1 in P, is observed by the $\bar{b}$-transition in C. We also need rule r2', which is the mirror rule of r2 (i.e., $L_1 \xrightarrow{\bar{a}} L_1', L_2 \xrightarrow{a} L_2'$ instead of $L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{\bar{a}} L_2'$ etc.).

If L_2 can not observe a-actions (i.e., there is no outgoing $\bar{a}$-transition), L_1 should still be able to execute a, since this execution does not depend on whether or not some other process is observing the action. This is expressed by

$$r3.\frac{L_1 \xrightarrow{a} L_1', L_2 \not\xrightarrow{\bar{a}}}{L_1|A|L_2 \xrightarrow{a} L_1'|A|L_2}(a \notin A)$$

and its mirror rule

$$r3'.\frac{L_1 \not\xrightarrow{\bar{a}}, L_2 \xrightarrow{a} L_2'}{L_1|A|L_2 \xrightarrow{a} L_1|A|L_2'}(a \notin A).$$

In Figure 4.1 we see for example that P has a b-transition from location P_3, which cannot be observed by C, but which is still present in $P||C$.

Note that because rules r2, r2', r3 and r3' are the only rules from which active transitions can be derived, it follows that if L_2 *can* observe a, then it also *will* observe a, i.e., if L_2 has a $\bar{a}$-transition, then it can not choose not to observe an a executed by the other component. This makes sense in many situations, e.g. when some system broadcasts a radio signal a and a receiver is able to receive the signal (i.e., it has a passive $\bar{a}$-transition), then we should not allow the possibility that the signal is broadcast while the receiver does not receive (i.e., does not synchronize its passive $\bar{a}$-action with the active a-action).

Negative premises in rules (as in rule r3) may in general lead to complications because of circularities (see [Gro91]), however in our rules there are no problems, as can easily be seen from the fact that $\xrightarrow{\bar{a}}$ transitions never depend on $\xrightarrow{a}$ transitions, and therefore no circularities may arise.

In the supervisory control context, observing as active/passive-synchronization means that the controller observes actions of the process. Outside the supervisory control context, this mechanism can be used for other kinds of observation (e.g. a person that observes an alarm signal as described in Section 4.1.4).

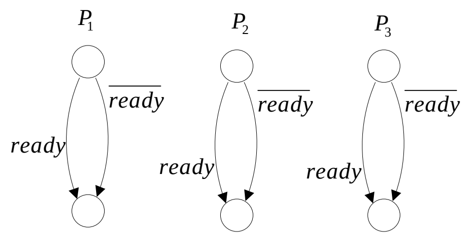

Figure 4.2: Situation where all persons can observe all others

4.1.2 Controlling the scope of observations

Passive transitions are intended to observe active transitions. This means that outside an open-systems context (i.e., when the system is closed and will not interact with other systems), passive transitions are supposed not to be present, since there is nothing to observe. Suppose we have a system that consists of a process P and a controller C, where the controller observes the actions of P. One could argue that $|A|$ should be defined such that after composition, there are no passive transitions anymore (which is the case if we only consider rules r1,r2 and r3). But that would not be suitable for a broader composition context like *modular supervisory control*: suppose we have a system with one process P and two controllers C_1 and C_2, where both C_1 and C_2 are concurrently observing P. If we define $C := C_1||C_2$ as the composed controller, then C should still observe P, in other words, C should still contain the passive transitions from C_1 and C_2 to observe P. This means that the 'real' observations happen when we compose C with P and not when we compose C_1 with C_2.

One solution one could think of is to indicate within the composition operator where the 'real' observations take place. We could use an observation set O and then $||_O$ (or together with A, $|^O_A|$) means that observations take place in this composition for the events in O. Then in the compound $(C_1||C_2)||_O P$ there are no passive transitions with labels from O anymore.

For the double-controller situation, $||_O$ seems to be a good solution. However, it seems that for modelling the following situation, we need a different solution: suppose three persons P_1, P_2 and P_3 are working on a problem. All persons start working independently on this problem. Once one of the persons found the solution, all three persons stop with the problem. This can be modelled as in Figure 4.2, where the signal *ready* is 'broadcast' by one of the persons as soon as this person solved the problem, and is then received by the others. In this situation, every P_i should be able to hear every P_j ($i \neq j$). With $||_O$ we can not express this. (In $(P_1||P_2)||_O P_3$, P_1 does not observe P_2. In $(P_1||_O P_2)||P_3$, P_3 does not observe P_1 and P_2 etc.).

Instead of using $||_O$, we use the closing operator $[\cdot]_C$. $[X]_C$ discards all passive transitions in X with labels from C. With this closing operator, we can solve both the double-controller and the 'three persons' problem. The composition rules for this operator are

$$r7.\frac{L \xrightarrow{a} L'}{[L]_C \xrightarrow{a} [L']_C}$$

$$r8.\frac{L \xrightarrow{\bar{a}} L'}{[L]_C \xrightarrow{\bar{a}} [L']_C}(\bar{a} \notin C).$$

Note that for locations L_1, L_2, etc. of transition system X, the corresponding locations in transition system $[X]_C$ are denoted by the names $[L_1]_C$, $[L_2]_C$, etc.

With $[\cdot]_C$ we can control the scope of the observations. Now, $||$ should be defined such that in $X||Y$, X can observe Y (and vice versa), but also $X||Y$ can still observe Z in $(X||Y)||Z$. This is expressed by

$$r4.\frac{L_1 \xrightarrow{\bar{a}} L_1'}{L_1|A|L_2 \xrightarrow{\bar{a}} L_1'|A|L_2}$$

and its mirror r4' which say that after the composition, every passive transition 'remains', such that the component to which this transition belongs, is still able to observe a new component which might be added to the composite system in a new composition operation. If, for example, we know that X and Y only observe each other and will (or can) not observe Z or other components, then we could specify this as $[X||Y]||Z$. ($[\cdot]$ is shorthand for $[\cdot]_{\bar{\Sigma}}$ with $\bar{\Sigma}$ the set of all passive actions).

4.1.3 Establishing multi-way synchronization

Consider the modular supervisory control situation again where we have two controllers C_1 and C_2 and a process C_3. (In this section P will get a different meaning, therefore we call the process C_3 and not P). Now suppose that C_3 has an action a which can be observed by both C_1 and C_2 (i.e., both controllers have $\bar{a}$-transitions). If we allow that both controllers can observe a concurrently (which is most natural), then we need to express the possibility of a multi-way synchronization (in this case: two passive actions and one active action). Another example (outside the supervisory control context) where we need a multi-way synchronization is the situation where an alarm signal in an office is heard (observed) by two different employees working in that office (this example is also described in Section 4.1.4).

The question now is whether there are situations we want to model that do not allow multi-way synchronizations. One such example (also considered in Section 4.1.4) is the telephone situation: a telephone in an office rings and only one of the employees may answer the call. Although both employees hear the telephone, only one may answer it (i.e., may synchronize its passive action with the active telephone signal).

We see that it is desirable to distinguish two types of passive actions: passive actions for which multi-way synchronization is allowed and passive actions for which this is not allowed. Therefore we introduce the set $P \subset \bar{\Sigma}$, which contains all passive actions for which multi-way synchronization is allowed. The composition operator will now be denoted by $|^P_A|$. Multi-way synchronization is expressed by

$$r5.\frac{L_1 \xrightarrow{\bar{a}} L_1', L_2 \xrightarrow{\bar{a}} L_2'}{L_1|_A^P|L_2 \xrightarrow{\bar{a}} L_1'|_A^P|L_2'}(\bar{a} \in P),$$

which means that two passive $\bar{a}$-transitions, one in C_1 and one in C_2, synchronize, which results in a $\bar{a}$-transition for the composite system $C_1||C_2$. This synchronized passive transition can observe an a in C_3, which results in a new (multi-way) synchronized transition in $(C_1||C_2)||C_3$ which then expresses that C_1 and C_2 concurrently observe C_3.

If $a \in P$ and C_1 can observe a, but C_2 cannot observe a, then in $(C_1||C_2)||C_3$, C_1 should synchronize its passive $\bar{a}$ with the active a of C_3, while C_2 idles. This situation is expressed by

$$r6.\frac{L_1 \xrightarrow{\bar{a}} L_1', L_2 \not\xrightarrow{\bar{a}}}{L_1|_A^P|L_2 \xrightarrow{\bar{a}} L_1'|_A^P|L_2}(\bar{a} \in P)$$

and its mirror rule r6'. In the new situation (where we have introduced the set P), we can see that rule r4 (which expresses that passive transitions should interleave) only applies for passive actions not in P. Therefore r4 should be changed to

$$r4.\frac{L_1 \xrightarrow{\bar{a}} L_1'}{L_1|_A^P|L_2 \xrightarrow{\bar{a}} L_1'|_A^P|L_2}(\bar{a} \notin P).$$

With the set P as part of the operator, rules r1,r2 and r3 should be changed to

$$r1.\frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{a} L_2'}{L_1|_A^P|L_2 \xrightarrow{a} L_1'|_A^P|L_2'}(a \in A),$$

$$r2.\frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{\bar{a}} L_2'}{L_1|_A^P|L_2 \xrightarrow{a} L_1'|_A^P|L_2'}(a \notin A),$$

$$r3.\frac{L_1 \xrightarrow{a} L_1', L_2 \not\xrightarrow{\bar{a}}}{L_1|_A^P|L_2 \xrightarrow{a} L_1'|_A^P|L_2}(a \notin A).$$

Now rules r1 till r6 (and the mirror rules r2',r3',r4' and r6') form the structural operational semantics for $|_A^P|$ and rule r7 and r8 form the structural operational semantics for $[\cdot]_C$.

Note that the synchronization-mechanisms for active transitions and passive transitions are different. If an active action a is synchronizing (i.e., $a \in A$), then a component can execute the action only when the other component in the composition can also execute the action (and vice versa). If a passive action $\bar{a}$ is synchronizing (i.e., $\bar{a} \in P$), then if both components can execute the action, they have to synchronize. However, if only one component can execute the action, the action can still be executed without the other component synchronizing with it. This is expressed in rule r6. For active-active synchronization we do not have an equivalent rule as r6. Because of this difference between the synchronization-mechanisms, we need to use different composition rules for both the active and the passive actions (i.e., we can not combine active and passive synchronization in the same rules).

The operators $|^P_A|$ and $[\cdot]_C$ are now formally defined as follows.

Definition 4.3. Let $X = (S_1, \Sigma \cup \bar{\Sigma}, \mathcal{T}_1)$ and $Y = (S_2, \Sigma \cup \bar{\Sigma}, \mathcal{T}_2)$ be transition systems. Then $X|^P_A|Y$ is defined as the transition system $(S_1 \times S_2, \Sigma \cup \bar{\Sigma}, \mathcal{T})$, where $\mathcal{T}$ is the least relation satisfying the rules r1,r2,r2',r3,r3',r4,r4',r5,r6 and r6' below. $[X]_C$ is defined as the transition system $([S_1]_C, \Sigma \cup \bar{\Sigma}, \tilde{\mathcal{T}})$, where $[S_1]_C := \{[s]_C | s \in S_1\}$ and $\tilde{\mathcal{T}}$ is the least relation satisfying the rules r7 and r8 below.

$$r1. \frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{a} L_2'}{L_1|^P_A|L_2 \xrightarrow{a} L_1'|^P_A|L_2'} (a \in A),$$

$$r2. \frac{L_1 \xrightarrow{a} L_1', L_2 \xrightarrow{\bar{a}} L_2'}{L_1|^P_A|L_2 \xrightarrow{a} L_1'|^P_A|L_2'} (a \notin A), \quad r2'. \frac{L_1 \xrightarrow{\bar{a}} L_1', L_2 \xrightarrow{a} L_2'}{L_1|^P_A|L_2 \xrightarrow{a} L_1'|^P_A|L_2'} (a \notin A),$$

$$r3. \frac{L_1 \xrightarrow{a} L_1', L_2 \not\xrightarrow{\bar{a}}}{L_1|^P_A|L_2 \xrightarrow{a} L_1'|^P_A|L_2} (a \notin A), \quad r3'. \frac{L_1 \not\xrightarrow{\bar{a}}, L_2 \xrightarrow{a} L_2'}{L_1|^P_A|L_2 \xrightarrow{a} L_1|^P_A|L_2'} (a \notin A),$$

$$r4. \frac{L_1 \xrightarrow{\bar{a}} L_1'}{L_1|^P_A|L_2 \xrightarrow{\bar{a}} L_1'|^P_A|L_2} (\bar{a} \notin P), \quad r4'. \frac{L_2 \xrightarrow{\bar{a}} L_2'}{L_1|^P_A|L_2 \xrightarrow{\bar{a}} L_1|^P_A|L_2'} (\bar{a} \notin P),$$

$$r5. \frac{L_1 \xrightarrow{\bar{a}} L_1', L_2 \xrightarrow{\bar{a}} L_2'}{L_1|^P_A|L_2 \xrightarrow{\bar{a}} L_1'|^P_A|L_2'} (\bar{a} \in P),$$

$$r6. \frac{L_1 \xrightarrow{\bar{a}} L_1', L_2 \not\xrightarrow{\bar{a}}}{L_1|^P_A|L_2 \xrightarrow{\bar{a}} L_1'|^P_A|L_2} (\bar{a} \in P), \quad r6'. \frac{L_1 \not\xrightarrow{\bar{a}}, L_2 \xrightarrow{\bar{a}} L_2'}{L_1|^P_A|L_2 \xrightarrow{\bar{a}} L_1|^P_A|L_2'} (\bar{a} \in P),$$

$$r7. \frac{L \xrightarrow{a} L'}{[L]_C \xrightarrow{a} [L']_C},$$

$$r8. \frac{L \xrightarrow{\bar{a}} L'}{[L]_C \xrightarrow{\bar{a}} [L']_C} (\bar{a} \notin C).$$

4.1.4 Example

In Figure 4.3 we see an example where we find back all interaction structures we described before: non-synchronizing active transitions, synchronizing active transitions, non-synchronizing passive transitions, synchronizing passive transitions and active-passive synchronization.

The example of Figure 4.3 concerns an office O with two employees E_1 and E_2. In the office there are two sources which can produce a signal: a telephone

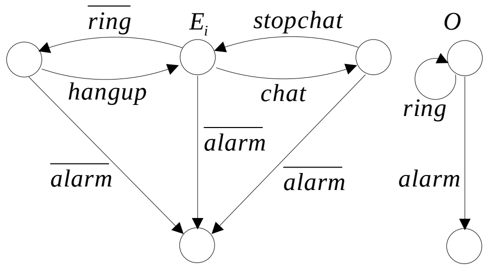

Figure 4.3: Example with different kinds of interaction

and an alarm. The telephone rings when somebody calls the office, the alarm fires when there is danger, which means that the employees should leave the office when they hear the alarm. Both the telephone and the alarm execute their signals independently from the employees. Therefore they are modelled as active transitions labelled *ring* and *alarm* respectively (see Figure 4.3). If the telephone rings, O makes a self-loop which means that O stays in the same location. This location has the meaning of 'normal-working-conditions'. If the alarm goes, O jumps to a second location which has as meaning 'dangerous-working-conditions'.

E_1 and E_2 have the same automaton-structure. From E_1 (or E_2), the employee can exhibit three different actions: he can chat with his fellow employee, he can pick up the phone and he can leave the office. Leaving the office only happens when the alarm goes off. This action is modelled as a passive transition which synchronizes with the independent active alarm-transition in O. We see that from every location, the employee can react on the alarm. When the phone rings, the employee can pick up the phone by synchronizing its passive *ring* event with the active *ring* from O. The employee hangs up with an active *hangup* event. The employee can start a chat with his office-mate via the active *chat* event. This should synchronize with an active *chat* event of the office-mate (both should be willing or able to chat). The chat will be ended with an active-active synchronization of *stopchat*. For all three processes E_1, E_2 and O we define the action sets as $\Sigma = \{chat, stopchat, ring, alarm\}$ and its mirror set $\bar{\Sigma} = \{\overline{chat}, \overline{stopchat}, \overline{ring}, \overline{alarm}\}$.

From the model we see that if the employees are chatting, they first have to end the chat before one of them can pick up the phone. This means that they will probably miss the first *ring* signal, but can maybe interact on the second *ring* signal (the phone might give multiple *ring* signals when somebody calls). Also, if one employee is phoning, he first has to hang up before a chat can be started.

Where do we see the different kinds of interaction? The passive *ring* transitions of E_1 and E_2 should be non-synchronizing since only one passive transition is allowed to synchronize with the active *ring* transition in O. The passive *alarm* transitions in E_1 and E_2 should synchronize because both employees react synchronously on the alarm. The active *hangup* transitions in E_1 and E_2 should be non-synchronizing (only one employee hangs up). The active *chat* and *stopchat* transitions in E_1 and E_2 should synchronize (both employees chat).

From the above follows that for the composition $E_1|_A^P|E_2$ we get $A = \{chat, stopchat\}$ and $P = \{\overline{alarm}\}$. For the composition $(E_1|_{chat,stopchat}^{\overline{alarm}}|E_2)|_A^P|O$ we get $A = \emptyset$ and $P = \bar{\Sigma}$. The total specification is then

$$(E_1|_{chat,stopchat}^{\overline{alarm}}|E_2)|_{\emptyset}^{\bar{\Sigma}}|O. \tag{4.1}$$

If (4.1) is the system that we want to analyze, then we could close down all observation channels (i.e., the passive transitions) with the $[\cdot]$ operator. Now suppose that (4.1) is only one chamber of a bigger office consisting of multiple chambers. Then this chamber is only one unit and lets call it U_1. The other units then are $U_2 = (E_1'|_{chat,stopchat}^{\overline{alarm}}|E_2')|_{\emptyset}^{\bar{\Sigma}}|O'$, $U_3 = (E_1''|_{chat,stopchat}^{\overline{alarm}}|E_2'')|_{\emptyset}^{\bar{\Sigma}}|O''$ etc., where $E_i = E_i' = E_i''$, $O = O' = O''$ etc. The telephones in each office are local. In other words, if a telephone rings in one office, only the employees in that office hear the phone and can answer it. The alarm however is global. If there is danger in the building, the alarm goes synchronously in all offices. To specify that the phones are local, we use $[\cdot]$ and get $[U_i]_{ring}$. To specify that the alarm is global, we use *alarm* as a synchronization event in the composition of the units. The total composition of the whole building is then

$$U_1'|_{alarm}|U_2'|_{alarm}|U_3'|_{alarm}|\cdots,$$

where $U_i' = [U_i]_{ring}$.

4.1.5 Supervisory control with $|_A^P|$

In this section we want to take a closer look at supervisory control. We will briefly describe the main concepts of supervisory control and thereafter we compare how specification of control systems can be done in our active/passive framework to how it can be done in a framework that only supports blocking-interaction.

In the supervisory control paradigm ([RW89, CL99]), the actions of a process can be observable or unobservable and they can be controllable or uncontrollable. Observable actions can be observed by the controller (and unobservable actions can not). Controllable actions can be controlled by the controller (and uncontrollable actions can not). This means that the controller can block these actions, i.e., can prevent them from happening.

If we specify the control system within a transition system framework, then the process and the controller are modelled as two separate transition systems which can interact. The process executing an action is then modelled as a transition (labelled with this action) from one process-location to another process-location. The controller observing an action is then modelled as a transition in the controller that synchronizes with the to-be-observed transition in the process. Whether the observing transition of the controller is active or passive, depends, as we will see, on the controllability of the action of the to-be-observed transition of the process.

Consider the process X in Figure 4.4 with a controllable/observable and b uncontrollable/observable. We want to control this process such that the behavior of X is restricted to $X||C$ of Figure 4.4.

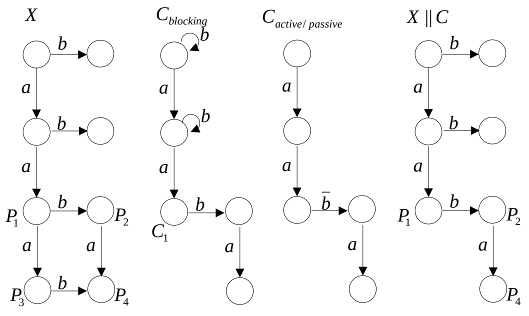

Figure 4.4: Control in only-blocking framework and in active/passive framework

First let us see how this can be done in a framework that only supports blocking-interaction. Since a and b are both observable, we mark them as synchronization actions (i.e., in the composition operation these actions must synchronize while non-synchronization actions will be interleaving). The controller that does the job is $C_{blocking}$ of Figure 4.4. We see that $C_{blocking}$ has two self-loops on action b. We need these self-loops because otherwise the two upper b-transitions of X are blocked and that is not what we want according to the specification $X||C$. We see that the transition $P_1 \xrightarrow{a} P_3$ of X is blocked by $C_{blocking}$ because of the absence of an a-transition in $C_{blocking}$ at location C_1. In the supervisory control framework, which only supports blocking-interaction (see [CL99]), the controller that implements the specification $X||C$ is expressed by $C_{blocking}$.

For the active/passive framework, $C_{active/passive}$ of Figure 4.4 does the job. Because a is controllable (i.e., blockable), $a \in A$ and because b is not controllable, $b \notin A$. Blocking the a transition from P_1 to P_3 is done in the same way as in a framework that only supports blocking-interaction. The difference however is, that here we do not need the self-loops on b. Concerning the blocking of the $P_1 \overset{a}{\to} P_3$ transition, the controller needs the information of X jumping from P_1 to P_2, because before this jump the action a should be blocked while after this jump the action a should be allowed. Therefore, in order to obtain this information, the controller should observe the $P_1 \overset{b}{\to} P_2$ transition of X. This is done via the passive $\bar{b}$-transition of $C_{active/passive}$.

We could say that in the active/passive framework, the controller will observe only (by means of a passive transition) when it needs the information. In Figure 4.4, $C_{active/passive}$ observes $P_1 \xrightarrow{b} P_2$, but does not observe the upper b-transitions of P, because P may execute them without the controller 'knowing' it. In a framework that only supports blocking-interaction, the controller must also synchronize on the upper b-transitions, because otherwise they will be blocked. We think that this is an important advantage of using active/passive transitions: for uncontrollable/observable actions, transitions are only needed where observations are needed. If there is only blocking-interaction, transitions in the controller are needed everywhere the process executes an uncontrollable/observable action (otherwise they will be blocked and that is not allowed).

We resume. For specifying supervisory control systems in the active/passive framework with operator $|^{P}_{A}|$, we get that A forms the set of controllable actions. The set P does not have any relevance in the context of a single controller with a single process, because P prescribes which passive actions should be interleaving, expressing therewith which actions can be observed by only one component and which can be observed by multiple components. P can have relevance in the context where we have a process with two controllers. Then $\bar{\sigma} \in P$ expresses that only one controller may observe a σ transition of the process, i.e., it expresses that it is not allowed that both controllers observe a σ-transition (at the same time).

Whether an action is observable or not is, in the active/passive framework, coded in the absence or presence of transitions in the controller: an observable/ controllable action σ is observed by the controller via an active σ-transition. An observable/uncontrollable action σ is observed by the controller via a passive $\bar{\sigma}$-transition. If an action σ is unobservable/controllable, then σ-transitions of the controller are always self-loops. By a self-loop the controller allows (i.e., does not block) the action, while it is not observed because the location of the controller does not change when a self-loop transition is executed. If an action is unobservable/uncontrollable, then the controller does not have any σ and $\bar{\sigma}$ transitions.

4.1.6 Commutativity and associativity of $|^{P}_{A}|$

Theorem 4.4. *$|^{P}_{A}|$ is commutative for all A and P. $|^{P}_{A}|$ is associative if and only if for all events a we have: $a \notin A \Rightarrow \bar{a} \in P$.*

Proof. Because we also have the symmetric rules 2',3',4' and 6' of 2,3,4 and 6, $|^{P}_{A}|$ is clearly commutative. For associativity, we first consider the case where P is full (the set of all labels). Independent of A, there are seven cases that bring forth a passive transition in the composition of three components (case 1-7 below). If $a \notin A$, there are 12 cases (case 8-19) that bring forth an active transition in the composition of three components. For the case that $a \in A$, there is only one case (case 20) where three components bring forth an active transition. To show how the twenty cases below should be read, we explain what case 1 means: if we compose X, Y and Z, and X and Y have no $\bar{a}$-transitions from locations l_X and l_Y and Z does have a $\bar{a}$-transition from location l_Z to l'_Z, then in the composition $(X|^{P}_{A}|Y)|^{P}_{A}|Z$ there is a $\bar{a}$-transition from joint location l_X, l_Y, l_Z to joint location l_X, l_Y, l'_Z (this follows from applying rule r6') and also in the composition $X|^{P}_{A}|(Y|^{P}_{A}|Z)$ there is a $\bar{a}$-transition from joint location l_X, l_Y, l_Z to joint location l_X, l_Y, l'_Z (this follows from applying rule r6' twice), which shows associativity for case 1. The symbol 0 in $\overset{0,r6'}{\Longrightarrow}$ means that no rule should be applied in this case for the in-bracket composition.

1. $(\not\xrightarrow{\bar{a}} |^{P}_{A}| \not\xrightarrow{\bar{a}})|^{P}_{A}| \xrightarrow{\bar{a}} \quad \overset{0,\mathrm{r6}'}{\Longrightarrow} \quad \xrightarrow{\bar{a}} \quad \overset{\mathrm{r6}',\mathrm{r6}'}{\Longleftarrow} \quad \not\xrightarrow{\bar{a}} |^{P}_{A}|(\not\xrightarrow{\bar{a}} |^{P}_{A}| \xrightarrow{\bar{a}})$
2. $(\not\xrightarrow{\bar{a}} |^{P}_{A}| \xrightarrow{\bar{a}})|^{P}_{A}| \not\xrightarrow{\bar{a}} \quad \overset{\mathrm{r6}',\mathrm{r6}}{\Longrightarrow} \quad \xrightarrow{\bar{a}} \quad \overset{\mathrm{r6},\mathrm{r6}'}{\Longleftarrow} \quad \not\xrightarrow{\bar{a}} |^{P}_{A}|(\xrightarrow{\bar{a}} |^{P}_{A}| \not\xrightarrow{\bar{a}})$
3. $(\xrightarrow{\bar{a}} |^{P}_{A}| \not\xrightarrow{\bar{a}})|^{P}_{A}| \not\xrightarrow{\bar{a}} \quad \overset{\mathrm{r6},\mathrm{r6}}{\Longrightarrow} \quad \xrightarrow{\bar{a}} \quad \overset{0,\mathrm{r6}}{\Longleftarrow} \quad \xrightarrow{\bar{a}} |^{P}_{A}|(\not\xrightarrow{\bar{a}} |^{P}_{A}| \not\xrightarrow{\bar{a}})$
4. $(\not\xrightarrow{\bar{a}} |^{P}_{A}| \xrightarrow{\bar{a}})|^{P}_{A}| \xrightarrow{\bar{a}} \quad \overset{\mathrm{r6}',\mathrm{r5}}{\Longrightarrow} \quad \xrightarrow{\bar{a}} \quad \overset{\mathrm{r5},\mathrm{r6}'}{\Longleftarrow} \quad \not\xrightarrow{\bar{a}} |^{P}_{A}|(\xrightarrow{\bar{a}} |^{P}_{A}| \xrightarrow{\bar{a}})$

5. $(\xrightarrow{\bar a} |^P_A| \not\xrightarrow{\bar a})|^P_A| \xrightarrow{\bar a} \quad \overset{\text{r6,r5}}{\Longrightarrow} \quad \xrightarrow{\bar a} \quad \overset{\text{r6}',\text{r5}}{\Longleftarrow} \quad \xrightarrow{\bar a} |^P_A|(\not\xrightarrow{\bar a} |^P_A| \xrightarrow{\bar a})$
6. $(\xrightarrow{\bar a} |^P_A| \xrightarrow{\bar a})|^P_A| \not\xrightarrow{\bar a} \quad \overset{\text{r5,r6}}{\Longrightarrow} \quad \xrightarrow{\bar a} \quad \overset{\text{r5,r6}}{\Longleftarrow} \quad \xrightarrow{\bar a} |^P_A|(\xrightarrow{\bar a} |^P_A| \not\xrightarrow{\bar a})$
7. $(\xrightarrow{\bar a} |^P_A| \xrightarrow{\bar a})|^P_A| \xrightarrow{\bar a} \quad \overset{\text{r5,r5}}{\Longrightarrow} \quad \xrightarrow{\bar a} \quad \overset{\text{r5,r5}}{\Longleftarrow} \quad \xrightarrow{\bar a} |^P_A|(\xrightarrow{\bar a} |^P_A| \xrightarrow{\bar a})$

$a \notin A$:

8. $(\not\xrightarrow{\bar a} |^P_A| \not\xrightarrow{\bar a})|^P_A| \xrightarrow{a} \quad \overset{\text{0,r3}'}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r3}',\text{r3}'}{\Longleftarrow} \quad \not\xrightarrow{\bar a} |^P_A|(\not\xrightarrow{\bar a} |^P_A| \xrightarrow{a})$
9. $(\not\xrightarrow{\bar a} |^P_A| \xrightarrow{a})|^P_A| \not\xrightarrow{\bar a} \quad \overset{\text{r3}',\text{r3}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r3,r3}'}{\Longleftarrow} \quad \not\xrightarrow{\bar a} |^P_A|(\xrightarrow{a} |^P_A| \not\xrightarrow{\bar a})$
10. $(\xrightarrow{a} |^P_A| \not\xrightarrow{\bar a})|^P_A| \not\xrightarrow{\bar a} \quad \overset{\text{r3,r3}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{0,r3}}{\Longleftarrow} \quad \xrightarrow{a} |^P_A|(\not\xrightarrow{\bar a} |^P_A| \not\xrightarrow{\bar a})$
11. $(\not\xrightarrow{\bar a} |^P_A| \xrightarrow{a})|^P_A| \xrightarrow{\bar a} \quad \overset{\text{r3}',\text{r2}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r2,r3}'}{\Longleftarrow} \quad \not\xrightarrow{\bar a} |^P_A|(\xrightarrow{a} |^P_A| \xrightarrow{\bar a})$
12. $(\not\xrightarrow{\bar a} |^P_A| \xrightarrow{\bar a})|^P_A| \xrightarrow{a} \quad \overset{\text{r6}',\text{r2}'}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r2}',\text{r3}'}{\Longleftarrow} \quad \not\xrightarrow{\bar a} |^P_A|(\xrightarrow{\bar a} |^P_A| \xrightarrow{a})$
13. $(\xrightarrow{a} |^P_A| \not\xrightarrow{\bar a})|^P_A| \xrightarrow{\bar a} \quad \overset{\text{r3,r2}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r6}',\text{r2}}{\Longleftarrow} \quad \xrightarrow{a} |^P_A|(\not\xrightarrow{\bar a} |^P_A| \xrightarrow{\bar a})$
14. $(\xrightarrow{\bar a} |^P_A| \not\xrightarrow{\bar a})|^P_A| \xrightarrow{a} \quad \overset{\text{r6,r2}'}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r3}',\text{r2}'}{\Longleftarrow} \quad \xrightarrow{\bar a} |^P_A|(\not\xrightarrow{\bar a} |^P_A| \xrightarrow{a})$
15. $(\xrightarrow{a} |^P_A| \xrightarrow{\bar a})|^P_A| \not\xrightarrow{\bar a} \quad \overset{\text{r2,r3}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r6,r2}}{\Longleftarrow} \quad \xrightarrow{a} |^P_A|(\xrightarrow{\bar a} |^P_A| \not\xrightarrow{\bar a})$
16. $(\xrightarrow{\bar a} |^P_A| \xrightarrow{a})|^P_A| \not\xrightarrow{\bar a} \quad \overset{\text{r2}',\text{r3}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r3,r2}'}{\Longleftarrow} \quad \xrightarrow{\bar a} |^P_A|(\xrightarrow{a} |^P_A| \not\xrightarrow{\bar a})$
17. $(\xrightarrow{\bar a} |^P_A| \xrightarrow{\bar a})|^P_A| \xrightarrow{a} \quad \overset{\text{r5,r2}'}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r2}',\text{r2}'}{\Longleftarrow} \quad \xrightarrow{\bar a} |^P_A|(\xrightarrow{\bar a} |^P_A| \xrightarrow{a})$
18. $(\xrightarrow{\bar a} |^P_A| \xrightarrow{a})|^P_A| \xrightarrow{\bar a} \quad \overset{\text{r2}',\text{r2}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r2,r2}'}{\Longleftarrow} \quad \xrightarrow{\bar a} |^P_A|(\xrightarrow{a} |^P_A| \xrightarrow{\bar a})$
19. $(\xrightarrow{a} |^P_A| \xrightarrow{\bar a})|^P_A| \xrightarrow{\bar a} \quad \overset{\text{r2,r2}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r5,r2}}{\Longleftarrow} \quad \xrightarrow{a} |^P_A|(\xrightarrow{\bar a} |^P_A| \xrightarrow{\bar a})$.

$a \in A$:

20. $(\xrightarrow{a} |^P_A| \xrightarrow{a})|^P_A| \xrightarrow{a} \quad \overset{\text{r1,r1}}{\Longrightarrow} \quad \xrightarrow{a} \quad \overset{\text{r1,r1}}{\Longleftarrow} \quad \xrightarrow{a} |^P_A|(\xrightarrow{a} |^P_A| \xrightarrow{a})$.

Now suppose we have $a \in A$ and $\bar a \notin P$. The only possibility to get an active transition with three components comes down to case 20 (already treated above). Passive transitions can now only be generated via composition rule 4, which says that all passive transitions are interleaving (i.e., they do not synchronize at all). It can now easily be seen that transitions $L_1 \xrightarrow{\bar a} L_1'$, $L_2 \xrightarrow{\bar a} L_2'$, $L_3 \xrightarrow{\bar a} L_3'$ (with L_i,L_i' locations of component i in the composition) bring forth the transitions $(L_1|^P_A|L_2)|^P_A|L_3 \xrightarrow{\bar a} (L_1'|^P_A|L_2)|^P_A|L_3$, $(L_1|^P_A|L_2)|^P_A|L_3 \xrightarrow{\bar a} (L_1|^P_A|L_2')|^P_A|L_3$ and $(L_1|^P_A|L_2)|^P_A|L_3 \xrightarrow{\bar a} (L_1|^P_A|L_2)|^P_A|L_3'$ but also the transitions $L_1|^P_A|(L_2|^P_A|L_3) \xrightarrow{\bar a} L_1'|^P_A|(L_2|^P_A|L_3)$, $L_1|^P_A|(L_2|^P_A|L_3) \xrightarrow{\bar a} L_1|^P_A|(L_2'|^P_A|L_3)$ and $L_1|^P_A|(L_2|^P_A|L_3) \xrightarrow{\bar a} L_1|^P_A|(L_2|^P_A|L_3')$, from which associativity follows.

Now suppose we have $a \notin A$ and $\bar a \notin P$. Then the following is a counterexample for associativity:

$$(L_1 \xrightarrow{a} L_1'|^P_A|L_2 \xrightarrow{\bar a} L_2')|^P_A|L_3 \xrightarrow{\bar a} L_3' \quad \overset{\text{r2,r2}}{\Longrightarrow} \quad (L_1|^P_A|L_2)|^P_A|L_3 \xrightarrow{a} (L_1'|^P_A|L_2')|^P_A|L_3',$$

$$L_1 \xrightarrow{a} L_1'|^P_A|(L_2 \xrightarrow{\bar a} L_2'|^P_A|L_3 \xrightarrow{\bar a} L_3') \quad \Longrightarrow \quad (L_1|^P_A|L_2)|^P_A|L_3 \not\xrightarrow{a} (L_1'|^P_A|L_2')|^P_A|L_3'.$$

We conclude that $|^P_A|$ is associative if and only if we have: $a \notin A \Rightarrow \bar a \in P$. $\Box$

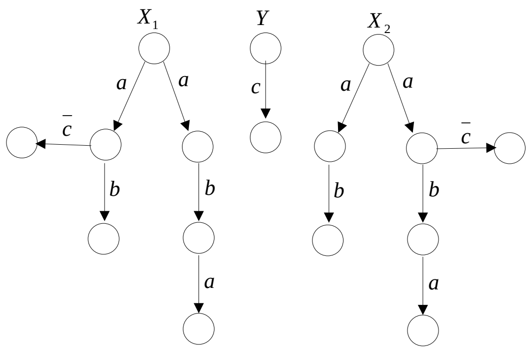

Figure 4.5: Processes X_1, X_2 and Y

If $|^P_{A_1}|$ and $|^P_{A_2}|$ are associative operators, we have $(X|^P_{A_i}|Y)|^P_{A_i}|Z = X|^P_{A_i}|(Y|^P_{A_i}|Z)$ for $i = 1, 2$, and therefore we could write $X|^P_{A_i}|Y|^P_{A_i}|Z$ instead. Note, however, that in general $(X|^P_{A_1}|Y)|^P_{A_2}|Z \neq X|^P_{A_1}|(Y|^P_{A_2}|Z)$ as in general $(X|A_1|Y)|A_2|Z \neq X|A_1|(Y|A_2|Z)$ in for example the specification languages CSP and LOTOS.

4.2 Trace semantics

We give a trace semantics for the operator $|^P_A|$. This operator is used in a context where we have active actions (like a) and passive actions (like $\bar{a}$). A trace of a transition system could then be $ab\bar{a}$ for example. However, this notion of trace is not strong enough to give a trace semantics for composition (i.e., the sets of traces of the components are not enough to determine the set of traces of the composition).

See for example Figure 4.5. There the set of traces of X_1 and X_2 are both equal to $\{\epsilon, a, ab, aba, a\bar{c}\}$. The set of traces of Y equals $\{\epsilon, c\}$. If we could determine the trace set of a composite system from the trace sets of the components, then $X_1||Y$ and $X_2||Y$ should have the same trace set (because X_1 and X_2 have the same trace set). But this is not the case, since $acba$ is a trace of $X_1||Y$ while it is not a trace of $X_2||Y$. Therefore this notion of traces is not strong enough for a compositional trace semantics of $|^P_A|$.

Another option for a trace semantics would be to look at traces of the form σP with $\sigma \in (\Sigma \cup \bar{\Sigma})^*$ and $P \subset \bar{\Sigma}$, where σP means that the process can execute the trace σ such that it ends up in a state where the set of passive events that are enabled equals P. This can be seen as an analogy of refusal traces in [Hen88]. However, this notion of trace is also not strong enough because the processes X_1 and X_2 in Figure 4.5 have the same traces of the form σP (which are $\epsilon\emptyset$, $a\{\bar{c}\}$, $a\emptyset$,$ab\emptyset$ and $aba\emptyset$) while $X_1||Y$ and $X_2||Y$ have different traces.

We now introduce a trace concept called *pie-traces* (which stands for passive-information-extended-traces). We will see that this notion of trace is indeed strong enough to give a trace semantics for $|^P_A|$. A pie-trace in this semantics looks like

$$P_1\alpha_1P_2\alpha_2\cdots P_n\alpha_n,$$

where $\alpha_i \in \Sigma\cup\bar{\Sigma}$ and $P_i \subset \bar{\Sigma}$. We could say that part of the tree-structure (as far as it concerns the passive actions) is contained in these pie-traces. $P_1\alpha_1P_2\alpha_2\cdots P_n\alpha_n$ means that within the transition system, there exists an execution of trace $\alpha_1\cdots\alpha_n$ such that at the state in the transition system where α_i is to be executed there are outgoing passive transitions for each $\bar{\alpha}\in P_i$ $(i = 1\cdots n)$ and there are no outgoing passive transitions for α if $\alpha\notin P_i$.

The processes in figure 4.5 have the following pie-traces: $Tr_{pie}(X_1) = \{\epsilon, a[\bar{c}]b, aba, ab, a\}$. Here $a[\bar{c}]b$ means P_1aP_2b with $P_1 = \emptyset$, $P_2 = \{\bar{c}\}$ etc. $Tr_{pie}(X_2) = \{\epsilon, a[\bar{c}]ba, ab, a[\bar{c}]b, a\}$ and $Tr_{pie}(Y) = \{\epsilon, c\}$. With this notion of trace semantics, X_1 and X_2 have different semantics, thus they are not equivalent with respect to this notion of trace semantics. We now show that our notion of trace semantics is strong enough for a trace semantics of the composition operator $|^P_A|$:

Let σ_1 and σ_2 be pie-traces. Then we define $\sigma_1|^P_A|\sigma_2$, which will turn out to be the set of interleavings of σ_1 and σ_2 with respect to $|^P_A|$, as follows:

- $\epsilon|^P_A|\epsilon := \{\epsilon\}$
- $(R_1\alpha_1\sigma_1')|^P_A|\epsilon := \{\epsilon\}\cup S_1$, where $S_1 := R_1\alpha_1(\sigma_1'|^P_A|\epsilon)$ if ($\alpha_1\in\Sigma$ and $\alpha_1\notin A$) or $\alpha_1\in\bar{\Sigma}$ else $S_1 := \emptyset$.
- $\epsilon|^P_A|(R_2\alpha_2\sigma_2') := \{\epsilon\}\cup S_1$, where $S_1 := R_2\alpha_2(\epsilon|^P_A|\sigma_2')$ if ($\alpha_2\in\Sigma$ and $\alpha_2\notin A$) or $\alpha_2\in\bar{\Sigma}$ else $S_1 := \emptyset$.
- $(R_1\alpha_1\sigma_1')|^P_A|(R_2\alpha_2\sigma_2') := \{\epsilon\}\cup S_1\cup S_2\cup S_3\cup S_4$, where
  $S_1 := (R_1\cup R_2)\alpha_1(\sigma_1'|^P_A|\sigma_2)$ if one of the cases r3,r4 or r6 is true, else $S_1 := \emptyset$,
  $S_2 := (R_1\cup R_2)\alpha_1(\sigma_1'|^P_A|\sigma_2')$ if one of the cases r1,r2 or r5 is true, else $S_2 := \emptyset$,
  $S_3 := (R_1\cup R_2)\alpha_2(\sigma_1|^P_A|\sigma_2')$ if one of the cases r3',r4' or r6' is true, else $S_3 := \emptyset$,
  $S_4 := (R_1\cup R_2)\alpha_2(\sigma_1'|^P_A|\sigma_2')$ if case r2' is true, else $S_4 := \emptyset$.

  Cases:
  r1: $\alpha_1 = \alpha_2\in\Sigma$ and $\alpha_1\in A$
  r2: $\alpha_1\in\Sigma$ and $\alpha_2 = \bar{\alpha}_1$ and $\alpha_1\notin A$
  r3: $\alpha_1\in\Sigma$ and $\bar{\alpha}_1\notin R_2$ and $\alpha_1\notin A$
  r4: $\alpha_1\in\bar{\Sigma}$ and $\alpha_1\notin P$
  r5: $\alpha_1 = \alpha_2\in\bar{\Sigma}$ and $\alpha_1\in P$
  r6: $\alpha_1\in\bar{\Sigma}$ and $\alpha_1\notin R_2$ and $\alpha_1\in P$
  r2': $\alpha_2\in\Sigma$ and $\alpha_1 = \bar{\alpha}_2$ and $\alpha_2\notin A$
  r3': $\alpha_2\in\Sigma$ and $\bar{\alpha}_2\notin R_1$ and $\alpha_2\notin A$
  r4': $\alpha_2\in\bar{\Sigma}$ and $\alpha_2\notin P$
  r6': $\alpha_2\in\bar{\Sigma}$ and $\alpha_2\notin R_1$ and $\alpha_2\in P$

Theorem 4.5. *σ is a pie-trace of $X|^P_A|Y$ if and only if there exist pie-traces σ_x and σ_y of X and Y respectively such that $\sigma\in\sigma_x|^P_A|\sigma_y$.*

Proof. It can be seen that the cases r1 till r6 and r2',r3',r4' and r6' correspond to the composition rules r1 till r6, r2',r3',r4' and r6'. This correspondence is such that when case r1 is true (from the initial states), then composition rule r1 can be applied in the composition and when case r2 is true then composition rule r2 can be applied, etc. Now it is easy to check that $\sigma_x|^P_A|\sigma_y$ is exactly the set of all interleavings (including synchronizations) of σ_x and σ_y that are accepted by the composition rules of $|^P_A|$, from which the result follows. $\square$

Similar to the definition of refusals in [Hen88], we can now define pie-refusal traces of the form σX, where σ is a pie-trace and X a set of active actions that can be refused after σ. In this way a semantics is defined that reduces to the standard testing semantics in the absence of passive actions.

4.3 Composition of NTSs

In this chapter we used labelled transition systems to specify and compose processes. No stochastics are involved for labelled transition systems. In this section we show that active/passive composition, as defined in this chapter with the $|^P_A|$ operator can be generalized to the context of the stochastic NTS model. In the NTS case we then have the following operational rules for the operator $|^P_A|$.

$$r1.\frac{\xi_1 \xrightarrow{a} m_1, \xi_2 \xrightarrow{a} m_2}{(\xi_1,\xi_2) \xrightarrow{a} m_1 \times m_2}(a \in A),$$

$$r2.\frac{\xi_1 \xrightarrow{a} m_1, \xi_2 \xrightarrow{\bar{a}} m_2}{(\xi_1,\xi_2) \xrightarrow{a} m_1 \times m_2}(a \notin A), \quad r2'.\frac{\xi_1 \xrightarrow{\bar{a}} m_1, \xi_2 \xrightarrow{a} m_2}{(\xi_1,\xi_2) \xrightarrow{a} m_1 \times m_2}(a \notin A),$$

$$r3.\frac{\xi_1 \xrightarrow{a} m_1, \xi_2 \not\xrightarrow{\bar{a}}}{(\xi_1,\xi_2) \xrightarrow{a} m_1 \times Id(\xi_2)}(a \notin A), \quad r3'.\frac{\xi_1 \not\xrightarrow{\bar{a}}, \xi_2 \xrightarrow{a} m_2}{(\xi_1,\xi_2) \xrightarrow{a} Id(\xi_1) \times m_2}(a \notin A),$$

$$r4.\frac{\xi_1 \xrightarrow{\bar{a}} m_1}{(\xi_1,\xi_2) \xrightarrow{\bar{a}} m_1 \times Id(\xi_2)}(\bar{a} \notin P), \quad r4'.\frac{\xi_2 \xrightarrow{\bar{a}} m_2}{(\xi_1,\xi_2) \xrightarrow{\bar{a}} Id(\xi_1) \times m_2}(\bar{a} \notin P),$$

$$r5.\frac{\xi_1 \xrightarrow{\bar{a}} m_1, \xi_2 \xrightarrow{\bar{a}} m_2}{(\xi_1,\xi_2) \xrightarrow{\bar{a}} m_1 \times m_2}(\bar{a} \in P),$$

$$r6.\frac{\xi_1 \xrightarrow{\bar{a}} m_1, \xi_2 \not\xrightarrow{\bar{a}}}{(\xi_1,\xi_2) \xrightarrow{\bar{a}} m_1 \times Id(\xi_2)}(\bar{a} \in P), \quad r6'.\frac{\xi_1 \xrightarrow{\not\bar{a}}, \xi_2 \xrightarrow{\bar{a}} m_2}{(\xi_1,\xi_2) \xrightarrow{\bar{a}} Id(\xi_1) \times m_2}(\bar{a} \in P),$$

For the scope operator $[\cdot]_C$, we get the following two rules for NTSs.

$$r7.\frac{\xi \xrightarrow{a} m}{[\xi]_C \xrightarrow{a} [m]_C},$$

$$r8.\frac{\xi \xrightarrow{\bar{a}} m}{[\xi]_C \xrightarrow{\bar{a}} [m]_C}(\bar{a} \notin C),$$

where $[m]_C$ is the measure defined as $[m]_C([A]_C) = m(A)$, where $[A]_C := \{[s]_C | s \in A\}$.

The operators $|_A^P|$ and $[\cdot]_C$ are now, in the context of NTSs, formally defined as follows.

Definition 4.6. Let $X = (S_1, \Sigma \cup \bar{\Sigma}, \mathcal{T}_1)$ and $Y = (S_2, \Sigma \cup \bar{\Sigma}, \mathcal{T}_2)$ be two NTSs. Then $X|_A^P|Y$ is defined as the NTS $(S_1 \times S_2, \Sigma \cup \bar{\Sigma}, \mathcal{T})$, where $\mathcal{T}$ is the least relation satisfying the rules r1,r2,r2',r3,r3',r4,r4',r5,r6 and r6'. $[X]_C$ is defined as the NTS $([S_1]_C, \Sigma \cup \bar{\Sigma}, \tilde{\mathcal{T}})$, where $[S_1]_C := \{[s]_C | s \in S_1\}$ and $\tilde{\mathcal{T}}$ is the least relation satisfying the rules r7 and r8.

This definition of composition for NTSs will form the basis for composition of CPDPs in chapter 7.

5

Interactive Markov Chains

In this chapter we review Interactive Markov Chains (IMCs). Most of the material of this chapter comes from [Her02]. IMC forms a compositional modelling and analysis framework for discrete state systems with spontaneous and nondeterministic transitions. CPDP can be seen as a hybrid extension of IMC, where each state/location of the IMC is extended to a hybrid location which has a continuous state space and continuous dynamics attached to it. We will see that the line of results of this Chapter (semantics, composition, maximal progress and bisimulation) will form the blue print of the line of result for CPDPs, described in the Chapters 7, 8 and 9.

This chapter is organized as follows. First we define IMC and give semantics in terms of CFSJS and NTS. Then we define composition and abstraction for IMC. Finally we define strong and weak bisimulation for IMC.

State reduction of IMCs can be done by finding the maximal bisimulation (algorithms for this can be found in [DHS04]). State reduction can be done in a compositional way by replacing components by their maximal state reduced bisimilar components. Under the maximal progress assumption (explained later in this chapter), IMCs sometimes can be reduced by weak bisimulation to Markov Chains. This happens if through bisimulation all interactive transitions are removed and the resulting IMC only has Markovian transitions. Bisimulation for IMCs forms a powerful compositional state reduction technique, which leads, in the cases where all interactive transitions are removed, to Markov chains. These Markov chains can then be analyzed via Markov chain analysis techniques. In the coming chapters, we will see that the analogue for CPDP will be that under maximal progress and bisimulation CPDPs may be reduced to PDPs and can therefore be analyzed via PDP analysis techniques.

5.1 Interactive Markov Chains

Definition 5.1. An Interactive Markov Chain is a tuple $(L, l_0, Act, \mathcal{I}, \mathcal{M})$, where

- L is a nonempty set of locations, $l_0 \in L$ is the initial location,
- Act is a set of actions,
- $\mathcal{I} \subset L \times Act \times L$ is a set of interactive transitions,

- $\mathcal{M} \subset L \times \mathbb{R}_+ \times L$ is a set of Markovian transitions.

For an interactive transition $\alpha = (l, a, l')$ with $a \in Act$ we call l the origin location of α, we call l' the target location of α and we call a the label or the action of α. We assume that $\tau \in Act$, where τ stands for an internal action. A τ-transition is not visible for an external observer. For a Markovian transition $\alpha = (l, \lambda, l')$ with $\lambda \in \mathbb{R}_+$ we call λ the transition rate or jump rate of α. We call interactive and Markovian transitions with origin location l *enabled* at l or l*-enabled*.

5.1.1 Semantics of IMCs

The semantics of an IMC can be split into a CFSJS part and an NTS part. Roughly said, the CFSJS is determined by the Markovian transitions and the NTS is determined by the interactive transitions.

Let $X = (L, l_0, Act, \mathcal{I}, \mathcal{M})$ be an IMC that starts at time t_0 in its initial location l_0. The CFSJS part of X at is determined by a race of the l_0-enabled spontaneous transitions. Each spontaneous transition α has a jump time that is exponentially distributed with some parameter λ_α. The l_0-enabled spontaneous transition that switches first, wins the race and determines the location after the switch. For example, at l_0 in Figure 5.1, there are two spontaneous transitions. If (l_0, λ, l_1) switches at time t_1 before (l_0, μ, l_0) switches, then at time t_1 the IMC process switches to location l_1.

From Section 3.2.2 we know that this combination of the two l_0-enabled spontaneous transitions can be represented as one l_0-enabled spontaneous transition with jump rate $\mu + \lambda$ and with reset measure $\frac{\mu}{\lambda+\mu} Id(l_0) + \frac{\lambda}{\lambda+\mu} Id(l_1)$, where $Id(l_i)$ is the identity reset map that jumps to location l_i with probability one. The measurable space we use here is $(L, 2^L)$, which is the natural measurable space for finite sets.

The CFSJS of X is now defined as $(L, l_0, \phi, \lambda, Q)$. Here, for all $t \geq 0$ and all $l \in L$, $\phi(t, l) = l$, which expresses that the state (which equals the location in this case) does not change between two switches. Furthermore,

$$\lambda(l) = \sum_{\alpha \in \mathcal{M}_{l\rightarrow}} \lambda_\alpha,$$

and

$$Q(l) = \sum_{\alpha \in \mathcal{M}_{l_0\rightarrow}} \frac{\lambda_\alpha}{\lambda(l)} Id(l'_\alpha),$$

where l'_α denotes the target location of transition α.

The NTS of X is defined as $(L, Act, \mathcal{T})$, where $\mathcal{T} := \{(l, a, Id(l')) | (l, a, l') \in \mathcal{I}\}$.

For example, the IMC X from Figure 5.1 has CFSJS $(L, l_0, \phi, \lambda, Q)$ with $L = \{l_0, l_1, l_2\}$, $\phi(t, l) = l$ for all $l \in L$ and all $t \geq 0$, $\lambda(l_0) = \lambda + \mu$, $\lambda(l_1) = \mu$, $\lambda(l_2) = 0$, $Q(l_0) = \frac{\mu}{\lambda+\mu} Id(l_0) + \frac{\lambda}{\lambda+\mu} Id(l_1)$, $Q(l_1) = Id(l_1)$, $Q(l_2)$ is not defined (and not relevant). The NTS of X equals $(L, Act, \mathcal{T})$, where $Act = \{a, b\}$ and $\mathcal{T} = \{(l_0, a, Id(l_1)), (l_1, b, Id(l_2))\}$.

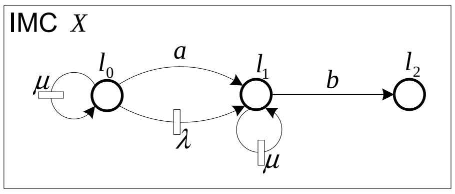

Figure 5.1: Example of an IMC

5.2 Composition and abstraction of IMCs

Composition of two IMCs with shared action set *Act* is defined via a parallel composition operator $|A|$, with $A \subset Act\backslash\{\tau\}$. For IMC there is no distinction of active and passive actions. Interaction between two IMCs is modelled as synchronization of events/actions. Composition $X_1|A|X_2$ expresses that, if $a \in A$, X_1 can execute an a-transition only if at the same time X_2 executes an a-transition, and vice versa (see also Definition 4.3, where this is also expressed). Thus, actions from A should synchronize. The Markovian transitions of two component IMCs in a composition are executed independently from each other. Consequently, two random variables determining the jump-times of two Markovian transitions, one transition of the first component and one transition of the second component of the composition, are independent. The class of IMCs is closed under parallel composition. Parallel composition is defined with structural operational semantical rules as follows.

Definition 5.2. Let $X_1 = (L_1, \hat{l}_1, Act, \mathcal{I}_1, \mathcal{M}_1)$ and $X_2 = (L_2, \hat{l}_2, Act, \mathcal{I}_2, \mathcal{M}_2)$ be two IMCs. Parallel composition of X_1 and X_2 on actions $a \in A$, where $A \subset Act\backslash\{\tau\}$ is the set of synchronization actions, is defined as the IMC $X_1|A|X_2 = (L, \hat{l}, Act, \mathcal{I}, \mathcal{M})$, where

- $L = \{l_1|A|l_2 \mid l_1 \in L_1, l_2 \in L_2\}$, $\hat{l} = \hat{l}_1|A|\hat{l}_2$,
- $\mathcal{I}$ is the least relation satisfying the rules $r1$, $r2$ and $r2'$,
- $\mathcal{M}$ is the least relation satisfying the rules $r3$ and $r3'$,

where rules $r1$, $r2$, $r2'$, $r3$ and $r3'$ are defined as

$$r1.\frac{l_1 \xrightarrow{a} l_1', l_2 \xrightarrow{a} l_2'}{l_1|A|l_2 \xrightarrow{a} l_1'|A|l_2'}(a \in A),$$

$$r2.\frac{l_1 \xrightarrow{a} l_1'}{l_1|A|l_2 \xrightarrow{a} l_1'|A|l_2}(a \notin A), \quad r2'.\frac{l_2 \xrightarrow{a} l_2'}{l_1|A|l_2 \xrightarrow{a} l_1|A|l_2'}(a \notin A),$$

$$r3.\frac{l_1 \xrightarrow{\lambda} l_1'}{l_1|A|l_2 \xrightarrow{\lambda} l_1'|A|l_2}, \quad r3'.\frac{l_2 \xrightarrow{\lambda} l_2'}{l_1|A|l_2 \xrightarrow{\lambda} l_1|A|l_2'}.$$

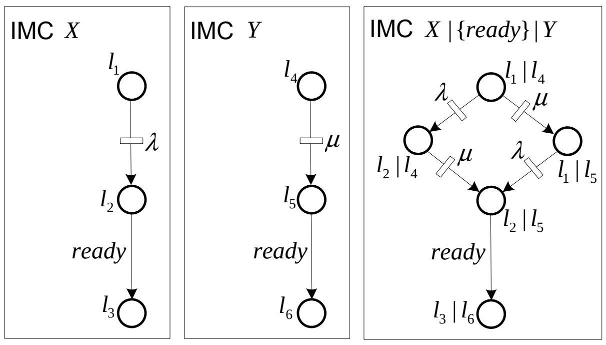

Figure 5.2: Example of composition of IMCs

It can easily be seen that the composition operator $|A|$ is commutative and associative. If IMCs X_1 and X_2 are composed with $|A|$, then if $a \in A$, the execution of an a-transition of X_1 depends on the presence of a-transitions in X_2. If at some time there is no a-transition enabled in X_2, then all enabled a-transitions of X_1 are blocked, i.e., these a-transitions cannot be executed because there is no synchronization partner (i.e., an a-transition in X_2), available.

The interplay between spontaneous transitions of different components in a composition is fully expressed by the simple composition rules r3 and r3'. That this interplay can be modelled in such a simple way, is due to the fact that the exponential distribution has the *memoryless* property. This can be see as a special case of the memoryless property for CFSJSs (see Section 3.2.1). We explain this by looking at the composition $X|\{ready\}|Y$ of Figure 5.2 X and Y synchronize on action *ready*. Before this synchronized transition can be executed, both X and Y need to have executed their spontaneous transitions with rates λ and μ respectively. If X executes the λ-transition first, then X has to wait till Y executed the μ-transition before action *ready* can be executed and vice versa if Y executes the μ-transition first. The question is: does IMC $X|\{ready\}|Y$ of Figure 5.2 model this situation correctly? 'X executing the λ-transition before Y executes the μ-transition' is modelled in $X|\{ready\}|Y$ by transition $(l_1|l_4, \lambda, l_2|l_4)$. 'X waiting till Y executes the μ-transition' is modelled by transition $(l_2|l_4, \mu, l_2|l_5)$. This is where the memoryless property plays its role. At first glance it might seem incorrect that a new sample is drawn from the same $exp(\mu)$ distribution at location $l_2|l_4$. However, if X executed the λ-transition at time t_1, and if random variable T models the time we have to wait from $t = t_1$ before Y executes the μ-transition, then

$$\mathrm{P}(T > \hat{t}) = \mathrm{P}(T_\mu > t_1 + \hat{t} | T_\mu > t_1),$$

where T_μ is a random variable exponentially distributed with parameter μ. The memoryless property of the exponential distribution implies that

$$\mathrm{P}(T_\mu > t_1 + \hat{t} | T_\mu > t_1) = \mathrm{P}(T_\mu > \hat{t}),$$

and we find that T and T_μ have the same probability distribution. Therefore, transition $(l_2|l_4, \mu, l_2|l_5)$ models the situation correctly. (We could also take the view of Section 3.2.1 to explain the correctness of IMC $X|\{ready\}|Y$. Then at location $l_2|l_4$ we have a situation of reassessing the jump-time of the μ-transition of Y. This reassessment is then expressed by transition $(l_2|l_4, \mu, l_2|l_5)$). Here we see the power of IMCs: the syntax of IMCs is simple, but still it is rich enough to fully express any composition of IMCs. In [Her02], this feature of IMCs is discussed and a comparison is made with other models from the literature that model such interplay between spontaneous transitions.

We make a distinction between external transitions, i.e., all a-transitions with $a \neq \tau$, and internal transitions, i.e., all τ-transitions and all spontaneous transitions. a-transitions have an external effect because in a composition context where a is a synchronization action, the absence of interactive transitions with action $a \in Act\backslash\{\tau\}$ of one IMC blocks a-transitions of other IMCs. The presence of such an a-transition allows a-transitions of other IMCs in that composition context. τ-transitions and Markovian transitions cannot block or be blocked and have therefore no external effect, they are internal transitions. For IMCs the abstraction operator $[X]_C$ is defined, which abstracts/internalizes actions from $C \subset Act\backslash\{\tau\}$ in IMC X.

Definition 5.3. Let $X = (L, l_0, Act, \mathcal{I}, \mathcal{M})$ be an IMC. Abstraction of actions from $A \subset Act$ in X results in the IMC $[X]_A = (\tilde{L}, \tilde{l}_0, Act, \tilde{\mathcal{I}}, \tilde{\mathcal{M}})$, where

- $\tilde{L} = \{[l]_A \mid l \in L\}$, $\tilde{l}_0 = [l_0]_A$,
- $\tilde{\mathcal{I}}$ is the least relation satisfying rules $r4$ and $r4'$,
- $\tilde{\mathcal{M}}$ is the least relation satisfying rule $r5$,

where rules $r4$,$r4'$ and $r5$ are defined as

$$r4.\frac{l_1 \xrightarrow{a} l_1'}{[l_1]_A \xrightarrow{a} [l_1']_A}(a \notin A), \quad r4'.\frac{l_1 \xrightarrow{a} l_1'}{[l_1]_A \xrightarrow{\tau} [l_1']_A}(a \in A)$$

$$r5.\frac{l_1 \xrightarrow{\lambda} l_1'}{[l_1]_A \xrightarrow{\lambda} [l_1']_A}.$$

Thus, abstraction, with abstraction-set A, turns a-transitions, where $a \in A$, into τ-transitions. Abstraction can be used to express that two IMCs X_1 and X_2 synchronize on some label a while a third IMC X_3 should execute its a-transitions independently from X_1 and X_2. This is expressed by

$$[X_1|A_1|X_2]_C|A_2|X_3,$$

where $a \in A_1$ and $a \in C$ and $a \notin A_2$. Note that the scope operator $[\cdot]_C$ (see Definition 4.3) also expresses that another component can execute its active transitions 'independently', where 'independently' means 'without triggering passive transitions in the other component'.

In [Her02] it is assumed that IMCs are executed under *maximal progress*. This means that if an IMC X enters a location l, then a transition has to be executed as soon as possible. This implies that if $l \xrightarrow{\tau}$ (i.e., there is a τ transition from l) then the time spent in location l is zero because a τ-transition can always be executed immediately since it is internal. Whether an a-transition can be executed immediately or is delayed depends on the environment and cannot be determined by X itself. A Markovian transition $\alpha = (l, \lambda, l')$ is never executed immediately because $\mathrm{P}(T_\alpha = 0) = 0$, where T_α is the random variable that determines the transition time of α. An external generator should, under maximal progress, provide τ-transitions with execution times equal to zero.

5.3 Equivalences on IMCs

Two equivalence notions are defined for IMCs: strong bisimulation and weak bisimulation. Weak bisimulation can be seen as external equivalence. Roughly said, an external agent which can at any time block any action from Σ of an IMC and which can observe at any time which actions from Σ are enabled in that IMC, cannot discriminate the behaviors of two weakly bisimilar IMCs. Strong bisimulation is stronger than weak bisimulation because it requires that two strongly bisimilar IMCs X_1 and X_2 simulate each other stepwise also for internal transitions, where weak bisimulation only requires stepwise simulation for external transitions. This means that for weakly bisimilar IMCs a τ-transition in the first IMC may be simulated by a series of τ-transitions in the other IMC (and vice versa).

In order to define strong and weak bisimulation for IMCs, we need the function $\gamma_M(l, C)$ that calculates the cumulative rate of the set of all l-enabled Markovian transitions with target location in $C \subset L$. γ_M is defined as

$$\gamma_M(l, C) = \sum_{\alpha \in \mathcal{M}_{l \to C}} \lambda_\alpha,$$

where $\mathcal{M}_{l \to C}$ denotes the set of all Markovian transitions with origin location l and target location in C. λ_α denotes the transition rate of transition α.

Definition 5.4. Let $X = (L, l_0, \mathit{Act}, \mathcal{I}, \mathcal{M})$ be an IMC. An equivalence relation $\mathcal{R} \subset L \times L$ is a strong bisimulation on X if $(l_1, l_2) \in \mathcal{R}$ implies for all $a \in \mathit{Act}$ and all equivalence classes C of $\mathcal{R}$

1. $l_1 \xrightarrow{a} l_1'$ implies $l_2 \xrightarrow{a} l_2'$ for some $l_2' \in L$ with $(l_1', l_2') \in \mathcal{R}$,
2. $l_1 \overset{\tau}{\not\to}$ implies $\gamma_M(l_1, C) = \gamma_M(l_2, C)$.

Two locations l_1 and l_2 are strongly bisimilar, written $l_1 \sim l_2$, if they are contained in some strong bisimulation on X.

Condition 2 in this definition says that if a location l has a τ-transition enabled, then all l-enabled Markovian transitions play no role in the execution, since under maximal progress a τ-transition will be executed immediately when location l is reached.

We say that two IMCs $X_1 = (L_1, \hat{l}_1, Act, \mathcal{I}_1, \mathcal{M}_1)$ and $X_2 = (L_2, \hat{l}_2, Act, \mathcal{I}_2, \mathcal{M}_2)$ are strongly bisimilar, and write $X_1 \sim X_2$, if $\hat{l}_1 \sim \hat{l}_2$, where $\sim$ denotes strong bisimilarity on IMC $(L_1 \cup L_2, \hat{l}, Act, \mathcal{I}_1 \cup \mathcal{I}_2, \mathcal{M}_1 \cup \mathcal{M}_2)$, where $\hat{l}$ is some location in $L_1 \cup L_2$. (Note that the initial state $\hat{l}$ does not influence the strong bisimilarity relation). In this way, strong bisimilarity is also defined on $\mathcal{IMC}$, which is the set of all IMCs, and it turns out to be an equivalence relation on $\mathcal{IMC}$.

In [Her02] it is proven that strong bisimilarity on the location set of a single IMC or on $\mathcal{IMC}$ is the largest (or maximal) bisimulation. Also, it is proven that strong bisimilarity on $\mathcal{IMC}$ is substitutive with respect to parallel composition and abstraction, i.e., $X_1 \sim X_2$ implies $X_1|A|Y \sim X_2|A|Y$, $X_1 \sim X_2$ implies $Y|A|X_1 \sim Y|A|X_2$ and $X_1 \sim X_2$ implies $[X_1]_A \sim [X_2]_A$.

Definition 5.5. Let $X = (L, l_0, Act, \mathcal{I}, \mathcal{M})$ be an IMC. An equivalence relation $\mathcal{R} \subset L \times L$ is a weak bisimulation on X if $(l_1, l_2) \in \mathcal{R}$ implies for all $a \in Act$

1. $l_1 \overset{a}{\Rightarrow} l_1'$ implies $l_2 \overset{a}{\Rightarrow} l_2'$ for some $l_2' \in L$ with $(l_1', l_2') \in \mathcal{R}$,
2. $l_1 \overset{\tau}{\Rightarrow} l_1'$ and $l_1 \overset{\tau}{\not\rightarrow}$ imply $l_2 \overset{\tau}{\Rightarrow} l_2'$ for some $l_2' \in L$ with $l_2' \overset{\tau}{\not\rightarrow}$ and $\gamma_M(l_1', C^\tau) = \gamma_M(l_2', C^\tau)$ for all equivalence classes C of $\mathcal{R}$,

where $l \overset{\tau}{\Rightarrow} l'$ if $l' = l$ or if there exists a series of τ-transitions from l leading to l'; $l \overset{a}{\Rightarrow} l'$ if there exist $\tilde{l}$ and $\tilde{l}'$ such that $l \overset{\tau}{\Rightarrow} \tilde{l} \overset{a}{\rightarrow} \tilde{l}' \overset{\tau}{\Rightarrow} l'$, and $C^\tau := \{l \in L | \exists l' \in C, l \overset{\tau}{\Rightarrow} l'\}$.

Two locations l_1 and l_2 are weakly bisimilar, written $l_1 \approx l_2$, if they are contained in some weak bisimulation on X.

Weak bisimulation can be defined on $\mathcal{IMC}$ similarly as it is done for strong bisimulation. Also for weak bisimulation it is true that weak bisimilarity is the largest bisimulation and the substitutive property holds for parallel composition and abstraction.

With strong/weak bisimulation we can reduce the state space of an IMC X by not considering the individual states of X, but by considering the equivalence classes of X with respect to strong/weak bisimilarity. The IMC strongly/weakly bisimilar to X with the smallest state space is the IMC that has as its states the equivalence classes of X with respect to strong/weak bisimilarity. Performance analysis of X can now be executed on the state reduced version of X if the performance measure does not consider the individual states of X but only the equivalence classes of X. A performance measure that asks for the probability that the process is in a specific state (or location) l at time $\hat{t}$ cannot be evaluated on the state reduced process. A strategy to overcome this problem is to use a *reward function Rew* which is a mapping from L, the set of locations, to $\mathbb{R}_+$. *Rew*-preserving strong/weak bisimilarity is then defined as: states l_1 and l_2 are *Rew*-preserving strongly/weakly bisimilar if $Rew(l_1) = Rew(l_2)$ and l_1 and l_2 are strongly/weakly bisimilar. Then, if we define $Rew(l) = 1$ and $Rew(l') = 0$ for all $l' \in L \backslash \{l\}$, then the performance measure above can be evaluated at the state reduced process because this reward function ensures that the equivalence class l does not contain other states than l.

In Chapter 7 it is shown how IMC can be seen as a subclass of CPDP. In Chapter 9 strong bisimulation is defined for CPDP and it is shown that, restricted to CPDPs of the IMC form, strong bisimulation for CPDP coincides with strong bisimulation for IMC.

Example 5.6. We illustrate the concepts of composition, abstraction and strong/ weak bisimulation with the IMCs of Figure 5.3. We assume that IMC X has initial state l_0 and IMC Y has initial state l_3. IMC $X|a|Y$, which is shorthand for $X|\{a\}|Y$, pictures the composition of X with Y with a the only synchronization action. $l_0|l_3$, which is shorthand for $l_0|\{a\}|l_3$, is the initial location of $X|a|Y$. Assume that $X|a|Y$ is a closed system, i.e., $X|a|Y$ does not interact with other IMCs. Then, the actions a and b can be internalized. The result is denoted by IMC Z, i.e., $Z := [X|a|Y]_A$ with $A = \{a, b\}$. Let Rew be a reward function such that $\mathit{Rew}(l_0|l_4) \neq \mathit{Rew}(l_0|l_5)$. The IMCs Z_1, Z_2, Z_3 and Z_4 with initial locations l_6, l_7, l_8 and l_9 respectively are all equivalent to Z according to some equivalence notion. Z_1 and Z are *Rew*-preserving strongly bisimilar, Z_2 and Z are *Rew*-preserving weakly bisimilar (but not strongly), Z_3 and Z are strongly bisimilar (but not *Rew*-preserving), Z_4 and Z are weakly bisimilar (but not strongly bisimilar and also not *Rew*-preserving).

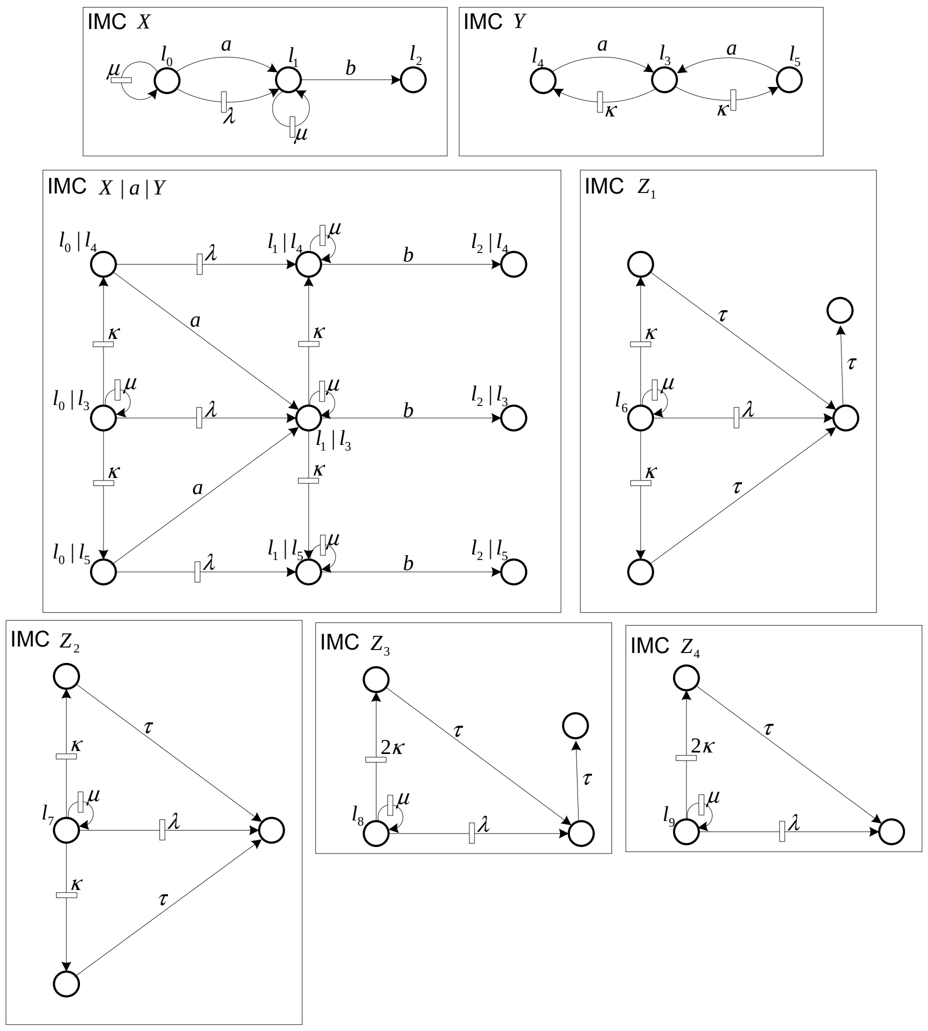

Figure 5.3: Composition, abstraction and bisimulation for IMCs

6

Piecewise Deterministic Markov Processes

The class of Piecewise Deterministic Markov Processes (PDPs) was introduced in 1984 in [Dav84]. A PDP consists of a mixture of motion determined by ODEs, and random jumps. The jumps are executed at random times $T_1 < T_2 < T_3 \ldots$. At these random times the process jumps instantly to another state. The target state, i.e., the state where the process jumps to, is determined by a probability measure that depends on the state of the process before the jump. In between the random times the PDP evolves deterministically according to some ODE. Thus, the process is piecewise deterministic. The process is also Markov, which means that the current state of the process contains all possible information for predicting the future of the process. The state space of a PDP is hybrid; it consists of a set of locations and attached to each location a continuous state space. The behavior of a PDP can be captured as a CFSJS together with an FTS.

The PDP framework is quite general. Virtually every stochastic system that does not include diffusions can be modelled as a PDP [Dav84]. In [Dav93] the generator and its domain of the PDP is characterized. This is a strong result which made it possible that powerful analysis and control techniques were developed. It is for example possible to calculate expectations of the form $\mathbb{E}(f(x_t))$, where $\{x_t\}$ is the PDP, by solving a set of so called integro-differential equations (see [Dav93]). This method is in many cases less computationally intensive than running computer simulations to answer, for example, reachability questions. In this chapter we define the PDP model and we give its CFSJS and FTS semantics. We also formally define the stochastic process of the PDP. By giving this stochastic process, we also show how any CFSJS/FTS combination that satisfies the PDP conditions, can be translated into a stochastic process. In the last section we give an example of a PDP. This example is a modified version of the 'repair shop model' example from [Dav93]. The example has a clear compositional structure, but because the PDP framework does not allow compositional specification, the model has to be defined monolithically as a single PDP. In Chapter 7 we will see how this example can be modelled in a compositional way by using CPDPs.

6.1 Definition of the PDP model

A PDP is a stochastic process $\{\xi_t\}$ on a state space E. This means that for each t, ξ_t is an E-valued random variable, i.e., $\xi_t : \Omega \to E$ for some sample space Ω. E is an open subset of a Borel space. This means that we can define the closure $\bar{E}$ of E and the boundary $\partial E := \bar{E} \backslash E$ of E. The formal definition is as follows.

Definition 6.1. A PDP is a tuple (L, Inv, F, λ, Q), where

1. L is a countable set of locations.
2. Inv assigns to each $l \in L$ an open subset of $\mathbb{R}^{d(l)}$, where $d(l) \in \mathbb{N}$ is the dimension of $Inv(l)$. With $E := \{(l, x) | l \in L, x \in Inv(l)\}$ we denote the hybrid state space of the PDP.
3. F assigns to each $l \in L$ a locally Lipschitz vector field in $Inv(l)$, determining a flow $\phi(t, \xi)$ for all $\xi \in E$.
4. $\lambda : E \to \mathbb{R}_+$, is a measurable function such that for each $\xi \in E$, there exists $\epsilon(\xi) > 0$ such that the function $s \mapsto \lambda(s, \phi(s, \xi))$ is integrable on $[0, \epsilon(\xi)[$.
5. $Q : E \cup \partial E \to Prob(E)$, which is the transition measure, is such that for each fixed $A \in \mathcal{E}$, the map $x \mapsto Q(A; x)$ is measurable and $Q(\{x\}; x) = 0$.

For a PDP, it is assumed that the following conditions: there are no *explosions* (i.e., for all $\xi \in E$, $\phi(t, \xi) < \infty$ for all $t > 0$) and the PDP is *non-zeno* (i.e., for any initial state $\xi_0 \in E$, the expectation of the number of transitions executed within $[0, t]$ is smaller than ∞ for any $t > 0$).

To illustrate the PDP model, we give the following simple example.

Example 6.2. We consider a PDP with two locations, l_1 and l_2, which both have as continuous state space the interval $]-1, 1[$, i.e. $Inv(l_1) = Inv(l_2) =]-1, 1[$. Both locations have clock dynamics, i.e., $F(l_1)(x) = F(l_2)(x) = 1$ for all $x \in]-1, 1[$. If one of the locations reaches the boundary of the invariant, then a transition takes place to continuous state 0 of the other location, i.e., $Q(\{(l_2, 0)\}; l_1, 1) = 1$ and $Q(\{(l_1, 0)\}; l_2, 1) = 1$, expressing that measures $Q(l_1, 1)$ and $Q(l_2, 1)$ assign probability one to the sets $\{(l_2, 0)\}$ and $\{(l_1, 0)\}$ respectively. In both locations a spontaneous transition can happen with jump rate $\tilde{\lambda} \in \mathbb{R}_+$ at all states, to state 0 of the other location, i.e., $\lambda(l_1, x) = \lambda(l_2, x) = \tilde{\lambda}$ for all $x \in]-1, 1[$; $Q(\{(l_2, 0)\}; l_1, x) = 1$ and $Q(\{(l_1, 0)\}; l_2, x) = 1$ for all $x \in]-1, 1[$.

6.2 CFSJS/FTS semantics for PDPs

We give the semantics of a PDP as its CFSJS and its FTS. In Section 3.4, the transition mechanism structure of the combination of CFSJS and FTS is determined. This means that by giving the CFSJS and FTS of a PDP, we determine how stochastic executions can be generated (via the transition mechanism).

The CFSJS of a PDP $X = (L, Inv, F, \lambda, Q)$ with state space $E = \{(l, x) | l \in L, x \in Inv(l)\}$ and initial state ξ_0, is defined as $(E \cup \partial E, \xi_0, \phi, \lambda, Q)$. The flow map $\phi(t, (l, x))$ is determined by the differential equation

$$\dot{x} = F(l)(x).$$

The FTS of PDP X is defined as $(E \cup \partial E, \mathcal{T})$, where $(\xi, m) \in \mathcal{T}$ if and only if $\xi \in \partial E$ and $Q(\xi) = m$.

Note that the state space $E \cup \partial E$ of the CFSJS can be bounded. This means that executing this CFSJS may lead to problems when no switch occurs when the boundary of E is reached. However, the CFSJS in combination with the FTS can always be executed because a forced transition will happen when the boundary ∂E is hit.

6.3 The stochastic process of a PDP

We formally define the stochastic process $\{\xi_t\}$ of a PDP $X = (L, Inv, F, \lambda, Q)$ in the way it is done in [Dav93]. Let E denote the state space of X, let ϕ denote the flow map of X and let ξ_0 be the initial state.

Let $\Omega = \prod_{i=1}^{\infty} [0, 1]$, i.e., Ω is the infinite product space of $[0, 1]$-intervals. Let $\mathcal{A}$ be the set of all Borel sets of Ω and let P be the product measure of all P_i's ($i = 1, 2, \cdots \infty$), where each P_i is the Lebesgue measure (restricted to $[0, 1]$) on the i-th interval $[0, 1]$. We define the stochastic process on the probability space $(\Omega, \mathcal{A}, P)$. This space is called the *Hilbert cube*.

First we define for $i \in \mathbb{N}$ $U_i : \Omega \to [0, 1]$ as $U_i(\omega) = \omega_i$, where $\omega = (\omega_1, \omega_2, \cdots)$. Now $U_1, U_2, \cdots$ form a series of independent random variables, all uniformly distributed on the interval $[0, 1]$. These random variables will be used to define the random variables ξ_t, for $t \geq 0$, of the stochastic process. Before we formally define the realization of the stochastic process, we first give a short description of this realization.

We assume an initial state ξ_0 is given. First an infinite series of random variables S_1,T_1,S_2,T_2,S_3,T_3,$\cdots$ is constructed on the probability space $(\Omega, \mathcal{A}, P)$. S_1 denotes the first period before a transition happens. S_2 denotes the period between the first and the second transition, S_3 denotes the period between the second and third transition, etc. T_1 denotes the time of the first transition, T_2, which equals $T_1 + S_2$, denotes the time of the second transition, etc. All these random variables are constructed in line with the survivor functions and the transition measure of X. From these random variables, the stochastic process ξ_t can directly be constructed: for any $\omega \in \Omega$, $x_t(\omega) := \phi(t, \xi_0)$ for $0 \leq t < T_1(\omega)$. Then $x_{T_1(\omega)}$ is chosen in line with the transition measure Q at state $\phi(T_1(\omega), \xi_0)$. Then, for $T_1(\omega) \leq t < T_2(\omega)$, $x_t(\omega) := \phi(t - T_1(\omega), x_{T_1(\omega)})$, etc. We now formally define this realization of the stochastic process.

Take $\omega \in \Omega$ arbitrary, we define the stochastic process by defining $\xi_t(\omega)$ for all $t \geq 0$.

For all $\xi \in E$ we define

$$F(t,\xi) = I_{(t<t_*(\xi))}\exp\left(-\int_0^t \lambda(\phi(s,\xi))\mathrm{d}s\right),$$

Let the random variable T_1 be the first jump time. Then its survivor function equals $F(t,\xi_0)$. For all $\xi \in E$ and all $u \in [0,1]$ we define

$$\Psi_1(u,\xi) = \begin{cases} \inf\{t|F(t,\xi) \le u\} \\ +\infty, \quad \text{if the above set is empty.} \end{cases}$$

Then we define $T_1(\omega) := \psi_1(U_1(\omega),\xi_0)$. We define the random variable S_1 as $S_1(\omega) := T_1(\omega)$. Now let $\Psi_2 : [0,1] \times (E \cup \partial E) \to E$ be a measurable function such that for all $\xi \in E \cup \partial E$ and all $A \in \mathcal{B}(E)$, $l(\{u|\psi_2(u,\xi) \in A\}) = Q(A,\xi)$. The existence of such a function is shown in [Dav93]. The sample path $\xi_t(\omega)$ up to the first jump time is now defined as follows: if $T_1(\omega) = \infty$, then

$$\xi_t(\omega) = \phi(t,\xi_0),$$

for $t \ge 0$. If $T_1(\omega) < \infty$, then

$$\xi_t(\omega) = \psi_2(U_2(\omega), \phi(T_1(\omega),\xi_0)).$$

The process now restarts from $\xi_{T_1}(\omega)$ according to the same recipe. Thus, if $T_1(\omega) < \infty$, we define

$$S_2(\omega) := \psi_1(U_3(\omega),\xi_{T_1}(\omega)),$$
$$T_2(\omega) := T_1(\omega) + S_2(\omega).$$

If $T_2(\omega) = \infty$, then

$$\xi_t(\omega) = \phi(t - T_1(\omega), \xi_{T_1}(\omega)), \quad t \ge T_1(\omega),$$

while if $T_2(\omega) < \infty$, then

$$\xi_t(\omega) = \phi(t - T_1(\omega), \xi_{T_1}(\omega)), \quad T_1(\omega) \le t < T_2(\omega)$$
$$\xi_{T_2}(\omega) = \psi_2(U_4(\omega), \phi(S_2(\omega), \xi_{T_1}(\omega)))$$

and so on. In [Dav93] it is shown that $\{\xi_t\}$ as defined above is a well-defined stochastic process.

6.4 Example

In this example we show how a system consisting of two independent machines that independently break down with jump rates λ_1 and λ_2 can be modelled as a PDP. The jump rates λ_1 and λ_2 are superimposed and therefore we find in this example a PDP jump rate of the form $\lambda_1 + \lambda_2$ and a transition measure of the form (3.4).

A production plant has two machines, M_1 and M_2, which operate independently. The machines break down with time-dependent jump-rates λ_i and the amount of time to repair a machine is probabilistic and is uniformly distributed over the interval $[t_1, t_2]$. When machine M_i reaches age s_i, then it has to be taken out of service for maintenance. The repair shop can fix only one machine at a time. If one machine breaks down while the other machine is under repair, then it has to wait till the other machine has been repaired before it can enter the repair shop. Let (L, Inv, F, λ, Q) be the PDP that models this system. Then,

1. L has five locations: l_1 for 'both machines are working', l_2 for 'M_1 working, M_2 under repair', l_3 for 'M_1 under repair, M_2 working', l_4 for 'M_1 under repair, M_2 down and waiting' and l_5 for 'M_1 down and waiting, M_2 under repair'.

2.
   - $Inv(l_1) = \{(x_1, x_2) | 0 \le x_1 < s_1, 0 \le x_2 < s_2\}$,
   - $Inv(l_2) = \{(x_1, r) | 0 \le x_1 < s_1, 0 \le r < t_2\}$,
   - $Inv(l_3) = \{(x_2, r) | 0 \le x_2 < s_2, 0 \le r < t_2\}$,
   - $Inv(l_4) = \{r | 0 \le r < t_2\}$,
   - $Inv(l_5) = Inv(l_4) = \{r | 0 \le r < t_2\}$.

3. The continuous dynamics of M_1 and M_2 is the clock dynamics and the repair shop has clock-countdown dynamics: $F(l_1, x_1, x_2) = [1, 1]^T$, $F(l_i, x, r) = [1, -1]^T$ for $i = 2, 3$, $F(l_i, x) = -1$ for $i = 4, 5$.

4. $\lambda(l_1, x_1, x_2) = \lambda_1(x_1) + \lambda_2(x_2)$, $\lambda(l_2, x_1, r) = \lambda_1(x_1)$, $\lambda(l_3, x_2, r) = \lambda_2(x_2)$, $\lambda(l_4, r) = \lambda(l_5, r) = 0$.

5. From l_1 transitions are possible to l_2 and l_3:

$$Q((l_2, \{x_1\} \times B); l_1, x_1, x_2) = \frac{\lambda_2(x_2)}{\lambda_1(x_1) + \lambda_2(x_2)} \frac{l(B \cap [t_1, t_2])}{t_2 - t_1},$$

where $B \in \mathcal{B}(\mathbb{R})$ and l is the Lebesgue measure.

$$Q((l_3, \{x_2\} \times B); l_1, x_1, x_2) = \frac{\lambda_1(x_1)}{\lambda_1(x_1) + \lambda_2(x_2)} \frac{l(B \cap [t_1, t_2])}{t_2 - t_1},$$

where $B \in \mathcal{B}(\mathbb{R})$. From l_2 transitions are possible to l_1 and l_5:

$$Q(\{(l_1, x_1, 0)\}; l_2, x_1, 0) = 1, \quad Q(\{(l_5, r)\}; l_2, x_1, r) = 1.$$

From l_3 transitions are possible to l_1 and l_4:

$$Q(\{(l_1, 0, x_2)\}; l_3, 0, x_2) = 1, \quad Q(\{(l_4, r)\}; l_3, x_2, r) = 1.$$

From l_4 transitions are possible to l_2:

$$Q((l_2, \{0\} \times B); l_4, 0) = \frac{l(B \cap [t_1, t_2])}{t_2 - t_1},$$

where $B \in \mathcal{B}(\mathbb{R})$. From l_5 transitions are possible to l_3:

$$Q((l_3, \{0\} \times B); l_5, 0) = \frac{l(B \cap [t_1, t_2])}{t_2 - t_1},$$

where $B \in \mathcal{B}(\mathbb{R})$.

In the next chapter we show how this system can be modelled in a compositional way by using CPDPs.

7 Communicating PDPs (CPDPs)

With PDPs we can model a broad class of stochastic systems. However, because PDPs do not allow modelling in a compositional way, the modelling process becomes nearly impossible if systems have a (very) complex structure. In this chapter we introduce the automaton model CPDP (communicating PDP), which makes it possible to model PDP-type systems in a compositional way. With CPDP it becomes therefore possible to model very complex systems in an organized way such that the (complex) model can still be interpreted and verified by exploiting the compositional structure of the system.

7.1 Definition of the CPDP model

Definition 7.1. A CPDP is a tuple $(L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$, where

- L is a set of locations
- V is a set of state variables. With $d(v)$ for $v \in V$ we denote the dimension of variable v. $v \in V$ takes its values in $\mathbb{R}^{d(v)}$.
- W is a set of output variables. With $d(w)$ for $w \in W$ we denote the dimension of the variable w. $w \in W$ takes its values in $\mathbb{R}^{d(w)}$.
- $\nu : L \to 2^V$ maps each location to a subset of V, which is the set of state variables of the corresponding location. We call the valuation space of the variables of $\nu(l)$ the state space of location l.
- $\omega : L \to 2^W$ maps each location to a subset of W, which is the set of output variables of the corresponding location. We call the valuation space of the variables of $\omega(l)$ the output space of location l.
- F assigns to each location l and each $v \in \nu(l)$ a mapping from $\mathbb{R}^{d(v)}$ to $\mathbb{R}^{d(v)}$, i.e., $F(l, v) : \mathbb{R}^{d(v)} \to \mathbb{R}^{d(v)}$. $F(l, v)$ is the vector field that determines the evolution of v for location l (i.e., $\dot{v} = F(l, v)$ for location l).
- G assigns to each location l and each $w \in \omega(l)$ a mapping from $\mathbb{R}^{d(v_1)+\cdots+d(v_m)}$ to $\mathbb{R}^{d(w)}$, where v_1 till v_m are the state variables of location l. $G(l, w)$ determines the output equation of w for location l (i.e., $w = G(l, w)$).

- Σ is the set of communication labels. $\bar{\Sigma}$ denotes the 'passive' mirror of Σ and is defined as $\bar{\Sigma} = \{\bar{a} | a \in \Sigma\}$.
- $\mathcal{A}$ is a set of active transitions and consists of five-tuples (l, a, l', G, R), denoting a transition from location $l \in L$ to location $l' \in L$ with communication label $a \in \Sigma$, guard G and reset map R. G is a subset of the *valuation space* of l, which notion is introduced in Section 7.1.1. The reset map R assigns to each point in G for each variable $v \in \nu(l')$ a probability measure on $\mathbb{R}^{d(v)}$, i.e., $R(g, v) \in Prob(\mathbb{R}^{d(v)})$ for all $g \in G$ and all $v \in \nu(l')$.
- $\mathcal{P}$ is a set of passive transitions of the form $(l, \bar{a}, l', R)$. R is defined on the valuation space of l as the R of an active transition is defined on the guard space.
- $\mathcal{S}$ is a set of spontaneous transitions and consists of four-tuples (l, λ, l', R), denoting a transition from location $l \in L$ to location $l' \in L$ with jump-rate λ and reset map R. The jump rate λ (i.e., the Poisson rate of the Poisson process of the spontaneous transition) is a mapping from the valuation space of l to $\mathbb{R}_+$. R is defined on the valuation space of l as it is done for passive transitions.

The graphical notation of a CPDP is as follows. The locations are pictured as circles (see for example Figure 7.1). The differential and output equations that belong to a location l are written inside the circle of l. A transition from location l to location l' is drawn as an arrow from l to l'. The communication label from Σ, the reset map and the guard of the transition are written above (or next to) the arrow. A passive transition is pictured as an active transition, except that the label is now from $\bar{\Sigma}$ and there is no guard. A spontaneous transition is pictured as an arrow with a little box in the middle. This notation is chosen in line with the IMC notation. The jump rate and reset map of a spontaneous transition are written above or next to this little box.

Example 7.2. In Figure 7.1 we see the CPDP X, which models a flying aircraft. The initial location of X is l_1. The initial state x_0 in l_1 represents the position and velocity of the aircraft at initial time t_0. Since the aircraft flies in three dimensional space, the state space of variable x is $\mathbb{R}^6$. Location l_1 represents a flying mode. This means that in l_1 the aircraft is somewhere up in the sky and not at the ground. The dynamics of the aircraft in flying mode l_1 is determined by the vector field f_1. In this model we do not discriminate between state and output, therefore in all locations the output is chosen to be a copy of the state, i.e., $y = x$. Location l_2 represents a non-nominal flying mode. In this mode the aircraft is flying while there is a defect. (This mode represents for example that the navigation system is not working properly). In this non-nominal flying mode the dynamics is determined by vector field f_2. In location l_1, the time till the defect occurs, is exponentially distributed with parameter λ_1. This is expressed by the spontaneous transition from l_1 to l_2 with jump rate λ_1. When this transition is executed at some state $(l_1, \{x = x_1\})$, i.e., when the defect occurs at this state, the state of l_2 is reset by $R_1(\{x = x_1\})$, which is a probability measure on $\mathbb{R}^6$, the state space of x. Since the position and velocity of the aircraft do not change when a switch from l_1 to l_2 happens, $R_1(\{x = x_1\})$ equals the probability measure $I_{\{x=x_1\}}$ which we call the identity measure and which assigns probability one to the singleton Borel set $\{x = x_1\}$. (Note that the identity measure $I_{\{x=x_1\}}$ is also called the Dirac measure for the point $\{x = x_1\}$). We call R_1 an identity reset map. In the non-nominal mode l_2, repair activities are undertaken. The time till the defect is repaired is exponentially distributed with parameter λ_2. This is expressed by the spontaneous transition from l_2 to l_1 which has reset map R_2, which also equals the identity reset map.

If the aircraft approaches the airport, the flying mode will change to a landing mode where the aircraft prepares for landing. These modes are represented by l_3 and l_4, where l_3 represents nominal landing and l_4 represents landing while there still is an unrepaired defect. l_1 switches to l_3 as soon as the altitude drops below a certain level h. Let x_1 till x_6 be the components of variable x and let x_3 denote the altitude of the aircraft. The guard G_1 of the transition from l_1 to l_3 is equal to $\{x = (x_1, x_2, \cdots, x_6) | x_3 \leq h\}$. This expresses that this transition is allowed to be executed as soon as $x \in G_1$, i.e., as soon as $x_3 \leq h$. In fact, we want that the transition is executed at the first time instant where $x \in G_1$. Later we will see that we can express this by assuming maximal progress, which means, roughly said, that active transitions should be executed as soon as the guard is satisfied. This notion of maximal progress is introduced and explained in Section 8.2. When this active transition is executed at some time t_2, then the identity reset map R_3 (also an identity reset map) resets the state and the discrete event *land* is executed at time t_2. The discrete event (and its transition) is executed instantaneously, i.e., does not consume time. Later we will see that these discrete events can be used for communication between CPDPs. In location l_3 the continuous dynamics (expressing the landing phase) is determined by vector field f_3. In the same way the transition from l_2 to l_4 with identity reset map R_4 represents the transition from flying mode to landing mode, but now while there is a defect present.

In this example the jump rates λ_1 and λ_2 are constants, expressing exponentially distributed times. However, the CPDP model also allows λ_1 and λ_2 to depend on the state, and therefore indirectly depend on the time. If for example λ_1 depends on x_3 such that $\lambda_1(x_1, x_2, \cdots, x_6) > \lambda_1(x'_1, x'_2, \cdots, x'_6)$ when $x_3 > x'_3$, then this would express that the rate of switching is larger for great altitudes, i.e., at great altitudes it is more likely that a defect occurs than at small altitudes.

One feature of the CPDP model, the passive transition, is not explained in the above example. The meaning of passive transitions becomes apparent in the context of communication between multiple CPDPs and is explained and illustrated in Section 7.3.

7.1.1 The state and output space of a CPDP

The state of a CPDP is hybrid; it consists of a location on the one hand and of values for the continuous variables on the other hand. For determining the semantics of a CPDP, we have to regard reset maps of transitions as objects that

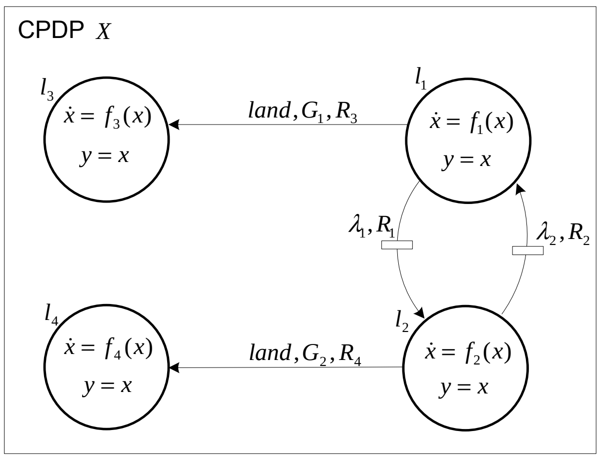

Figure 7.1: Landing aircraft modelled as CPDP

assign probability measures on the whole state space of a CPDP (rather than on the state spaces of the individual state variables). In order to define probability measures on the whole state space of a CPDP, we first need to formally define the state space of a CPDP and a topology on it such that it forms a Borel space. That is what we do in the rest of this section.

At every time instant t of the running time of a CPDP $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$, X has one current location $l \in L$ and one value for each of the state variables that are active at location l. The active state variables of location l are the elements of the set $\nu(l)$. This means that the state of a CPDP consists of a location and of a valuation for the active state variables of that location. We call the location the discrete part of the state and we call the valuation the continuous part of the state. The state, consisting of the discrete and the continuous part, is called the hybrid state of the CPDP. Suppose that at time t, the current location of X is l and suppose that $\nu(l) = \{v_1, v_2\}$. If variables v_1 and v_2 have values r_1 and r_2 at time t, then the valuation of X at time t equals $val := \{v_1 = r_1, v_2 = r_2\}$ and the hybrid state of X at time t equals (l, val). The state and output space of a CPDP are formally defined as follows.

Definition 7.3. Let X be a CPDP with location set L, state and output variables V and W and for each $l \in L$ active state and output variables $\nu(l) \subset V$ and $\omega(l) \subset W$. The state space or hybrid state space of X is defined as

$$\{(l, val) | l \in L, val \in vs(l)\},$$

where $vs(l)$ denotes the valuation space of location l, which in case $\nu(l) = \{v_1, v_2, \cdots, v_n\}$, is defined as

$$\{\{v_1 = r_1, v_2 = r_2, \cdots, v_n = r_n\} | r_1 \in \mathbb{R}^{d(v_1)}, r_2 \in \mathbb{R}^{d(v_2)}, \cdots, r_n \in \mathbb{R}^{d(v_n)}\}$$

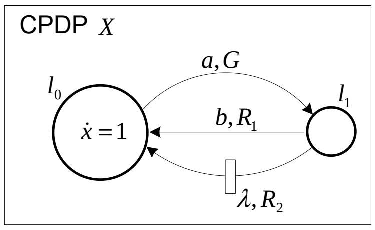

Figure 7.2: CPDP with an empty location

and in case $\nu(l) = \emptyset$ is defined as $\{0\}$. The output space of X is defined as

$$\cup_{l \in L} os(l),$$

where $os(l)$ denotes the output space of location l, which in case $\omega(l) = \{w_1, \cdots, w_m\}$ is defined as $\{\{w_1 = r_1, \cdots, w_m = r_m\} | \omega(l) = \{w_1, \cdots, w_m\}, r_i \in \mathbb{R}^{d(w_i)}\}$ and in case $\omega(l) = \emptyset$ is defined as $\{0\}$. The output value 0 is used for CPDP states where no output is defined.

Example 7.4. The state space of CPDP X of Figure 7.1 equals

$$\{(l, val) | l \in \{l_1, l_2, l_3, l_4\}, val \in \{\{x = r\} | r \in \mathbb{R}^6\}\}.$$

The output space of X equals $\{\{y = r\} | r \in \mathbb{R}^6\}$.

Remark 7.5. We allow that for CPDP locations l we have $\nu(l) = \emptyset$, i.e., we allow that locations do not have continuous variables attached to it. We call these locations *empty locations*. According to Definition 7.3, the valuation space of an empty location l equals $\{0\}$ and therefore l contributes one state to the state space of the CPDP; the state $(l, 0)$. The guard of an active transition α with origin location l is then equal to $\{0\}$, which means that at an empty location, active transitions are always enabled. Spontaneous transitions at empty locations assign a constant λ to the single state of the valuation space. This means that the jump time of such a spontaneous transition is exponentially distributed with parameter λ. A reset map of a transition whose target location is an empty location assigns probability one to the single state 0 of the valuation space of that empty location.

We also allow for CPDP locations that $\omega(l) = \emptyset$. This means that no output dynamics is defined for such locations. The output at states of these locations will later be defined as 0.

Example 7.6. Consider CPDP X in Figure 7.2. X has an empty location l_1, i.e., there are no continuous variables attached to location l_1. The reset map of the active transition from l_0 to l_1 is trivial, i.e., it assigns probability one to the state $(l_1, 0)$, and is therefore not pictured in Figure 7.2. The active transition from l_1 to l_0 has trivial guard $\{(l_1, 0)\}$ and is therefore also not pictured. The spontaneous transition assigns constant λ to state $(l_1, 0)$. For this CPDP we do not consider any output dynamics, therefore $\omega(l_0) = \omega(l_1) = \emptyset$.

We define a topology on the state space E of a CPDP X with location set L and variable set V such that E with this topology is a Borel space. Suppose that L contains n variables. Rename these variables, in arbitrary order, as v_1 till v_n. For each $l \in L$, let ι_l be the bijection from $vs(l)$ to $\mathbb{R}^{d(l)}$, where $d(l) := \sum_{v\in\nu(l)} d(v)$, that is defined as follows. Let $\nu(l) = \{v_{i_1}, v_{i_2}, \cdots, v_{i_m}\}$, where for $j = 1 \cdots m$ $i_j \in \{1, 2, \cdots, n\}$ and $i_k < i_l$ if $k < l$. Then

$$\iota_l(\{v_{i_1} = r_{i_1}, v_{i_2} = r_{i_2}, \cdots, v_{i_m} = r_{i_m}\}) = (r_{i_1}, r_{i_2}, \cdots, r_{i_m}),$$

which is an element of $\mathbb{R}^{d(l)}$. If $n = 0$ for location l, i.e., l has no continuous variables, then $vs(l) = \{0\}$ and we define the bijection $\iota_l : vs(l) \to \mathbb{R}^0$ as $\iota_l(0) = 0$. (Note that $\mathbb{R}^0$ equals $\{0\}$ by definition).

Now, we define a set $O \in vs(l)$ as *open* if $\iota_l(O)$ is an open set of $\mathbb{R}^{d(l)}$, where $\iota_l(O) := \{\iota_l(x)|x \in O\}$. We define a subset of E as open if it can be written as a union of sets of the form (l, O), where $l \in L$, O is an open set of $vs(l)$ and (l, O) is defined as $\{(l, val)|l \in L, val \in O\}$. The space $\{(l, x)|l \in L, x = \iota_l(val)$ for some $val \in vs(l)\}$ with the above topology is Borel isomorphic to a PDP space (see Chapter 2) with isomorphism $\iota((l, val)) := (l, \iota_l(val))$. Because a PDP space is a Borel space and ι is a Borel isomorphism, we have, by definition, that E with the topology defined above is a Borel space.

In almost the same way we can define the Borel sets of an output space E_O of a CPDP: order the output variables as $\{w_1, \cdots, w_n\}$. Let, for arbitrary $l \in L$, $\{w_{i_1}, \cdots, w_{i_m}\}$ be equal to $\omega(l)$ and let B be a Borel set of $\mathbb{R}^{d(w_{i_1})+\cdots+d(w_{i_m})}$. Then we define $\{\{w_{i_1} = r_1, \cdots, w_{i_m} = r_m\}|(r_1, \cdots, r_m) \in O\}$ to be a Borel set of E_O if O is a Borel set of $\mathbb{R}^{d(\omega_{i_1})+\cdots+d(\omega_{i_m})}$. If for a location l, $\omega(l) = \emptyset$, then this location contributes the Borel set $\{0\}$ to the Borel sets of E_O. Any countable union of these Borel sets is also defined to be a Borel set. With these sets E_O is a Borel space.

As we said before, in order to define the semantics of CPDPs, we need to transform reset maps such that they assign probability measures on the state space of the CPDP. Sometimes it also convenient to have similar transformations of the vector fields, the output mappings and the guards of transitions. Before we define these transformations, we first introduce the following notation.

Let $\alpha = (l, a, l', G, R)$ be an active transition. Then we we define the mappings *oloc* (origin location), *lab* (label), *tloc* (target location), *guard* and *rmap* (reset map) as: $oloc(\alpha) = l$, $lab(\alpha) = a$, $tloc(\alpha) = l'$, $guard(\alpha) = G$, $rmap(\alpha) = R$. These mappings, except for *guard*, are also defined in the same way for passive transitions. *oloc*, *tloc* and *rmap* are also defined in the same way for spontaneous transitions. Furthermore, let $\xi = (l, val)$ be some hybrid state. Then $loc(\xi) := l$ maps ξ to its discrete part, and $val(\xi) := val$ maps ξ to its continuous part.

Transformations and flow map

Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ be a CPDP with state space E and output space E_O. We define the elements F' and G' as: for all $\xi = (l, \{v_1 = r_1, \cdots, v_m = r_m\}) \in E$,

$$F'(\xi) := (l, \{v_1 = F(l, v_1)(r_1), \cdots, v_m = F(l, v_m)(r_m)\})$$

and in case $\nu(l) = \emptyset$ we define $F'(\xi) := (l, 0)$.

$$G'(\xi) := \{w_1 = G(l, w_1)(r_1, \cdots, r_m), \cdots, w_k = G(l, w_k)(r_1, \cdots, r_m)\},$$

where w_1 till w_k are the output variables of l, and in case $\omega(l) = \emptyset$ we define $G'(\xi) := 0$. Now F' is the 'vector field' on E that corresponds to F and G' is the output map from E to E_O that corresponds to G. Let R be a reset map of some transition α. Let $l = oloc(\alpha)$ and $l' = tloc(\alpha)$ and let $\{v_1, \cdots, v_m\} = \nu(l)$ and $\{v'_1, \cdots, v'_k\} = \nu(l')$. We define R' as follows. For all $\xi = (l, val)$ with $val = \{v_1 = r_1, \cdots, v_m = r_m\}$, such that $val \in guard(\alpha)$ and for all Borel sets $B_i \in \mathbb{R}^{d(v'_i)}$ $(i = 1 \cdots k)$,

$$R'(\xi)((l', B_1) \times (l', B_2) \cdots \times (l', B_k)) :=$$

$$R(val, v'_1)(B_1) R(val, v'_2)(B_2) \cdots R(val, v'_k)(B_k).$$

In case $\nu(tloc(\alpha)) = \emptyset$, we define $R'(\{0\}) = 1$. R' is uniquely extended to a probability measure on E. Note that for all $A \in \mathcal{B}(E)$ $R'(A) = R'(A \cap (l', vs(l')))$ because the probability that the target state lies in location l' is one. R' is a reset map on E that corresponds to R. Let the set S be the guard of a transition with origin location l. We define S', which is a subset of E, as $S' := \{(l, val) | val \in S\}$. S' is a subset/guard of E that corresponds to S.

For notational convenience we will often write F,G,R and S where we mean F',G',R' and S'. Whether we mean F or F', G or G', R or R', S or S' will be clear from the context.

We define the flow map $\phi : \mathbb{R}_+ \times E \to E$ of a CPDP $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ with state space E. $\phi(t, \xi)$ is determined by the differential equations

$$\dot{x}_1 = F(l, x_1), \quad \dot{x}_2 = F(l, x_2), \cdots \quad \dot{x}_n = F(l, x_n), \tag{7.1}$$

where $l = loc(\xi)$ and $\nu(l) = \{x_1, x_2, \cdots, x_n\}$. Thus, for $t \geq 0$ and $\xi = (l, \{x_1 = r_1, \cdots, x_n = r_n\}) \in E$, $\phi(t, \xi)$ equals $\xi' = (l, \{x_1 = r'_1, \cdots, x_n = r'_n\})$, where $r_1, \cdots, r_n$ are the solutions of (7.1) for $x_1 \cdots x_n$ at time t where $x_1 \cdots x_n$ at time zero have values $r_1 \cdots r_n$. For empty locations l we define the flow map as $\phi(t, (l, 0)) := (l, 0)$ for all $t \geq 0$.

7.1.2 IMC modelled as CPDP

The class of IMCs forms a subset of the class of CPDPs. The conversion of IMC $(L, l_0, Act, \mathcal{I}, \mathcal{M})$ to CPDP syntax results in the CPDP $(L, V, \nu, W, \omega, F, G, Act, \mathcal{A}, \mathcal{P}, \mathcal{S})$, with $V = W = \mathcal{P} = \emptyset$,

$$\mathcal{A} = \{\{l, a, l', G, R\} \mid (l, a, l') \in \mathcal{I}, G = \{0\}, R = R_{l'}\},$$

where $R_{l'}$ is the reset map that chooses state $(l', 0)$ with probability 1 and

$$\mathcal{S} = \{\{l, \lambda, l', R\} \mid (l, \lambda(0), l') \in \mathcal{I}, R = R_{l'}\}.$$

Note that λ is a mapping from $vs(l) = \{0\}$ to $\mathbb{R}_+$.

7.2 Semantics of CPDPs

Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ be a CPDP with state space E, flow map ϕ and initial state ξ_0. We define the semantics of X as the combination of a CFSJS and an NTS. Let X_C denote the CFSJS of X and let X_N denote the NTS of X. Then, $X_C = (E, \xi_0, \phi, \lambda, Q)$, where

$$\lambda(l, val) := \sum_{\alpha \in \mathcal{S}_{l\rightarrow}} \lambda_\alpha(val),$$

and for all $A \in \mathcal{B}(E)$,

$$Q(l, val)(A) = \sum_{\alpha \in \mathcal{S}_{l\rightarrow}} \frac{\lambda_\alpha(val)}{\lambda(l, val)} R_\alpha(A)$$

and $X_N = (E, \Sigma \cup \bar{\Sigma}, \mathcal{T})$, where

- $(\xi, a, m) \in \mathcal{T}$ if and only if there exists an $\alpha \in \mathcal{A}$ such that $lab(\alpha) = a$, $oloc(\alpha) = loc(\xi)$, $val(\xi) \in guard(\alpha)$ and $rmap(\alpha)(\xi) = m$.
- $(\xi, \bar{a}, m) \in \mathcal{T}$ if and only if there exists an $\alpha \in \mathcal{P}$ such that $lab(\alpha) = \bar{a}$, $oloc(\alpha) = loc(\xi)$ and $rmap(\alpha)(\xi) = m$.

Note that the CFSJS defined above expresses correctly that in each location there is a 'race' between the spontaneous transitions enabled at that location just as the 'race' between the spontaneous transitions of two CFSJSs that are running in parallel as described in Section 3.2.2. That the λ and Q of the CFSJS correctly express this 'race' can, mutatis mutandis, also be found in Section 3.2.2.

Example 7.7. The semantics of CPDP X of Figure 7.1 with state space E, flow map ϕ and initial state ξ_0 is as follows. $X_C = (E, \xi_0, \phi, \lambda, Q)$, where for $\xi = (l, val) \in E$,

$$\lambda(\xi) = \begin{cases} \lambda_1 & \text{if } l = l_1, \\ \lambda_2 & \text{if } l = l_2, \\ 0 & \text{if } l \in \{l_3, l_4\} \end{cases}$$

and for all $B \in \mathcal{B}(E)$

$$Q(\xi)(B) = \begin{cases} R_1(\xi)(B) & \text{if } l = l_1, \\ R_2(\xi)(B) & \text{if } l = l_2, \\ \text{undefined} & \text{if } l \in \{l_3, l_4\}. \end{cases}$$

$X_N = (E, \Sigma \cup \bar{\Sigma}, \mathcal{T})$, where

$$\mathcal{T} = \{(\xi, land, m) | \xi \in G_1, m = R_3(\xi)\} \cup \{(\xi, land, m) | \xi \in G_2, m = R_4(\xi)\}.$$

We cannot give the execution of a CPDP X with active/passive transitions in terms of a transition mechanism system, because it is not determined when transitions from $\mathcal{T}$ are executed. However, we can describe the execution of a general CPDP $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ as follows. Let $X_C = (E, \xi_0, \phi, \lambda, Q)$ be the CFSJS of X and let $X_N = (E, \Sigma, \mathcal{T})$ be the NTS of X. Then, the execution of X can be seen as the execution of X_C while at every state ξ the process has the potential to switch with measure m if $(\xi, \sigma, m) \in \mathcal{T}$ for some $\sigma \in \Sigma \cup \bar{\Sigma}$.

Output semantics

The CFSJS and NTS do not capture the complete behavior of a CPDP. At every state $\xi \in E$ of a CPDP, the CPDP also has an output value which lies in its output space E_O and which is determined by the output mapping $G : E \to E_O$. Therefore we could say that the complete behavior of a CPDP is captured by its CFSJS, its NTS and its output mapping.

7.3 Composition of CPDPs

First, we describe informally how two interacting CPDPs behave: suppose CPDPs X and Y are running simultaneously and suppose that $|_A^P|$ determines their interaction. This means that active transitions of X and Y with labels in A should synchronize, that passive transitions of X and Y with labels in P synchronize, etc. (See Chapter 4 for the interaction possibilities of $|_A^P|$). A synchronization of two active a-transitions is possible at some time t only when both X and Y have a-transitions, whose guards are both satisfied at time t. Given this informal description of interaction between CPDPs, it can be easily seen that the behavior of this composition can be captured as the CFSJS $X_C|Y_C$ together with the NTS $X_N|_A^P|Y_N$, where X_C and Y_C are the CFSJSs of X and Y, X_N and Y_N are the NTSs of X and Y. The composition operator $|$ for CFSJSs is defined in Chapter 3 and the composition operator $|_A^P|$ for NTSs is defined in Chapter 4.

The key result concerning composition of CPDPs is that we can define $|_A^P|$ in an operational way as a mapping from CPDPs X and Y to a new CPDP $X|_A^P|Y$, such that the semantics of $X|_A^P|Y$ is exactly equal to CFSJS $X_C|Y_C$ together with NTS $X_N|_A^P|Y_N$.

We now define $|_A^P|$ as a composition operator for CPDPs. After that we prove that the semantics $(X|_A^P|Y_C, X|_A^P|Y_N)$ of CPDP $X|_A^P|Y$ equals $(X_C|Y_C, X_N|_A^P|Y_N)$. Then we give two examples, illustrating the use of $|_A^P|$ in the context of CPDPs. In the definition of composition of CPDPs, communication is expressed through synchronization of transitions and not through the sharing of continuous variables. Therefore, each component should have its own continuous variables, i.e., the intersection of the sets of continuous variables of the two components should be empty. If this is not the case, then the two components are not compatible for composition.

Definition 7.8. Let $X = (L_X, V_X, \nu_X, W_X, \omega_X, F_X, G_X, \Sigma, \mathcal{A}_X, \mathcal{P}_X, \mathcal{S}_X)$ and $Y = (L_Y, V_Y, \nu_Y, W_Y, \omega_Y, F_Y, G_Y, \Sigma, \mathcal{A}_Y, \mathcal{P}_Y, \mathcal{S}_Y)$ be two CPDPs such that $V_X \cap V_Y = W_X \cap W_Y = \emptyset$. Then $X|_A^P|Y$ is defined as the CPDP $(L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$, where

- $L = \{l_1|_A^P|l_2 \mid l_1 \in L_X, l_2 \in L_Y\}$,
- $V = V_X \cup V_Y$, $W = W_X \cup W_Y$,
- $\nu(l_1|_A^P|l_2) = \nu(l_1) \cup \nu(l_2)$, $\omega(l_1|_A^P|l_2) = \omega(l_1) \cup \omega(l_2)$,
- $F(l_1|_A^P|l_2, v)$ equals $F_X(l_1, v)$ if $v \in \nu_X(l_1)$ and equals $F_Y(l_2, v)$ if $v \in \nu_Y(l_2)$.

- $G(l_1|_A^P|l_2, w)$ equals $G_X(l_1, w)$ if $w \in \omega_X(l_1)$ and equals $G_Y(l_2, w)$ if $w \in \omega_Y(l_2)$.
- $\mathcal{A}$, $\mathcal{P}$ and $\mathcal{S}$ are the least relations satisfying the rules r1, r2, r2', r3, r3', r4, r4', r5, r6, r6', r7 and r7', defined below

$$r1.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,G_1\times G_2,R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \in A).$$

$$r2.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,G_1\times vs(l_2),R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \notin A).$$

$$r2'.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,vs(l_1)\times G_2,R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \notin A).$$

$$r3.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a}}{\not\longrightarrow}}{l_1|_A^P|l_2 \overset{a,G_1\times vs(l_2),R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(a \notin A).$$

$$r3'.\frac{l_1 \overset{\bar{a}}{\not\longrightarrow}, l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,vs(l_1)\times G_2,Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(a \notin A).$$

$$r4.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1'}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(\bar{a} \notin P), \quad r4'.\frac{l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\bar{a},Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(\bar{a} \notin P)$$

$$r5.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(\bar{a} \in P).$$

$$r6.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a}}{\not\longrightarrow}}{l_1|_A^P|l_2 \overset{\bar{a},R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(\bar{a} \in P), \quad r6'.\frac{l_1 \overset{\bar{a}}{\not\longrightarrow}, l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\bar{a},Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(\bar{a} \in P)$$

$$r7.\frac{l_1 \overset{\lambda_1,R_1}{\longrightarrow} l_1'}{l_1|_A^P|l_2 \overset{\hat{\lambda}_1,R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}, \quad r7'.\frac{l_2 \overset{\lambda_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{\hat{\lambda}_2,Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'},$$

where $\hat{\lambda}_1$ and $\hat{\lambda}_2$ are defined as $\hat{\lambda}_1(\xi_1, \xi_2) := \lambda_1(\xi_1)$ and $\hat{\lambda}_2(\xi_1, \xi_2) := \lambda_2(\xi_2)$.

Theorem 7.9. *Let X and Y be two CPDPs with semantics (X_C, X_N) and (Y_C, Y_N) respectively. Let $(X|_A^P|Y_C, X|_A^P|Y_N)$ be the semantics of CPDP $X|_A^P|Y$. Then,*

$$(X|_A^P|Y_C, X|_A^P|Y_N) = (X_C|Y_C, X_N|_A^P|Y_N).$$

Proof. The proof is done in two steps. In the first step we prove that the CFSJS of $X|_A^P|Y$ is equal to $X_C|Y_C$. In the second step we prove that the NTS of $X|_A^P|Y$ is equals to $X_N|_A^P|Y_N$.

Step 1. We have that $X_C|Y_C = (E_X \times E_Y, (\xi_{X,0}, \xi_{Y,0}), (\phi_X, \phi_Y), \lambda, Q)$, where

$$\lambda(\xi_1, \xi_2) = \lambda_X(\xi_1) + \lambda_Y(\xi_2)$$

and

$$Q(\xi_1, \xi_2)(A_1 \times A_2) = \frac{\lambda_X(\xi_1)}{\lambda(\xi_1, \xi_2)} Q_X(\xi_1)(A_1) + \frac{\lambda_Y(\xi_2)}{\lambda(\xi_1, \xi_2)} Q_Y(\xi_2)(A_2).$$

The CFSJS of $X|_A^P|Y$ equals $(E_X \times E_Y, (\xi_{X,0}, \xi_{Y,0}), (\phi_X, \phi_Y), \tilde{\lambda}, \tilde{Q})$, where

$$\tilde{\lambda}(\xi_1, \xi_2) = \sum_{\alpha \in \mathcal{S}_{(\xi_1,\xi_2)\rightarrow}} \lambda_\alpha(\xi_1, \xi_2) \overset{r7,r7'}{=} \sum_{\alpha \in \mathcal{S}_{X,\xi_1\rightarrow}} \lambda_\alpha(\xi_1) + \sum_{\alpha \in \mathcal{S}_{Y,\xi_2\rightarrow}} \lambda_\alpha(\xi_2) =$$

$$\lambda_X(\xi_1) + \lambda_Y(\xi_2),$$

where $\overset{r7,r7'}{=}$ means that this step follows from composition rules $r7$ and $r7'$, and

$$\tilde{Q}(\xi_1, \xi_2)(A_1 \times A_2) = \sum_{\alpha \in \mathcal{S}_{(\xi_1,\xi_2)\rightarrow}} \frac{\lambda_\alpha(\xi_1, \xi_2)}{\lambda(\xi_1, \xi_2)} R_\alpha(\xi_1, \xi_2)(A_1 \times A_2) =$$

$$\frac{\lambda_X(\xi_1)}{\lambda(\xi_1, \xi_2)} \sum_{\alpha \in \mathcal{S}_{X,\xi_1\rightarrow}} \frac{\lambda_\alpha(\xi_1)}{\lambda_X(\xi_1)} R_\alpha(\xi_1)(A_1) + \frac{\lambda_Y(\xi_2)}{\lambda(\xi_1, \xi_2)} \sum_{\alpha \in \mathcal{S}_{Y,\xi_2\rightarrow}} \frac{\lambda_\alpha(\xi_2)}{\lambda_Y(\xi_2)} R_\alpha(\xi_2)(A_2) =$$

$$\frac{\lambda_X(\xi_1)}{\lambda(\xi_1, \xi_2)} Q_X(\xi_1)(A_1) + \frac{\lambda_Y(\xi_2)}{\lambda(\xi_1, \xi_2)} Q_Y(\xi_2)(A_2).$$

We found that the two CFSJSs agree on all sets of the form $A_1 \times A_2$, with A_1 and A_2 Borel sets. A π-system is by definition a set of subsets that is closed under finite intersections. Because $(A_1 \times A_2) \cap (A_1' \times A_2') = ((A_1 \cap A_1') \times (A_2 \cap A_2'))$, we have that the set of all Borel sets of the form $A_1 \times A_2$ is a π-system. Probability measures that agree on a π-system also agree on the σ-algebra generated by that π-system. The σ-algebra generated by all $A_1 \times A_2$, with A_1 and A_2 Borel sets, is exactly the Borel σ-algebra of $E_1 \times E_2$. Thus, the probability measures of the two CFSJSs agree on all Borel sets which means that the two CFSJSs are exactly the same.

Step 2. Let $X_N = (E_X, \Sigma \cup \bar{\Sigma}, T_X)$ and let $Y_N = (E_Y, \Sigma \cup \bar{\Sigma}, T_Y)$. Let the NTS of $X|_A^P|Y$ be denoted by $(E_X \times E_Y, \Sigma \cup \bar{\Sigma}, T)$ and let $X_N|_A^P|Y_N$ be denoted by $(E_X \times E_Y, \Sigma \cup \bar{\Sigma}, \tilde{T})$.

We have to prove that for every $\alpha \in T$ we have $\alpha \in \tilde{T}$ and vice versa. We prove this in four steps: 1. for a-transitions with $a \in A$, 2. for a-transitions with $a \notin A$, 3. for $\bar{a}$-transitions with $\bar{a} \in P$ and 4. for $\bar{a}$-transitions with $\bar{a} \notin P$.

$a \in A$: $((\xi_1, \xi_2), a, m_1 \times m_2) \in T \overset{r_1^C}{\Leftrightarrow}$ 'there exists $(l_1, a, l_1', G_1, R_1) \in \mathcal{A}_X$ such that $\xi_1 \in G_1$ and $R_1(\xi_1) = m_1$ and there exists $(l_2, a, l_2', G_2, R_2) \in \mathcal{A}_Y$ such

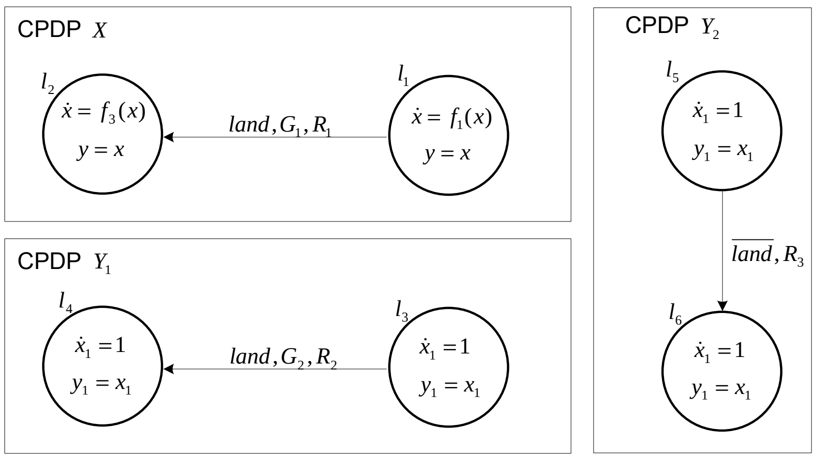

Figure 7.3: Landing aircraft and control tower modelled as interacting CPDPs

that $\xi_2 \in G_2$ and $R_2(\xi_2) = m_2$' $\Leftrightarrow$ '$(\xi_1, a, m_1) \in T_X$ and $(\xi_2, a, m_2) \in T_Y$' $\overset{r_1^N}{\Leftrightarrow}$ $((\xi_1, \xi_2), a, m_1 \times m_2) \in \tilde{T}$. Here r_1^C denotes rule r_1 from CPDP composition of Definition 7.8 and r_1^N denotes rule r_1 from NTS composition of Definition 4.6. The other three steps have the same structure $\alpha \in T \Leftrightarrow P_1 \Leftrightarrow P_2 \Leftrightarrow \alpha \in \tilde{T}$. In those steps we will not write down the P_1 part.

$a \notin A$: $((\xi_1, \xi_2), a, m_1 \times m_2) \in T \overset{r_2^C, r_2^{C'}, r_3^C, r_3^{C'}}{\Leftrightarrow}$ 'one of the following four cases is true: 1. $(\xi_1, a, m_1) \in T_X$ and $(\xi_2, \bar{a}, m_2) \in T_Y$, 2. $(\xi_1, \bar{a}, m_1) \in T_X$ and $(\xi_2, a, m_2) \in T_Y$, 3. $(\xi_1, a, m_1) \in T_X$ and $m_2 = Id$ and $\xi_2 \overset{\bar{a}}{\not\rightarrow}$, 4. $m_1 = Id$ and $\xi_1 \overset{\bar{a}}{\not\rightarrow}$ and $(\xi_2, a, m_2) \in T_Y$.' $\overset{r_2^N, r_2^{N'}, r_3^N, r_3^{N'}}{\Leftrightarrow}$ $((\xi_1, \xi_2), a, m_1 \times m_2) \in \tilde{T}$.

$\bar{a} \in P$: $((\xi_1, \xi_2), \bar{a}, m_1 \times m_2) \in T \overset{r_5^C, r_6^C, r_6^{C'}}{\Leftrightarrow}$ 'one of the following three cases is true: 1. $(\xi_1, \bar{a}, m_1) \in T_X$ and $(\xi_2, \bar{a}, m_2) \in T_Y$, 2. $(\xi_1, \bar{a}, m_1) \in T_X$ and $m_2 = Id$ and $\xi_2 \overset{\bar{a}}{\not\rightarrow}$, 3. $m_1 = Id$ and $\xi_1 \overset{\bar{a}}{\not\rightarrow}$ and $(\xi_2, \bar{a}, m_2) \in T_Y$.' $\overset{r_5^N, r_6^N, r_6^{N'}}{\Leftrightarrow}$ $((\xi_1, \xi_2), \bar{a}, m_1 \times m_2) \in \tilde{T}$.

$\bar{a} \notin P$: $((\xi_1, \xi_2), \bar{a}, m_1 \times m_2) \in T \overset{r_4^C, r_4^{C'}}{\Leftrightarrow}$ 'one of the following two cases is true: 1. $(\xi_1, \bar{a}, m_1) \in T_X$ and $m_2 = Id$, 2. $m_1 = Id$ and $(\xi_2, \bar{a}, m_2) \in T_Y$.' $\overset{r_4^N, r_4^{N'}}{\Leftrightarrow}$ $((\xi_1, \xi_2), \bar{a}, m_1 \times m_2) \in \tilde{T}$.

We found that all transitions of T are transitions of $\tilde{T}$ and vice versa. Consequently, the two NTSs are the same. □

Corollary 7.10. *$|_A^P|$ for CPDPs is commutative for all A and P. $|_A^P|$ for CPDPs is associative if and only if for all events a we have: $a \notin A \Rightarrow \bar{a} \in P$.*

This result follows directly from Theorems 4.4 and 7.9.

Example 7.11. CPDP X of Figure 7.3 models a flying aircraft and has initial state $\xi_{X,0} = (l_1, x_0)$. Note that, for reasons of simplicity, the non-nominal locations from Figure 7.1 are not modelled here. CPDP Y_1 of Figure 7.3 models a control tower at an airport that can communicate with the aircraft modelled by X. Location l_3 is the location where Y_1 waits for a signal from X. The dynamics of l_3 is a clock dynamics expressing the time that Y has to wait before X sends a signal. Therefore, the initial valuation of initial location l_3 equals $\{x_1 = 0\}$. Location l_4 is the location where Y_1 'knows' that X has send a signal. The dynamics of this location is again a clock dynamics. If Y_1 enters location l_4, then the timer is reset to zero, which means that reset map R_2 assigns for each value of x_1 in l_3 probability one to the Borel set $\{\{x_1 = 0\}\}$.

We connect X and Y_1 via composition operator $|_A^P|$, where $A = \{land\}$ (P is not relevant here). This means that the signal/label *land* is used as a shared synchronization action between X and Y_1. Then, Y_1 can execute the *land* transition only when at the same time X executes its *land* transition. We want to model that X can execute its *land* transition independently from Y. Once this happens, this transition should be communicated to Y. We can express this via the guards G_1 and G_2. G_1 equals G_1 from example 7.2, expressing that this switch may happen as soon as the altitude of the aircraft drops under a certain level h. G_2 equals the whole valuation space of location l_3. This expresses that this transition can always be taken and consequently it expresses that this transition cannot block the *land* transition of X. We assume maximal progress. Then, the synchronized *land* transition is executed as soon as guard G_1 is satisfied. After the synchronized *land* transition, Y_1 is in location l_4. We could say that the information 'X switched to landing mode', which is received by Y_1, is stored in the discrete component of the hybrid state of Y_1. In other words, discrete state l_4 of Y_2 has the meaning 'X is in landing mode'.

CPDP $X|_A^P|Y_1$, which expresses the composite system of X interconnected with Y_1, is pictured in Figure 7.4. According to composition rule r1, the guard G_3 equals $G_1 \times G_2$ and the reset map R_4 equals $R_1 \times R_2$. If we look at the behavior of CPDP $X|_A^P|Y_1$ under maximal progress, then we will see that this CPDP indeed expresses the communication from X to Y_1 that we wanted to model: the initial hybrid state of $X|_A^P|Y_1$ equals $(l_1|l_3, \{x = x_0, x_1 = 0\})$. The continuous state variables x and x_1 evolve along vector fields f_1 and 1 respectively until guard G_3 is satisfied. G_3 is satisfied when the vertical position of x reaches the level h. Then, the *land* transition is executed and the state variables x and x_1 are reset by R_4, which means that x is reset by R_1 and x_1 is reset by R_2. Thus, we see that at the moment that X switches to landing mode, Y_1 switches to location l_4, which indeed establishes the communication that we intended.

Now we show how the aircraft/control tower system can be modelled by using a passive transition. For this example, we think modelling the communication with a passive transition is more natural, since there is a clear distinction between an active system (the aircraft which sends the information of the switch) and a passive system (the control tower which receives the information). Now, the control tower is modelled as the CPDP Y_2 of Figure 7.3. Y_2 is exactly the same as Y_1, except that the active transition is replaced by a passive transition with label $\overline{land}$. This

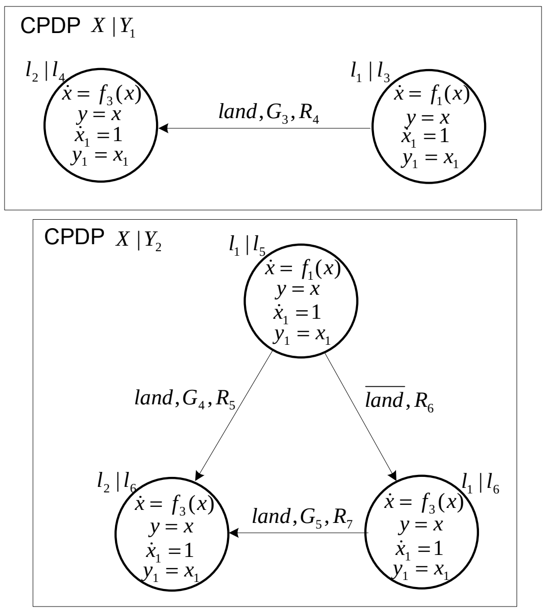

Figure 7.4: Composite CPDP of landing aircraft and control tower

passive transition expresses that as soon as a *land* signal is received (from X), the passive transition is executed and reset map R_3 (whose action equals the action of R_2) resets the timer x_1 to zero. Since *land* is not a synchronization action here, we connect X and Y_2 via $|^P_A|$, where $A = \emptyset$ (P is not relevant). The resulting CPDP $X|^P_A|Y_2$ is pictured in Figure 7.4. It can be seen from rule r2, that guard G_4 is equal to G_3 and reset map R_5 is equal to R_4. This means that as far as locations $l_1|l_3$ and $l_2|l_4$ / $l_1|l_5$ and $l_2|l_6$ are concerned, $X|^P_A|Y_1$ and $X|^P_A|Y_2$ have the same behavior. The difference between $X|^P_A|Y_1$ and $X|^P_A|Y_2$ lies in the fact that $X|^P_A|Y_2$ can switch to location $l_1|l_6$ via a passive transition, while $X|^P_A|Y_1$ cannot do this. The meaning of this switch to $l_1|l_6$ becomes apparent in a composition context with more than two components. We refer to Chapter 4 for an explanation of this.

Example 7.12. In Figure 7.5, the repair shop system, which was modelled as a PDP in Section 6.4, is modelled as the composition of CPDPs M_1,M_2 and R. CPDPs M_1 and M_2 model the two machines and CPDP R models the repair shop. M_1 initially starts in location $l_{1,0}$ with a clock dynamics for its state variable x_1. M_1 can break down with state dependent jump rate λ_1. This is modelled by the spontaneous transition to $l_{1,1}$. $l_{1,1}$ is an empty location, therefore the spontaneous transition to $l_{1,1}$ has a trivial reset map that assigs probability 1 to state $(l_{1,1}, 0)$. This reset map is not pictured in Figure 7.5. From $l_{1,1}$ an active transition is executed to $l_{1,3}$, with label *down*. We want to model that this *down*-transition is executed immediately after location $l_{1,1}$ is reached. Then, the *down* signal is

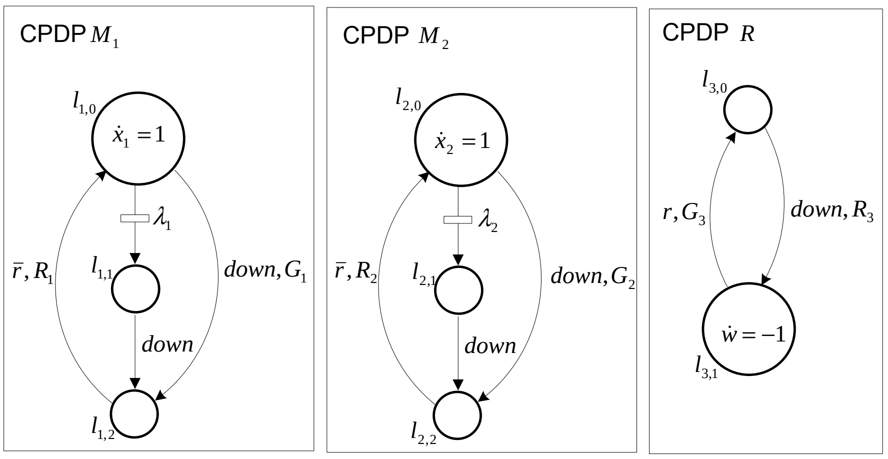

Figure 7.5: CPDP model of the repair shop system

executed exactly when M_1 breaks down. In the next chapter we will see that with 'maximal progress' M_1 indeed models that no time is consumed in location $l_{1,1}$. If the machine does not break down via the spontaneous transition before s_1 time units, i.e., the maximal age of the machine, then the machine should be taken out of order to the repair shop. This is modelled by the *down*-transition from $l_{1,0}$ to $l_{1,2}$ with guard G_1 equal to $x_1 \geq s_1$. In location $l_{1,2}$, machine M_1 waits for an r signal. This is expressed by the passive $\bar{r}$-transition. This r signal will be sent by the repair shop, indicating that the machine has been repaired. Reset map R_1 resets state variable x_1 to zero, which expresses that the machine starts 'brand new'. Machine M_2 is modelled likewise.

The repair shop CPDP R starts in empty location $l_{3,0}$. Here it waits until one of the machines needs to be repaired. The switch to repair mode $l_{3,1}$ is modelled by the active *down*-transition. We define *down* to be a synchronization action and therefore this *down*-transition synchronizes with either a *down*-transition of M_1 or a *down*-transition of M_2. Due to this synchronization, R switches to repair mode $l_{3,1}$ exactly when one of the machines need to be repaired. Reset map R_3 resets state variable w with a uniform distribution on the interval $[t_1, t_2]$, determining the time needed to repair the machine. In $l_{3,1}$, w counts down to zero, expressed by the dynamics $\dot{w} = -1$. If w has been counted down to zero, R switches back to $l_{3,0}$ where it waits for a new machine to be repaired. This switch is modelled by the active r transition. The guard G_3 of this transition equals $w = 0$. The passive $\bar{r}$ transitions of M_1 and M_2 can synchronize with this active r-transition, therefore these passive $\bar{r}$-transitions are executed exactly when the machine is repaired.

From the description above, we get that *down* is an interleaving action between M_1 and M_2, *down* is a synchronization action between R and M_1 or M_2, and r is an interleaving action between R and M_1 or M_2. The passive action $\bar{r}$ may be chosen interleaving or synchronizing. This choice does not influence the behavior since M_1 and M_2 will never visit their locations $l_{1,2}$ and $l_{2,2}$ at the same time (i.e., joint location $(l_{1,2}, l_{2,2})$ is never reached). This gives that the total repair shop system is modelled as the CPDP

$$(M_1|_{\emptyset}^{\emptyset}|M_2)|_{down}^{\emptyset}|R.$$

Since M_1 and M_2 both have three locations, and R has two locations, the number of locations of the composite CPDP, which is the number of product locations, equals eighteen. This is much more than the five locations of the PDP that models the same system (see Section 6.4). However, the composite CPDP has exactly five (joint) locations where time is consumed: $(l_{1,0}, l_{2,0}, l_{3,0})$,$(l_{1,0}, l_{2,2}, l_{3,1})$, $(l_{1,2}, l_{2,0}, l_{3,1})$, $(l_{1,2}, l_{2,1}, l_{3,1})$ and $(l_{1,1}, l_{2,2}, l_{3,1})$, which correspond to the locations l_1 till l_5 of the PDP of Section 6.4. All the other joint locations are either never visited or visited as an intermediate location between two time-consuming locations, which means that as soon as one of these intermediate locations is visited a switch happens immediately to another location. In the next chapter we will see that if we convert the composite CPDP to a PDP, the intermediate locations and the locations that are unreachable will fall away, resulting in a PDP with exactly these five locations.

Remark 7.13. It can be easily checked that composition of CPDPs that are transformations of IMCs as in Section 7.1.2, coincides with composition for IMCs as defined in Chapter 5. With this we mean that for IMCs X and Y and their CPDP transformations X_C and Y_C we get that the CPDP transformation of $X|A|Y$ is equal to $X_C|_A^P|Y_C$ for arbitrary $P \subset \bar{\Sigma}$. P does not play a role in this CPDP composition because IMCs (and their CPDP transformations) do not have passive transitions, therefore this result holds for arbitrary P.

7.3.1 Scope operator for CPDP

We generalize the scope operator $[\cdot]_C$, as defined in Definition 4.3, to the context of CPDPs as follows.

Definition 7.14. Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ be a CPDP and let $C \subset \bar{\Sigma}$. $[X]_C$ is defined as the CPDP $(L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \tilde{\mathcal{P}}, \mathcal{S})$, where $\tilde{\mathcal{P}}$ equals $\{\alpha \in \mathcal{P} | lab(\alpha) \notin C\}$.

7.4 Value passing CPDPs

In this section we extend the CPDP model to *value passing CPDPs*. For CPDPs, interaction is established through synchronization of transitions. This means that the information that one CPDP can obtain concerning other CPDPs in the composition is, first, which active actions are executed and, second, at which times these actions are executed. For example, via a passive $\bar{a}$-transition, a CPDP 'knows' when another CPDP executes an a-transition. With value passing CPDPs we extend the CPDP model such that it is possible for one CPDP to obtain information about the values of the output variables of other CPDPs. The moments where this information is communicated from one CPDP to another CPDP are the moments where the transitions synchronize. In other words, this communication of output information is expressed through synchronization of transitions. This idea of passing values through synchronizing transitions is called *value passing* in the literature and has been developed for example for the specification languages LOTOS [BB87, LL92] and CSP [Hoa85].

This section is organized as follows. First we define the value passing CPDP model and we give the CFSJS/NTS semantics of a value passing CPDP. Then we define the composition operator $|_A^P|$ for value passing CPDPs. As in the case of CPDPs, we will see that the behavior of two interacting value passing CPDPs X and Y is equal to the CFSJS and NTS of the value passing CPDP $X|_A^P|Y$. Finally, we give some examples illustrating the expressiveness of value passing in the context of value passing CPDPs.

7.4.1 Definition and semantics of value passing CPDPs

Definition 7.15. A value-passing CPDP is a tuple $(L, V, W, \nu, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$, where all elements except $\mathcal{A}$ are defined as in Definition 7.1 and where $\mathcal{A}$ is a finite set of active transitions that consists of six-tuples (l, a, l', G, R, vp), denoting a transition from location $l \in L$ to location $l' \in L$ with communication label $a \in \Sigma$, guard G, reset map R and value-passing element vp. G is a subset of the valuation space of l. vp can be equal to either $!Y$, $?U$ or $\emptyset$. For the case $!Y$, Y is an ordered tuple $(w_1, w_2, \cdots, w_m)$ where $w_i \in w(l)$ for $i = 1 \cdots m$. If for a transition we have $vp =!Y$ for some Y, then this means that in a synchronization with other transitions, this transition passes the values of the variables in Y to the other transition. For the case $?U$, we have $U \subset \mathbb{R}^n$ for some $n \in \mathbb{N}$. If for a transition we have $vp =?U$, then this means that in a synchronization with another transition that has $vp =!Y$, this transition receives the values from the variables of Y as long as these values are contained in the set U. If the other transition wants to pass values that do not lie in U, then the synchronization will not take place, i.e., it is blocked by U. If a transition is not used for value passing (either output $!Y$ or input $?U$), then this transition has $vp = \emptyset$. The reset map R assigns to each point in $G \times U$ (for the case $vp =?U$) or to each point in G (for the cases $vp =!Y$ and $vp = \emptyset$) for each state variable $v \in \nu(l')$ a probability measure on $\mathbb{R}^{d(v)}$. Active transitions α with $\omega(oloc(\alpha)) = \emptyset$, i.e., whose origin locations have no continuous variables, have value passing element $vp = \emptyset$.

Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ be a value passing CPDP with state space E, flow map ϕ and initial state ξ_0. We define the CFSJS and NTS semantics of X. Let X_C be the CFSJS of X and let X_N be the NTS of X. X_C is defined as in Section 7.2. $X_N = (E, \Sigma^{vp} \cup \Sigma \cup \bar{\Sigma}, \mathcal{T})$, where

- $\Sigma^{vp} := \{(a, r) | a \in \Sigma, r \in \mathbb{R}^n \text{ for some } n \in \mathbb{N}\}$,
- $(\xi, a, m) \in \mathcal{T}$ if and only if there exists an $\alpha \in \mathcal{A}$ such that $lab(\alpha) = a$, $oloc(\alpha) = loc(\xi)$, $val(\xi) \in guard(\alpha)$, $rmap(\alpha)(\xi) = m$ and $vp(\alpha) = \emptyset$,

- $(\xi, (a, r), m) \in \mathcal{T}$, with $r \in \mathbb{R}^n$, if and only if there exists an $\alpha \in \mathcal{A}$ such that $lab(\alpha) = a$, $oloc(\alpha) = loc(\xi)$ and $val(\xi) \in guard(\alpha)$ and
    1. $vp(\alpha) = !(w_1, \cdots, w_k)$, where $(G(loc(\xi), w_1)(val(\xi)), \cdots, G(loc(\xi), w_k)(val(\xi))) = r$ (i.e., the output for w_1 at ξ equals r) and $rmap(\alpha)(\xi) = m$, or
    2. $vp(\alpha) = ?U$ and $r \in U$ and $rmap(\alpha)(\xi, r) = m$
- $(\xi, \bar{a}, m) \in \mathcal{T}$ if and only if there exists an $\alpha \in \mathcal{P}$ such that $lab(\alpha) = \bar{a}$, $oloc(\alpha) = loc(\xi)$ and $rmap(\alpha) = m$.

Example 7.16. Let X be a CPDP with one location l. At l there is continuous dynamics $\dot{x} = 1$ and the output map equals $y = x$. There is one active transition $\alpha = (l, a, l, G, R, vp)$ with guard G satisfied if $x \geq 1$, with reset map $R(\xi)(\{x = 0\}) = 1$, i.e., R resets x to 0 at all states $\xi = (l, \{x = r\})$ with $r \geq 1$, and with value passing element $vp = !y$.

The NTS of X, whose state space we denote by E, equals $(E, \Sigma^{vp} \cup \Sigma \cup \bar{\Sigma}, \mathcal{T})$ with $\Sigma = \{a\}$, $\Sigma^{vp} = \{(a, r) | r \in \mathbb{R}^n$ for some $n \in \mathbb{N}\}$ and

$$\mathcal{T} = \{((l, \{x = r\}), (a, r), m) | r \geq 1, m = Id(l, \{x = 0\})\}.$$

If we have $vp = ?U$ for some $U \subset \mathbb{R}$ instead of $vp = !y$ and we have $R(\xi, r) = Id(\{x = r\}$, then we get

$$\mathcal{T} = \{((l, \{x = r\}), (a, r'), m) | r \geq 1, r' \in U, m = Id(l, \{x = r'\})\}.$$

In the latter case we have that the NTS has for states $\xi \in G$ for all $r' \in U$ a transition with label (a, r'). If another CPDP Y outputs value $r' \in U$ through an a-transition, then the NTS of Y has a transition with label (a, r'). In the NTS composition of the NTSs of X with Y, these (a, r') transitions synchronize, which expresses that X accepts the output r' of Y. X then resets its state to r' as expressed by the reset measure $Id(l, \{x = r'\})$. This idea of composition of value passing CPDPs is formally defined in the next section.

7.4.2 Composition of value passing CPDPs

Definition 7.17. Let $X = (L_X, V_X, \nu_X, W_X, \omega_X, F_X, G_X, \Sigma, \mathcal{A}_X, \mathcal{P}_X, \mathcal{S}_X)$ and $Y = (L_Y, V_Y, \nu_Y, W_Y, \omega_Y, F_Y, G_Y, \Sigma, \mathcal{A}_Y, \mathcal{P}_Y, \mathcal{S}_Y)$ be two value passing CPDPs such that $V_X \cap V_Y = W_X \cap W_Y = \emptyset$. Then $X|_A^P|Y$ is defined as the CPDP $(L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$, where L, V, ν, W, ω, F, G, Σ, $\mathcal{P}$ and $\mathcal{S}$ are defined as in Definition 7.8 and $\mathcal{A}$ is the least relation satisfying the rules

$$r1. \frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,G_1 \times G_2, R_1 \times R_2}{\longrightarrow} l_1'|_A^P|l_2'} (a \in A).$$

$$r2. \frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a},R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,G_1 \times vs(l_2), R_1 \times R_2}{\longrightarrow} l_1'|_A^P|l_2'} (a \notin A).$$

$$r2'.\frac{l_1 \overset{\bar{a},R_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,vs(l_1)\times G_2,R_1\times R_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \notin A).$$

$$r3.\frac{l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1', l_2 \overset{\bar{a}}{\not\longrightarrow}}{l_1|_A^P|l_2 \overset{a,G_1\times vs(l_2),R_1\times Id}{\longrightarrow} l_1'|_A^P|l_2}(a \notin A).$$

$$r3'.\frac{l_1 \overset{\bar{a}}{\not\longrightarrow}, l_2 \overset{a,G_2,R_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,vs(l_1)\times G_2,Id\times R_2}{\longrightarrow} l_1|_A^P|l_2'}(a \notin A).$$

$$r1data.\frac{l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1', l_2 \overset{a,G_2,R_2,v_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,G_1|G_2,R_1\times R_2,v_1|v_2}{\longrightarrow} l_1'|_A^P|l_2'}(a \in A, v_1|v_2 \neq \bot).$$

$$r2data.\frac{l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1'}{l_1|_A^P|l_2 \overset{a,G_1\times val(l_2),R_1\times Id,v_1}{\longrightarrow} l_1'|_A^P|l_2}(a \notin A).$$

$$r2data'.\frac{l_2 \overset{a,G_2,R_2,v_2}{\longrightarrow} l_2'}{l_1|_A^P|l_2 \overset{a,val(l_1)\times G_2,Id\times R_2,v_2}{\longrightarrow} l_1|_A^P|l_2'}(a \notin A),$$

where $l_1 \overset{a,G_1,R_1,v_1}{\longrightarrow} l_1'$ means $(l_1, a, l_1', G_1, R_1, v_1) \in \mathcal{A}_X$ with $v_1 \neq \emptyset$, and $l_1 \overset{a,G_1,R_1}{\longrightarrow} l_1'$ means $(l_1, a, l_1', G_1, R_1, \emptyset)$ and $v_1|v_2$ is defined as:

- $v_1|v_2 := !Y$ if $v_1 = !Y$ and $v_2 := ?U$ and dim(U)=dim(Y) or if $v_2 = !Y$ and $v_1 := ?U$ and dim(U)=dim(Y),
- $v_1|v_2 := ?(U_1 \cap U_2)$ if $v_1 = ?U_1$ and $v_2 = ?U_2$ and dim(U_1)=dim(U_2),
- $v_1|v_2 := \bot$ otherwise, where $\bot$ means that v_1 and v_2 are not compatible.

Furthermore, $G_1|G_2$ is, only when $v_1|v_2 \neq \bot$, defined as:

- $G_1|G_2 := (G_1 \cap U) \times G_2$ if $v_1 = !Y$ and $v_2 = ?U$,
- $G_1|G_2 := G_1 \times (G_2 \cap U)$ if $v_1 = ?U$ and $v_2 = !Y$,
- $G_1|G_2 := G_1 \times G_2$ if $v_1 = ?U_1$ and $v_2 = ?U_2$.

Here we define $G \cap U$ as the set of all states in G whose output values lie in U.

Theorem 7.18. *Let X and Y be two value passing CPDPs with semantics (X_C, X_N) and (Y_C, Y_N) respectively. Let $(X|_A^P|Y_C, X|_A^P|Y_N)$ be the semantics of value passing CPDP $X|_A^P|Y$. Assume that there do not exist value-passing transitions $(l_1, a, l_1', G_1, R_1, !(w_1, \cdots, w_k)) \in \mathcal{A}_X$ and $(l_2, a, l_2', G_2, R_2, !(\tilde{w}_1, \cdots, \tilde{w}_l)) \in \mathcal{A}_Y$ such that $a \in A$ and there exist $\xi_1 \in G_1$ and $\xi_2 \in G_2$ such that $(G_X(\xi_1, w_1), \cdots, G_X(\xi_1, w_k)) = (G_Y(\xi_2, \tilde{w}_1), \cdots, G_Y(\xi_2, \tilde{w}_l))$. Then,*

$$(X|_A^P|Y_C, X|_A^P|Y_N) = (X_C|Y_C, X_N|_A^P|Y_N).$$

Remark 7.19. The assumption in Theorem 7.18 says that there may not be two value passing output transitions with the same label (in A) and with the same output value for some states. Rule $r1data$ expresses that two value passing output transitions can not synchronize, which is in line with the philosophy that at any moment only one component can determine the output, while multiple components may receive this value via value passing input transitions. If the assumption is not satisfied, then on the level of composition of NTSs, there will be synchronized transition that comes from these two output transitions, while the NTS of the composition does not have this synchronized transition because of rule $r1data$.

Proof. The CFSJS part and the NTS part concerning non-value-passing transitions, is already proven in the proof of Theorem 7.9. This means that we only have to prove the NTS part concerning the rules $r1data$,$r2data$ and $r2data'$. We do the proof in two steps. The first step concerns a-transitions with $a \in A$ and the second step concerns a-transitions with $a \notin A$.

$a \in A$: $((\xi_1, \xi_2), (a, r), m_1 \times m_2) \in T \overset{r1data}{\Leftrightarrow}$ 'one of the following two cases is true

1. there exists $(l, a, l', \tilde{G}, R, !(\tilde{w}_1, \cdots, \tilde{w}_l))$ such that $(\xi_1, \xi_2) \in \tilde{G}$ and $(G((\xi_1, \xi_2), \tilde{w}_1), \cdots, G((\xi_1, \xi_2), \tilde{w}_l)) = r$ and $R(\xi_1, \xi_2) = m_1 \times m_2$
2. there exists $(l, a, l', \tilde{G}, R, ?U)$ such that $(\xi_1, \xi_2) \in \tilde{G}$ and $r \in U$ and $R((\xi_1, \xi_2), r) = m_1 \times m_2$'

$\Leftrightarrow$ 'one of the following three cases is true

1. there exist $(l_1, a, l_1', G_1, R_1, !(w_1, \cdots, w_k)) \in \mathcal{A}_X$ and $(l_2, a, l_2', G_2, R_2, ?U) \in \mathcal{A}_Y$ such that $\xi_1 \in G_1$, $\xi_2 \in G_2$, $(G_X(\xi_1, w_1), \cdots, G_X(\xi_1, w_k)) = r$, $r \in U$, $R_1(\xi_1) = m_1$ and $R_2(\xi_2, r) = m_2$
2. there exist $(l_2, a, l_2', G_2, R_2, !(w_1, \cdots, w_k)) \in \mathcal{A}_Y$ and $(l_1, a, l_1', G_1, R_1, ?U) \in \mathcal{A}_X$ such that $\xi_1 \in G_1$, $\xi_2 \in G_2$, $(G_Y(\xi_2, w_1), \cdots, G_Y(\xi_2, w_k)) = r$, $r \in U$, $R_1(\xi_1) = m_1$ and $R_2(\xi_2, r) = m_2$.
3. there exist $(l_1, a, l_1', G_1, R_1, ?U_1) \in \mathcal{A}_X$ and $(l_2, a, l_2', G_2, R_2, ?U_2) \in \mathcal{A}_Y$ such that $\xi_1 \in G_1$, $\xi_2 \in G_2$, $r \in U_1$, $r \in U_2$, $R(\xi_1, r) = m_1$ and $R(\xi_2, r) = m_2$'

$\overset{as.}{\Leftrightarrow}$ '$(\xi_1, (a, r), m_1) \in T_X$ and $(\xi_2, (a, r), m_2) \in T_Y$' $\overset{r1^N}{\Leftrightarrow}$ $((\xi_1, \xi_2), (a, r), m_1 \times m_2) \in \tilde{T}$. Here $\overset{as.}{\Leftrightarrow}$ means that we use the assumption from the Theorem in this step.

$a \notin A$: $((\xi_1, \xi_2), (a, r), m_1 \times m_2) \in T \overset{r2data,r2data'}{\Leftrightarrow}$ 'one of the following two cases is true

1. $(\xi_1, (a, r), m_1) \in T_X$ and $m_2 = Id$
2. $(\xi_2, (a, r), m_2) \in T_Y$ and $m_1 = Id$

$\overset{r3^N,r3^{N'}}{\Leftrightarrow}$ $((\xi_1, \xi_2), (a, r), m_1 \times m_2) \in \tilde{T}$. Note that for value-passing transitions rules $r3^N$ and $r3^{N'}$ do on the NTS level what rules $r2data$ and $r2data'$ do on the CPDP level. This is because there do not exist passive transitions with labels of the form (a, r). □

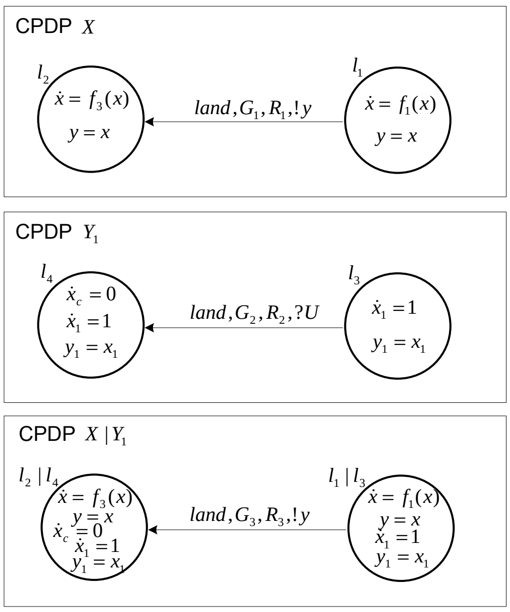

Figure 7.6: Value passing CPDPs

Corollary 7.20. *$|^P_A|$ for value passing CPDPs is commutative for all A and P. $|^P_A|$ for value passing CPDPs is associative if and only if for all events a we have: $a \notin A \Rightarrow \bar{a} \in P$.*

This result follows directly from Theorems 4.4 and 7.18.

Example 7.21. In Figure 7.6 we see the value passing CPDPs X and Y_1. X and Y are the same as the X and Y_1 of Figure 7.3, except that here the active transitions are value passing active transitions. More specific, at the moment of switching to landing mode, the aircraft X sends the value of its state (position and velocity) to the control tower Y_1.

Sending the state information is modelled as the value $!y$ for the value passing part of the transition. y is the only output variable of X and is a copy of x and contains therefore the exact information of the state. The value passing element of the transition in Y_1 equals $?U$, where $U = \mathbb{R}^6$. This means that this transition can receive all six dimensional real values. Note that if we would have $r \notin U$ for some $r \in \mathbb{R}^6$, then the transition of X would be blocked at state $\{x = r\}$.

Location l_4 of Y_1 has the new state variable x_c. This variable is used to store the information received from X. At the moment that X switches to l_2, Y_1 will switch to l_4 and the value of y, communicated by X, will be stored in x_c. Storing received data is done via the reset maps and in the case of Figure 7.6 it is expressed as $R(\{x_1 = r_1, x = r_2\})(\{\{x_1 = r_1, x_c = r_2\}\}) = 1$. Note that this indeed expresses that x_1 will not change by the switch and x_c holds the value of y after the switch.

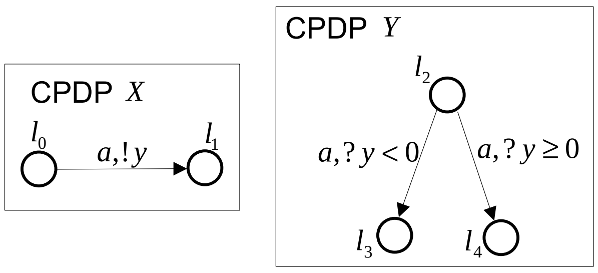

Figure 7.7: Value passing used to express scheduling

7.4.3 Expressiveness of value passing

In Example 7.21 we have seen that value passing can express sending/receiving of the value of output variables. There are more types of communication that can be expressed by using value passing. We give two more examples which show two more types of communication: scheduling via value passing, constraint conjunction via value passing.

Example 7.22. In this example we show how one CPDP can schedule transitions of another CPDP. In Figure 7.7, we see two CPDPs, X and Y, which are pictured without the details concerning state/output dynamics, guards and reset maps. CPDP X can switch from location l_0 to location l_1. With this switch, the value of output variable y is communicated over channel a. This value of y can be received by Y at initial location l_2. Y uses this information to schedule its two transitions at location l_2. If the value of y is smaller than zero, then the transition to location l_3 is taken, otherwise the transition to location l_4 is taken. In Figure 7.7, $?y < 0$ actually stands for $?\{r \in \mathbb{R} | r < 0\}$, and $?y \geq 0$ stands for $?\{r \in \mathbb{R} | r \geq 0\}$. In fact, these value passing transitions of Y can receive any one dimensional value that is communicated over channel a. This means that, if we compose Y with a component that sends at some time the value of some one dimensional variable y_2 over channel a, then Y can receive this value. We specifically write $y < 0$, to clarify that we intend that this transition is used to receive values of variable y of CPDP X.

In the composition $X|_A^P|Y$, with $A = \{a\}$ and P not relevant since no passive transitions are involved, X schedules the transitions of Y through the values of y. This method can, for example, be applied to systems where one component can perform different strategies, while the specific strategy that is chosen depends on the output variables of some other component.

Example 7.23. In Figure 7.8, CPDP X can switch from initial location l_0 to location l_1. The guard of this transition (not pictured) equals the whole valuation space of l_0. If X would be executed as a stand alone CPDP, it would, because of maximal progress, switch immediately to l_1. In this example we show how other

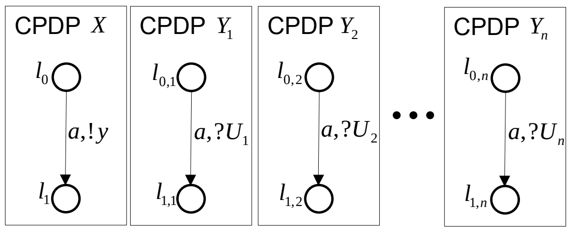

Figure 7.8: Value passing used for constraint conjunction

CPDPs, Y_1 till Y_n, can independently put constraints on the execution time of the active transition of X. For $i = 1 \cdots n$, U_i is the constraint put by CPDP Y_i on the execution time of the transition of X. Let y be one dimensional. Then, if $n = 2$, U_1 equals $y \geq -1$ and U_2 equals $y \leq 1$, then in $X|_A^P|Y_1|_A^P|Y_2$, with $A = \{a\}$ and P not relevant, the guard on the a-transition from location $l_0|l_{0,1}|l_{0,2}$ to location $l_1|l_{1,1}|l_{1,2}$ is equal to the part of the valuation space where $y \in [-1, 1]$.

7.5 Summary

In this section we summarize some aspects of the CPDP model. We describe which types of systems and which types of communication between those systems can be captured with the theory of this chapter.

A CPDP models a system with multiple locations. In each location, the continuous state of the CPDP has dynamics determined by some ordinary differential equation. The CPDP can jump from one location to another by means of a spontaneous transition or by means of a non-deterministic (or active) transition. A spontaneous transition is determined by some probability distribution. A non-deterministic (or active) transition can happen only if the continuous state lies inside the guard of that transition. However, if the process enters the guard of an active transition, then the process is not forced to execute the transition, but it is allowed to execute the transition. When exactly the active transition is executed, is not determined. In the next chapter we introduce the *maximal progress* assumption. Under this assumption, an active transition is *forced* to be executed as soon as its guard becomes satisfied. This means that under maximal progress, a CPDP models hybrid systems with spontaneous and forced transitions.

Two CPDPs can communicate via the synchronization of transitions. If a is a synchronization action, then active a-transitions of the CPDPs should synchronize. This means that if one CPDP has an a-transition enabled (i.e. is inside the guard of some a-transition), and the other CPDP has no a-transition enabled, then this other CPDP blocks the enabled a-transitions of the first CPDP. We call this kind of communication *blocking-interaction.* The other kind of communication that can be expressed is called *broadcasting-interaction.* This happens if an active a-transition of one CPDP triggers a passive $\bar{a}$-transition of the other CPDP. Then the other

CPDP 'observes' that the first CPDP executes an a-transition. Thus, communication/interaction for CPDPs means that CPDPs can get knowledge about the execution of transitions in other CPDPs. Although two (or more) CPDPs cannot have shared continuous variables (as is the case in some other compositional hybrid systems frameworks), it is still possible that information concerning the continuous variables is communicated from one CPDP to the other. For this we need value-passing CPDPs, where active transitions of one CPDP can pass values (which come from the continuous variables) to active transitions in other CPDPs. These passed values can then influence the reset maps of the transitions that received these values and in that way one CPDP can get knowledge about the continuous variables of other CPDPs.

8

Schedulers for CPDPs

If we want the CPDP model to be executable, in other words, if we want that the CPDP can unambiguously be simulated, then all non-determinism within the CPDP specification needs to be resolved. We distinguish three sources of non-determinism in the CPDP model:

- The time of execution of an active transition.
- The choice between the enabled active transitions.
- The choice between passive transitions with the same label.

An active transition can be executed only when the guard of this transition is satisfied. But this does not determine the time of execution, since a transition does not necessarily have to be executed when its guard is satisfied. To resolve this source of non-determinism, we use in this chapter the maximal progress strategy which says that as soon as the guard of one of the active transitions is enabled, then an active transition has to be executed.

It could be the case that multiple active transitions become enabled (i.e., get their guards satisfied) at the same time. To resolve this second source of non-determinism, i.e., to choose which of these enabled active transitions is executed, we define in this chapter a scheduler for CPDPs, which probabilistically chooses which transition is executed. The action of a scheduler for (probabilistic) processes in general depends on the execution fragment before the point that the scheduler has to do its scheduling action (as in [CS02] and [CLSV05]). Our scheduler, which is a special version of this, is such that the action of the scheduler depends only on the state of the process. We will see that restricting the class of allowed schedulers in this way brings the scheduled CPDP in the same semantic domain as the PDP.

If in a composition one CPDP executes an a-transition, then this active transition can trigger a passive $\bar{a}$-transition in another component. If this other component has multiple $\bar{a}$-transitions enabled, a choice has to be made which $\bar{a}$-transition will synchronize on the active a-transition. This choice, which forms the third source of non-determinism, is also solved probabilistically by the scheduler of the component with the passive $\bar{a}$-transitions.

We divide the action of a scheduler in two parts. First, the scheduler determines which active action is executed. We call this the external part of the scheduler.

Second, given which action gets executed, the scheduler determines which transition with that action is executed. (It could be that there are multiple transitions with the same action). We call this the internal part of the scheduler. Also the choice between passive transitions is part of the internal scheduler. We now define the concepts of internal and external schedulers for CPDPs and we define how an internal and an external scheduler together form a scheduler. In the definition we use the concepts *guarded state space*, which consists of all states that lie in the guard of some active transitions and *unguarded state space*, which consists of all states at which no active transitions are enabled.

Definition 8.1. An *internal scheduler* S^i for the CPDP $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ with state space E, assigns to each $\xi \in E$ and each $\sigma \in \Sigma \cup \bar{\Sigma}$ such that $\xi \xrightarrow{\sigma}$, i.e., such that there exists a ξ-enabled σ-transition, a probability measure on the set of ξ-enabled σ-transitions. We write $S(\xi)(\alpha)$ for the probability of transition α at state ξ.

Definition 8.2. An *external scheduler* S^e for CPDP $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ with guarded state space E_g, assigns to each $\xi \in E_g$ a probability measure on Σ. We write $S(\xi)(\sigma)$ for the probability of action σ at state ξ.

Definition 8.3. Let S^i and S^e be internal and external schedulers for the CPDP $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P} = \emptyset, \mathcal{S})$ with guarded state space E_g. The *scheduler* S, formed by S^i and S^e, assigns to each $\xi \in E_g$ a probability measure on the set of ξ-enabled active transitions as follows.

$$S(\xi)(\alpha) := S^e(\xi, lab(\alpha))S^i(\xi, \alpha).$$

Internal and external schedulers can also be defined on the level of NTSs:

Definition 8.4. An *internal scheduler* S^i for NTS $X = (E, \Sigma \cup \bar{\Sigma}, \mathcal{T}, E_O, O)$ assigns to each $\xi \in E$ and each $\sigma \in \Sigma \cup \Sigma$ such that $\xi \xrightarrow{\sigma}$ a probability measure on the set of ξ-enabled σ-transitions.

An *external scheduler* S^e for X assigns to each $\xi \in E_g$ a probability measure on Σ.

Let S^i and S^e be internal and external schedulers for X. If $\bar{\Sigma} = \emptyset$, i.e., there are no passive transitions, then the *scheduler* S, formed by S^i and S^e, assigns to each $\xi \in E_g$ a probability measure on the set of ξ-enabled active transitions as:

$$S(\xi)(\alpha) = S^e(\xi, lab(\alpha))S^i(\xi, \alpha).$$

If S is a scheduler or internal scheduler for X, then we write $S(\xi)(\alpha)$ for the probability of transition α at state ξ. If S^e is an external scheduler for X, then we write $S^e(\sigma)$ for the probability of action σ at state ξ.

8.1 Semantics for scheduled processes

The behavior of an internally scheduled NTS can be captured as a standard NTS as follows.

Definition 8.5. Let $X_N = (E, \Sigma, \mathcal{T}, E_O, O)$ be an NTS with internal scheduler S^i. The corresponding NTS X of (X_N, S^i) is defined as $(E, \Sigma, \hat{\mathcal{T}}, E_O, O)$, where

$$\hat{\mathcal{T}} = \{(\xi, \sigma, m) \mid \xi \xrightarrow{\sigma}, \sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) R_\alpha(\xi) = m\},$$

where '$\xi \xrightarrow{\sigma}$' means 'X_N has a ξ-enabled σ-transition'.

We show how an internal scheduler for a CPDP operates on the level of its semantics, i.e., on the level of the corresponding NTS. We do this by defining how an internal scheduler for a CPDP is transformed into an internal scheduler for the corresponding NTS. Since an internally scheduled NTS behaves as an NTS, we have that the behavior of an internally scheduled CPDP can be captured by an NTS together with a CFSJS, i.e., it uses the same semantic domain as an ordinary CPDP. We also show how an external scheduler for CPDPs is transformed into an external scheduler for the corresponding NTS. The behavior of an externally scheduled NTS cannot be captured as another NTS. However, in the next section we will see that under the assumption of maximal progress, the behavior of a scheduled NTS (i.e., internally and externally scheduled NTS) can be captured as an FTS.

Definition 8.6. Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P}, \mathcal{S})$ be a CPDP with state space E. An internal scheduler S^i for X is transformed into an internal scheduler S^i_N for X_N, the NTS of X, as follows: for all $\alpha = (\xi, \sigma, m)$ of X_N with $\sigma \in \Sigma$

$$S^i_N(\alpha) = \sum_{\tilde{\alpha} \in \mathcal{A}_{\xi \xrightarrow{\sigma} m}} S^i(\xi)(\tilde{\alpha}),$$

where $\mathcal{A}_{\xi \xrightarrow{\sigma} m}$ is the set of σ-transitions α such that $R_\alpha(\xi) = m$, and for all $\alpha = (\xi, \bar{\sigma}, m)$ of X_N with $\bar{\sigma} \in \bar{\Sigma}$

$$S^i_N(\alpha) = \sum_{\tilde{\alpha} \in \mathcal{P}_{\xi \xrightarrow{\sigma} m}} S^i(\xi)(\tilde{\alpha}).$$

An external scheduler S^e for X is transformed into S^e_N for X_N, where $S^e_N(\sigma) := S^e(\sigma)$ for all $\sigma \in \Sigma$, i.e., S^e_N is a copy of S^e.

The NTS semantics of (X, S^i), where X is a CPDP and S^i an internal scheduler for X, is now defined as (X_N, S^i_N), where X_N is the NTS of X and S^i_N is the transformation of S^i.

The NTS semantics of (X, S), where X is a CPDP and $S = (S^i, S^e)$ is a scheduler for X consisting of internal scheduler S^i and external scheduler S^e, is now defined as (X_N, S_N), where X_N is the NTS of X and $S_N = (S^i_N, S^e_N)$, where S^i_N and S^e_N are the transformations of S^i and S^e.

Example 8.7. Consider CPDP X of Figure 8.1 Both l_0 and l_1 have one state variable $x \in \mathbb{R}$, one output variable $y \in \mathbb{R}$ and dynamics $\dot{x} = 1$, $y = x$. Let E and E_O denote the state and output space of X. Initial state is $(l_0, \{x = x_0\})$. All

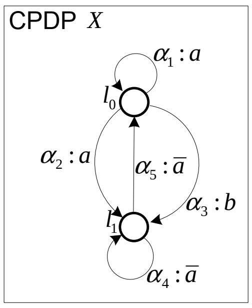

Figure 8.1: Scheduled CPDP

three active transitions, α_1, α_2 and α_3, have guard $x \geq 1$. The internal scheduler S^i is defined for X as: for all $\xi \in (l_0, vs(l_0))$, $S^i(\xi)(l_0 \xrightarrow{a} l_0) = S^i(\xi)(l_0 \xrightarrow{a} l_1) = S^i(\xi)(l_1 \xrightarrow{\bar{a}} l_1) = S^i(\xi)(l_1 \xrightarrow{\bar{a}} l_0) = \frac{1}{2}$. The external scheduler S^e is defined for X as: for all $\xi \in (l_0, vs(l_0))$, $S^e(\xi)(a) = S^e(\xi)(b) = \frac{1}{2}$.

Then, according to Definition 8.6, (X_N, S_N^i) and (X_N, S_N) are defined as: $X_N = (E, \{a, b, \bar{a}\}, \mathcal{T}, E_O, O)$, where

$$\mathcal{T} = \left(\cup_{i\in\{1,2,3\}}\{(\xi,\sigma,m)|\xi \in \mathit{guard}(\alpha_i), m = \mathit{rmap}(\alpha_i)(\xi), \sigma = \mathit{lab}(\alpha_i)\}\right) \cup$$

$$\left(\cup_{i\in\{4,5\}}\{(\xi,\bar{\sigma},m)|m = \mathit{rmap}(\alpha_i)(\xi), \bar{\sigma} = \mathit{lab}(\alpha_i)\}\right),$$

for $l \in \{l_0, l_1\}$ and $r \in \mathbb{R}$, $O((l, \{x = r\})) = \{y = r\}$, for all $\alpha = (\xi, \bar{a}, m) \in \mathcal{T}$, $S_N^i(\alpha) = \frac{1}{2}$, for all $\alpha = (\xi, a, m) \in \mathcal{T}$, $S_N^i(\alpha) = \frac{1}{2}$ and $S_N(\alpha) = \frac{1}{4}$, for all $\alpha = (\xi, b, m) \in \mathcal{T}$, $S_N^i(\alpha) = 1$ and $S_N(\alpha) = \frac{1}{2}$.

8.2 PDP semantics for CPDP

The CPDPs that we consider in this section do not have passive transitions. We call CPDPs that do not have passive transitions *closed*. Passive transitions could be seen as channels through which CPDPs can receive (active) signals from other CPDPs. If we remove passive transitions, then we close these channels. Closing a CPDP can be done by the scope operator $[\cdot]_C$ (see Definition 7.3.1). If we take $C = \bar{\Sigma}$, then the scope operator removes all passive transitions and the result is a closed CPDP. If a composite CPDP is complete, in the sense that it contains all components, then this CPDP should be closed before it is analyzed, because the passive transitions have no function anymore since no more extra components will be added to the composition that could trigger these passive transitions.

In this section we show that the behavior of a scheduled closed CPDP can be captured as a CFSJS and an FTS. This means that a scheduled closed CPDP and a PDP have the same semantic domain. We show that under certain conditions, the semantics of a scheduled closed CPDP is such that a PDP can be found with equivalent semantics. We first have to define what we mean with equivalent semantics. We will see that our notion of equivalent semantics means that the stochastic executions are exactly the same. In other words, if the semantics are equivalent it does not matter, from a stochastic execution point of view, which of the equivalent models we execute.

The key result of this section is that we exactly determine what the conditions for a CPDP are under which the CPDP has equivalent semantics with a PDP. We also give an algorithm that, given a CPDP, determines a PDP with equivalent semantics.

8.2.1 Scheduling and maximal progress

We introduce the notion of *maximal progress* in the context of scheduled NTSs. If a process with NTS $X = (E, \Sigma, \mathcal{T}, E_O, O)$ and scheduler S_X on X is executed under maximal progress, then this means that at every time instant the process should execute a transition from $\mathcal{T}$ as soon as possible. This means that as soon as the process enters an enabled state ξ, a transition is executed. Which transition is executed at state ξ, is determined probabilistically by the probability measure $S_X(\xi)$. We see that with scheduler and under maximal progress, a non-deterministic transition of an NTS is in fact transformed into a forced transition. Therefore we can, under maximal progress, define the FTS of a scheduled NTS.

Definition 8.8. Let $X_N = (E, \Sigma, \mathcal{T}, E_O, O)$ be an NTS with scheduler S. The corresponding FTS X_F of X_N is defined as $(E, \mathcal{T}_F)$, where $(\xi, m) \in \mathcal{T}_F$ if and only if $\mathcal{T}_{\xi\rightarrow} \neq \emptyset$ and

$$m = \sum_{\alpha \in \mathcal{T}_{\xi\rightarrow}} S(\alpha) m_\alpha,$$

where m_α is the reset measure of transition α.

Remark 8.9. In [Her02], maximal progress applies only to (internal) τ-transitions. Because internal transitions can not be delayed by other components, the process may choose to execute them immediately, which will also be done under the assumption of maximal progress. We use maximal progress only for complete CPDPs, i.e., for composite CPDPs that cannot be influenced anymore by other components. For complete CPDPs an active transition with arbitrary label cannot be delayed by other components. Therefore, in our case, maximal progress means that 'an active transition should be executed as soon as possible' and not, as in the IMC case, 'a τ-transition should be executed immediately when a location with one or more τ-transitions is reached'.

Remark 8.10. Because we know how the behavior of a scheduled CPDP can be expressed as a scheduled NTS together with a CFSJS and because we know how the behavior of a scheduled NTS can, under maximal progress, be expressed as an FTS, we now know how the behavior of a scheduled CPDP can, under maximal progress, be captured as a CFSJS together with an FTS. This brings a scheduled CPDP in the same semantic domain as a PDP.

Example 8.11. The FTS of (X_N, S_N) of Example 8.7 equals $(E, \mathcal{T}_F)$, where

$$\mathcal{T}_F = \{(\xi, m) | \xi \in \{(l_0, \{x = r\}) | r \geq 1\},$$
$$m = \frac{1}{4} rmap(\alpha_1)(\xi) + \frac{1}{4} rmap(\alpha_2)(\xi) + \frac{1}{2} rmap(\alpha_3)(\xi)\}.$$

8.2.2 Equivalent FTSs

Two FTSs may be different but indistinguishable from a stochastic execution point of view. For example, consider an FTS X with an enabled state ξ which jumps with probability one to another enabled state ξ' which jumps with probability one to some non-enabled state ξ''. If we have a process with corresponding FTS X and that process reaches state ξ at time t, then the process jumps instantaneously to state ξ'' from where the evolution of the process state continues. The jump goes via state ξ', however in the execution path of the process we do not see state ξ' but the path jumps at time t from ξ to ξ''. Therefore, if we consider an FTS X which is exactly like X except that state ξ jumps to state ξ'' with probability one (i.e., it skips state ξ'), then X and X' behave, from a stochastic execution point of view, exactly the same. Therefore we call FTSs X and X' equivalent. More general, an FTS can jump instantaneously from ξ to ξ''' via states ξ' and ξ'', etc. We now formally define this notion of equivalent FTSs. The definition comes in two steps with the following two definitions.

Definition 8.12. Let $X = (E, \mathcal{T})$ be an FTS and let $\alpha = (\xi, m) \in \mathcal{T}$. We define for all $A \in \mathcal{B}(E)$

$$m_1(A) = m(A), \quad m_i(A) := m_{i-1}(A \cap E_u) + \int_{\hat{\xi} \in E_g} m_{\hat{\xi}}(A) \mathrm{d}m_{i-1}(\hat{\xi}),$$
$$m_\infty(A) := \lim_{n \to \infty} m_n(A),$$

where E_u is the unguarded part of E and m_ξ is the reset measure of the forced transition of X at state ξ. Note that m_∞ is a well defined measure. Now, FTS X_∞ is defined as $(E, \mathcal{T}_\infty)$, where $\mathcal{T}_\infty := \{(\xi, m_\infty) | (\xi, m) \in \mathcal{T}\}$.

Definition 8.13. Two FTSs $X = (E, \mathcal{T}^X)$ and $Y = (E, \mathcal{T}^Y)$ are called equivalent if for all $\xi \in E$ we have

- ξ is an enabled state of X if and only if ξ is an enabled state of Y,
- if m^X and m^Y are such that $(\xi, m^X) \in \mathcal{T}^X$ and $(\xi, m^Y) \in \mathcal{T}^Y$, then for all $A \in \mathcal{B}(E_u)$, $m^X_\infty(A) = m^Y_\infty(A)$, where E_u is the set of all enabled states of E.

For situations where we want to compare two FTSs X and Y with different state spaces E_X and E_Y but where we identify each state of E_X with one state of E_Y and vice versa, we define the notion of *isomorphic equivalence*.

Definition 8.14. Let $X = (E_X, \mathcal{T}^X)$ and $Y = (E_Y, \mathcal{T}^Y)$ be two FTSs and let $\iota : E_X \to E_Y$ be a Borel isomorphism. X and Y are called isomorphic equivalent with respect to ι if for all $\xi \in E_X$ we have

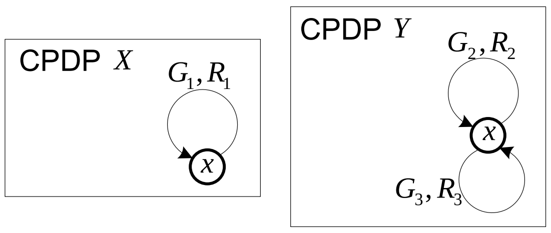

Figure 8.2: CPDPs with equivalent FTSs

- ξ is an enabled state of X if and only if $\iota(\xi)$ is an enabled state of Y,
- if m^X and m^Y are such that $(\xi, m^X) \in \mathcal{T}^X$ and $(\iota(\xi), m^Y) \in \mathcal{T}^Y$, then for all $A \in \mathcal{B}(E_{X,u})$, $m^X_\infty(A) = m^Y_\infty(\iota(A))$, where $E_{X,u}$ is the set of all enabled states of E_X.

Example 8.15. Consider CPDP X and Y of Figure 8.2. X and Y share the same state space E. Guard G_1 is satisfies when $x \in [-\infty, -1] \cup [1, \infty]$, G_2 is satisfied when $x \in [-\infty, -1]$ and G_3 is satisfied when $x \in [1, \infty]$. Reset map R_1 equals $U[-1, 1]$, i.e., the uniform distribution on $[-1, 1]$, for all states in G_1. R_2 equals $U[1, 2]$ for all states in G_2 and R_3 equals $U[-1, 1]$ for all states in G_3.

The sets of enabled states of X_F and Y_F, i.e., the FTSs of X and Y, are both equal to $G_1 = G_2 \cup G_3$. For $\xi \in G_3$, (ξ, m), with $m = R_1(\xi)$, is a transition of both X_F and Y_F, and for both X and Y we have $m_\infty = m$. For $\xi \in G_2$, (ξ, m^X), with $m^X = R_1(\xi)$, is a transition of X_F and (ξ, m^Y), with $m^Y = R_2(\xi)$, is a transition of Y_F. We have $m^X_\infty = m^X$ and $m^Y_\infty = m^Y_1 \neq m^Y$. Now, m^Y_1 equals m^X because m^Y_1 jumps with probability one in G_3 followed directly by jumping with distribution $U[-1, 1]$ which equals m^X, which jumps in one step with distribution $U[-1, 1]$. Thus, we found that according to Definition 8.13, X_F and Y_F are equivalent.

8.2.3 Equivalent TMSs

This FTS (isomorphic) equivalence notion can also be defined on the level of TMSs.

Definition 8.16. Let $X = (E, \xi_0, \phi, (T, Q))$ be a TMS. We split E into E_g and E_u, where $E_g = \{\xi \in E | \mathrm{P}(T(\xi) = 0) = 1\}$ and $E_u = E \backslash E_g$. We define for all $A \in \mathcal{B}(E)$:

$$Q_1(\xi)(A) := Q(A), \quad Q_n(\xi)(A) := Q_{n_1}(A \cap E_u) + \int_{\hat{\xi} \in E_g} Q(\hat{\xi})(A) \mathrm{d}Q_{n-1}(\xi)(\hat{\xi}),$$

$$Q_\infty(\xi)(A) := \lim_{n \to \infty} Q_n(\xi)(A).$$

TMSs $(E, \xi_0, \phi, (T, Q))$ and $(E, \xi_0, \phi, (T, \hat{Q}))$ are called *equivalent* if for all $\xi \in E$ we have: for all $A \in \mathcal{B}(E_u)$, $Q_\infty(\xi)(A) = \hat{Q}_\infty(\xi)(A)$.

Definition 8.17. Let $X = (E_X, \xi_{X,0}, \phi_X, (T_X, Q_X))$ and $Y = (E_Y, \xi_{Y,0}, \phi_Y, (T_Y, Q_Y))$ be two TMSs and let $\iota : E_X \to E_Y$ be a Borel isomorphism. X and Y are called *isomorphic equivalent* with respect to ι if for all $\xi \in E_X$ we have:

- $\xi_{Y,0} = \iota(\xi_{X,0})$,
- for all $t > 0$ $\iota(\phi_X(t, \xi)) = \phi_Y(t, \iota(\xi))$,
- for all $t > 0$ $\mathrm{P}(T_X(\xi) > t) = \mathrm{P}(T_Y(\iota(\xi)) > t)$ and
- for all $A \in \mathcal{B}(E_{X,u})$, $Q_{X,\infty}(\xi)(A) = Q_{Y,\infty}(\iota(\xi))(\iota(A))$.

Remark 8.18. If $X = (E, \xi_0, \phi, (T, Q))$ is a TMS such that $T_\infty(E_u) = 1$ for all $\xi \in E$, then it can be easily seen that instead of generating stochastic executions for X, we could as well generate stochastic executions for $X_\infty = (E, \xi_0, \phi, (T, Q_\infty))$. For X_∞ we know that with any transition we reach E_g in one step, where for X it may take several steps/transitions before we reach E_g.

The following lemma shows how the equivalence notions for FTS and TMS are related.

Lemma 8.19. *Let $X_C = (E, \xi_0, \phi, \lambda, Q)$ be a CFSJS. Let $X_F^1 = (E, \mathcal{T}^1)$ and $X_F^2 = (E, \mathcal{T}^2)$ be equivalent FTSs. Then, the TMSs of (X_C, X_F^1) and (X_C, X_F^2) are equivalent.*

Proof. It is clear that the state space, initial state and flow map are the same for the TMS of (X_C, X_F^1) and the TMS of (X_C, X_F^2). Because X_F^1 and X_F^2 are equivalent, they have the same guarded state space, which means in TMS terms that they have the same set of states ξ for which we have $\mathrm{P}(T(\xi) = 0) = 1\}$. This, combined with the fact that the TMSs have the same underlying CFSJS, gives that the jump-time stochastic variable of the two TMSs is the same. We only have to check that the transition measures Q^1 and Q^2 of the TMSs of (X_C, X_F^1) and (X_C, X_F^2) are equivalent. It can be easily seen that for a state ξ, $Q^1_\infty(\xi) = m^1_\infty$ and $Q^2_\infty(\xi) = m^2_\infty$ where m^1 and m^2 are the reset measures of the FTSs at state ξ. Since the FTSs are equivalent we get $Q^1_\infty(\xi)(A) = Q^2_\infty(\xi)(A)$ for all Borel sets of the unguarded state space, from which the equivalence of the TMSs follows. □

8.2.4 Transforming CPDP-transitions

An active transition of a scheduled CPDP may, with some probability, jump into the guard area of another active transition. Under maximal progress this means that, with some probability, these two active transitions are executed one after the other, without any time consumed between the two transitions. This means that then a chain of transitions is instantaneously executed. As an FTS transition jumping from ξ to ξ'' via ξ' could be replaced by a transition jumping from ξ to ξ'' directly, we now show how a chain of active CPDP transitions that are executed at the same time one after the other, can be replaced by a single active CPDP transition, which skips all the intermediate states of the intermediate transitions of the chain. We show by which transition we should replace these chains of transitions and we also show that by doing this, the FTS of the new CPDP (i.e., the CPDP where the chain of transitions is replaced by the single transition) is equivalent to the FTS of the original CPDP.

The first step of this transformation process is splitting all active transitions in *stable* and *unstable* transitions, where stable and unstable transitions are defined as follows.

Definition 8.20. Let α be an active transition on a state space $E = E_g \cup E_u$. If $rmap(\alpha)(E_u) = 1$, then we call α a *stable* transition. If $rmap(\alpha)(E_g) = 1$, then we call α an *unstable* transition.

Note that an active transition is neither stable nor unstable when it can jump into both the guarded and unguarded area of the state space.

Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P} = \emptyset, \mathcal{S})$ be a CPDP with state space E and scheduler S. Let E_g and E_u be the guarded and unguarded part of E respectively. Take any $\alpha \in \mathcal{A}$. We now split up α in a stable transition α_s and an unstable transition α_u as follows.

We define G_{α_s} as the set of all $\xi \in guard(\alpha)$ such that $rmap(\alpha)(E_u) \neq 0$. Then for all $\xi \in G_{\alpha_s}$ we define R_{α_s} and $\hat{S}$, which is a scheduler for all new stable and unstable transitions, as

$$R_{\alpha_s}(\xi)(A) := \frac{rmap(\alpha)(\xi)(A \cap E_u)}{rmap(\alpha)(\xi)(E_u)},$$

$$\hat{S}(\xi)(\alpha_s) := S(\xi)(\alpha) rmap(\alpha)(\xi)(E_u).$$

We define G_{α_u} as the set of all $\xi \in guard(\alpha)$ such that $rmap(\alpha)(E_g) \neq 0$. Then for all $\xi \in G_{\alpha_u}$ we define R_{α_u} and $\hat{S}$ as

$$R_{\alpha_u}(\xi)(A) := \frac{rmap(\alpha)(\xi)(A \cap E_g)}{rmap(\alpha)(\xi)(E_g)},$$

$$\hat{S}(\xi)(\alpha_u) := S(\xi)(\alpha) rmap(\alpha)(\xi)(E_g).$$

The following lemma says that replacing all active transitions by their stable and unstable parts, results in an equivalent FTS.

Lemma 8.21. *Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P} = \emptyset, \mathcal{S})$ be a CPDP with scheduler S. Let $\hat{X}$ be the CPDP $(L, V, \nu, W, \omega, F, G, \Sigma, \hat{\mathcal{A}}, \mathcal{P} = \emptyset, \mathcal{S})$ with scheduler $\hat{S}$, where*

$$\hat{\mathcal{A}} = \{\alpha_s | \alpha \in \mathcal{A}\} \cup \{\alpha_u | \alpha \in \mathcal{A}\},$$

and α_s, α_u and $\hat{S}$ are defined as above. Then, the FTSs of X and $\hat{X}$ are equivalent.

Proof. E_g is clearly the same for both X and $\tilde{X}$. Take $\xi \in E_g$ arbitrary. Let m denote the FTS reset measure at state ξ for X and let $\tilde{m}$ denote the FTS reset measure at state ξ for $\tilde{X}$. Let A be a Borel set of E. We show that $m(A) = \tilde{m}(A)$.

$$m(A) = \sum_{\alpha \in \mathcal{A}_{\xi\rightarrow}} S(\xi)(\alpha) rmap(\alpha)(\xi)(A) =$$

$$\sum_{\alpha\in\mathcal{A}_{\xi\rightarrow}} S(\xi)(\alpha)\left(rmap(\alpha)(\xi)(A\cap E_g)+rmap(\alpha)(\xi)(A\cap E_u)\right)=$$

$$\sum_{\alpha\in\mathcal{A}_{\xi\rightarrow}}(S(\xi)(\alpha)rmap(\alpha)(\xi)(E_g)\frac{rmap(\alpha)(\xi)(A\cap E_g)}{rmap(\alpha)(\xi)(E_g)}+$$

$$S(\xi)(\alpha)rmap(\alpha)(\xi)(E_u)\frac{rmap(\alpha)(\xi)(A\cap E_u)}{rmap(\alpha)(\xi)(E_u)})=$$

$$\sum_{\alpha\in\mathcal{A}^1_{s,\xi\rightarrow}\cup\mathcal{A}^1_{u,\xi\rightarrow}}\hat{S}(\xi)(\alpha)rmap(\alpha)(\xi)(A)=\hat{m}(A).$$

□

We now define how a chain of two transitions α and β, where β is to be executed directly after α, can be expressed as a single transition, which we denote by $\beta\circ\alpha$.

Definition 8.22. Let α and β be active transitions of some CPDP with state space E and scheduler S, such that $tloc(\alpha)=oloc(\beta)$. Let G_α,G_β and R_α, R_β denote the guards and reset maps of α and β. Let for $A\in\mathcal{B}(E)$, $P_\xi(A\circ\beta\circ\alpha)$ denote the probability that (under maximal progress) at state ξ the transition α is executed followed directly by the transition β jumping into the set A. Then,

$$P_\xi(A\circ\beta\circ\alpha)=S(\xi)(\alpha)\int_{\hat{\xi}\in G_\beta}S(\hat{\xi})(\beta)R_\beta(\xi)(A)\mathrm{d}R_\alpha(\xi)(\hat{\xi}).$$

We define the new scheduler $\tilde{S}$ and the composed active transition $\beta\circ\alpha=(oloc(\alpha),$ $\tau, tloc(\beta), G_{\beta\circ\alpha}, R_{\beta\circ\alpha})$ as

$$G_{\beta\circ\alpha}:=\{\xi\in guard(\alpha)|R_\alpha(\xi)(G_\beta)\neq\emptyset\},$$

$$\tilde{S}(\xi)(\beta\circ\alpha):=P_\xi(E\circ\beta\circ\alpha),$$

and for all $A\in\mathcal{B}(E)$,

$$R_{\beta\circ\alpha}(\xi)(A):=\frac{P_\xi(A\circ\beta\circ\alpha)}{\tilde{S}(\xi)(\beta\circ\alpha)}.$$

Let α, β and γ be transitions from $\mathcal{A}$. We call α, β and γ transitions of multiplicity one. We call $\beta\circ\alpha$ a transition with multiplicity two. We call $\gamma\circ(\beta\circ\alpha)$ a transition of multiplicity three, etc.

We now define

$$\mathcal{A}^1_s:=\{\alpha_s|\alpha\in\mathcal{A}, guard(\alpha_s)\neq\emptyset\},$$

and

$$\mathcal{A}^1_u:=\{\alpha_u|\alpha\in\mathcal{A}, guard(\alpha_u)\neq\emptyset\}.$$

$\mathcal{A}^1_s$ forms the set of all stable parts of all transitions of $\mathcal{A}$ that are not unstable. $\mathcal{A}^1_u$ forms the set of all unstable parts of all transitions of $\mathcal{A}$ that are not stable. We now inductively define $\mathcal{A}_s^n$ and $\mathcal{A}_u^n$ for all natural $n > 1$. Suppose $\mathcal{A}_s^{n-1}$ and $\mathcal{A}_u^{n-1}$ are known. Then

$$\mathcal{A}_s^n := \{\beta \circ \alpha | \alpha \in \mathcal{A}_u^{n-1}, \beta \in \mathcal{A}_s^1, R_\alpha(\xi)(G_\beta) \neq \emptyset\}$$

and

$$\mathcal{A}_u^n := \{\beta \circ \alpha | \alpha \in \mathcal{A}_u^{n-1}, \beta \in \mathcal{A}_u^1, R_\alpha(\xi)(G_\beta) \neq \emptyset\}.$$

$\mathcal{A}_s^n$ forms a set of stable active transitions of multiplicity n and $\mathcal{A}_u^n$ forms a set of unstable active transitions of multiplicity n. For $i \in \mathbb{N}$, $\tilde{S}$ is defined for transitions from $\mathcal{A}_s^i$ and $\mathcal{A}_u^i$ as in Definition 8.22.

The following lemma shows how for any $n \in \mathbb{N}$ we can change the CPDP, without changing its semantics up to equivalence, such that there are no unstable transitions anymore of multiplicity smaller than n

Lemma 8.23. *Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P} = \emptyset, \mathcal{S})$ be a CPDP with scheduler S. Let X^n be the CPDP $(L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}^n, \mathcal{P} = \emptyset, \mathcal{S})$, where*

$$\mathcal{A}^n := \mathcal{A}_s^1 \cup \mathcal{A}_s^2 \cup \cdots \cup \mathcal{A}_s^n \cup \mathcal{A}_u^n,$$

where $\mathcal{A}_s^i$ and $\mathcal{A}_u^i$ are derived from $\mathcal{A}$ as described above. Then, $\tilde{S}$ is a well-defined scheduler for X^n. Furthermore, for all $n \in \mathbb{N}$, the FTS of X with scheduler S is equivalent to the FTS of X^n with scheduler $\tilde{S}$.

Proof. We first prove that $\tilde{S}$ is a well-defined scheduler for X^n. We do this inductively. For $n = 1$ we know from lemma 8.21 that $\tilde{S}$ is a scheduler. Assume that $\tilde{S}$ is a scheduler for some $n-1$. We show that then $\tilde{S}$ is also a scheduler for n.

In order to show that $\tilde{S}$ is a scheduler for X^n, we have to show for all $\xi \in E_g$ that $\sum_{\hat{\alpha} \in \mathcal{A}^n} \tilde{S}(\xi)(\alpha) = 1$. From the fact that $\sum_{\hat{\alpha} \in \mathcal{A}_s^1 \cup \mathcal{A}_u^1} \tilde{S}(\xi)(\alpha) = 1$ we can deduce that for any $\alpha \in \mathcal{A}_u^{n-1}$

$$\sum_{\hat{\alpha} \in \mathcal{A}_s^1 \cup \mathcal{A}_u^1} \mathrm{P}(E \circ \tilde{\alpha} \circ \alpha)(\xi) = \tilde{S}(\xi)(\alpha).$$

Then we can derive

$$\sum_{\alpha \in \mathcal{A}^n} \tilde{S}(\xi)(\alpha) = \sum_{\alpha \in \cup_{i=1}^{n-1} \mathcal{A}_s^i} \hat{S}(\xi)(\alpha) + \sum_{\alpha \in \mathcal{A}_u^{n-1}} \sum_{\hat{\alpha} \in \mathcal{A}_s^1 \cup \mathcal{A}_u^1} \mathrm{P}(E \circ \tilde{\alpha} \circ \alpha)(\xi) =$$

$$\sum_{\alpha \in \cup_{i=1}^{n-1} \mathcal{A}_s^i} \tilde{S}(\xi)(\alpha) + \sum_{\alpha \in \mathcal{A}_u^{n-1}} \tilde{S}(\alpha)(\xi) = \sum_{\alpha \in \mathcal{A}^{n-1}} \tilde{S}(\alpha)(\xi) = 1.$$

Now we prove the equivalence of the FTSs of X and X^n. Take $\xi \in E_g$ arbitrary. Let m be the reset measure of the FTS of X at state ξ and let for all $n \in \mathbb{N}$ m^n be the reset measure of the FTS of X^n at state ξ. From Lemma 8.21 we know that m_1 is equal to m^1. Assume that for some $n \in \mathbb{N}$ we have that m_{n-1} is equal to m^{n-1}. We show that then m_n is equal to m^n. By definition (see Section 8.2.2) we have $m_n(A) = p_1(A) + p_2(A)$, where

$$p_1 = m_{n-1}(A \cap E_u), \quad p_2 = \int_{\hat{\xi} \in E_g} m_{\hat{\xi}}(A) \mathrm{d}m_{n-1}(\hat{\xi}).$$

Now it can be seen that p_1 is exactly the same as the contribution of the transitions from $\mathcal{A}_s^i$, $i = 1 \cdots n-1$, to the reset measure m^n and p_2 is exactly the same as the contribution of the transitions from $\mathcal{A}_s^n$ and $\mathcal{A}_u^n$ to the reset measure m^n. Thus we find that $m_n = m^n$ and we inductively proved that $m_n = m^n$ for all $n \in \mathbb{N}$. It can easily be seen that we then also have $m_{2n} = m_2^n$, $m_{3n} = m_3^n$,... and $m_\infty = m_\infty^n$, from which we conclude that the FTSs are equivalent. □

Now we can state the main theorem of this section, which says when and how a scheduled CPDP can be converted to a PDP without changing the semantics up to equivalence.

Theorem 8.24. *Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P} = \emptyset, \mathcal{S})$ be a CPDP with initial state ξ_0, state space E, scheduler S and CFSJS $(E, \xi_0, \phi, \lambda, Q)$. Let X^n be defined as in Lemma 8.23. Define $R^n_{tot,s}$, the total stable reset map as follows. For all $\xi \in E$,*

$$R^n_{tot,s}(\xi) := \sum_{\alpha \in \mathcal{A}_s^{\Sigma n}} \tilde{S}(\xi)(\alpha) R_\alpha,$$

where $\mathcal{A}_s^{\Sigma n} := \cup_{i=1\cdots n} \mathcal{A}_s^i$. There exists a PDP X_P such that the FTS of X_P is equivalent to the FTS of X if

1. *for all $\xi \in E_g$,*
$$Q_s := \lim_{n\to\infty} R^n_{tot,s}(\xi)(E) = 1,$$
with E_g the guarded part of E,
2. *the vector fields of the locations are locally Lipschitz,*
3. *λ and Q satisfy the conditions of items 4 and 5 of Definition 6.1 respectively,*
4. *there are no explosions and the CPDP is non-zeno. (The notions* explosions *and* non-zeno *are defined for CPDPs as they are defined for PDPs in Definition 6.1).*

If these conditions are satisfied, then the FTS of PDP $X_P = (L_P, Inv_P, F_P, \lambda_P, Q_P)$, where (L_P, Inv) is Borel isomorphic to E_u with isomorphism ι and

- *$F_P(\iota(\xi)) = \pi_{cont}(\iota(F(\xi)))$, where π_{cont} maps a PDP state (l, r), with $r \in Inv_P(l)$, to its continuous component r,*
- *$\lambda_P(\iota(\xi)) = \lambda(\xi)$,*
- *for $\xi \in \partial E_u$ and $A \in \mathcal{B}(E_u)$, $Q_P(\iota(\xi))(\iota(A)) = Q_s(\xi)(A)$, where $\iota(A) := \{\iota(\xi) | \xi \in A\}$, and for $\xi \in E_u$*
$$Q_P(\iota(\xi))(\iota(A)) = Q(\xi)(A \cap E_u) + \int_{\hat{\xi} \in E_g} Q_s(\xi)(A) \mathrm{d}Q(\xi)(\hat{\xi}), \qquad (8.1)$$

is isomorphic equivalent with respect to ι to the FTS of X with scheduler S. Furthermore, the TMS of X is isomorphic equivalent with respect to ι to the TMS of X_P.

Proof. $Q_s(E) = 1$ means that the reset measure m_∞ of the FTS jumps to the unguarded state space E_u with probability one. Since a PDP does not have guarded states, it is clear that for any reachable state $\xi \in E_g$ we should have $Q_s(E) = 1$, because $Q_s(E) < 1$ means that there is a probability greater than zero that we do not return in finitely many jumps to the unguarded state space. With $Q_s(E) = 1$, we are sure that a spontaneous jump of X, via transition measure Q will end up in the unguarded state space within two steps and $Q_2(\xi)(A)$, which then equals $Q_\infty(\xi)(A)$, is equal to the right hand side of Equation 8.1. Then, the TMS of X has this transition measure $Q_\infty(\xi)$ for states $\xi \in E_u$ and has transition measure $Q_s(\xi)$ for (boundary) states $\xi \in \partial E_u$. From Conditions 2,3 and 4 follows that Conditions 3,4 and 5 of Definition 6.1 are satisfied. Proving isomorphic equivalence with the FTS and TMS of the PDP is trivial now. □

Remark 8.25. The condition on Q_s in Theorem 8.24 is not an if-and-only-if condition for equivalence of CPDP and PDP. It can be easily seen that the if-and-only-if condition should be that the condition does not hold for all $\xi \in E_g$, but for all $\xi \in E_g$ such that ξ is reachable from the initial state with probability greater than zero. However, this condition is hard to check in general.

Corollary 8.26. *If for some $n \in N$ we have that, for CPDP X, $\mathcal{A}_u^n = \emptyset$, then, for all $\xi \in \partial E_u$, $R^n_{tot,s}(\xi)(E) = 1$ and the the FTS of the PDP defined in Theorem 8.24, now with $Q_s := R^n_{tot,s}(\xi)$, is isomorphic equivalent to the FTS of X.*

Corollary 8.26 provides an algorithm to convert a CPDP to a PDP by inductively determining $\mathcal{A}_u^1$, $\mathcal{A}_u^2$, $\mathcal{A}_u^3$ until for some n we find $\mathcal{A}_u^n = \emptyset$. It is not guaranteed that the algorithm finishes in a finite number of steps, but if it does, then we can directly construct the PDP whose TMS is equivalent to the TMS of X.

Example 8.27. In this example we show how composition can lead to unstable transitions and how these unstable transitions can be transformed into stable PDP transitions. Consider CPDPs X, M_1 and M_2 of Figure 8.3. M_1 and M_2 are memory units and the values of variables m_1 and m_2 form the contents of the memory units. X is a part of some process, whose evolution depends on the contents of the memory units. X shows the part of the process where the values of m_1 and m_2 are downloaded and stored into variables $\tilde{m}_1$ and $\tilde{m}_2$. At location l_2, the continuous dynamics of X is determined by vector field f_1, until the guard of the *getmem*1 transition, which is not pictured in Figure 8.3, is satisfied. Once the values are downloaded, the continuous dynamics of X depend on these values via vector field f_2.

The downloading process is modelled via value passing in the composition of the three CPDPs. We compose as follows: $(X|^P_{A_1}|M_1)|^P_{A_2}|M_2$, where $A_1 = \{getmem1\}$, $A_2 = \{getmem2\}$ and P is not relevant. In the *getmem*1 transition of X, which synchronizes with the transition of M_1, the value of m_1 is downloaded and stored into $\tilde{m}_1$. Storing is done via the reset map of the value passing *getmem*1 transition of X which, if we denote it by R, is defined as $R(\{x = r_1\}, \{m_1 = r_2\})(\{\{x = r_1, \tilde{m}_1 = r_2\}\}) = 1$. Note that at location l_3 also the value of x before the transition is stored. The guard of the transitions of M_1 and M_2 equal the whole state space

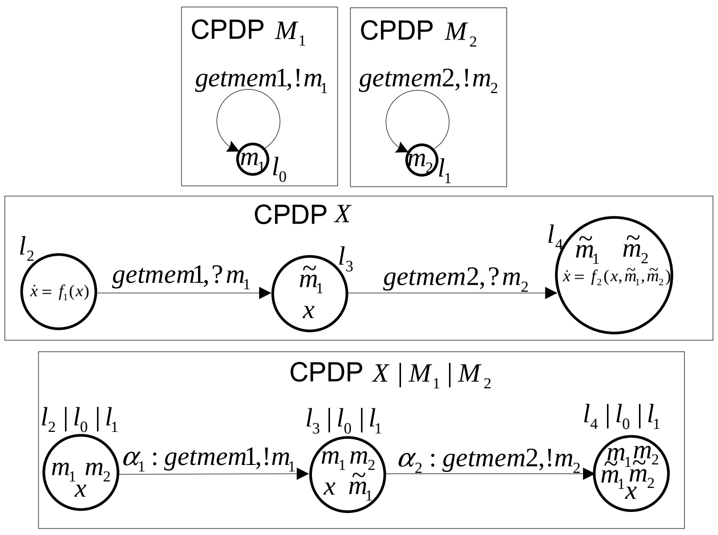

Figure 8.3: CPDP transformed into PDP

of m_1 and m_2. This expresses that X can download m_1 or m_2 at all times. The guard of the *getmem*2 transition of X equals the whole state space. Then, after the synchronized *getmem*1 transition, we directly get a synchronized *getmem*2 transition which expresses the download of m_2 from memory M_2. The reset map R then stores m_2 into $\tilde{m}_2$ and uses the identity reset measure for $\tilde{m}_1$ and x, i.e., $R(\{x = r_1, \tilde{m}_1 = r_2\}, \{\tilde{m}_2 = r_3\})(\{\{x = r_1, \tilde{m}_1 = r_2\}, \{\tilde{m}_2 = r_3\}\}) = 1$. The composite CPDP equals $X|M_1|M_2$ of Figure 8.3.

Now we will determine for $n > 1$ $\mathcal{A}_s^n$ and $\mathcal{A}_u^n$ for CPDP $X|M_1|M_2$. The unguarded state space of $X|M_1|M_2$ is formed by the unguarded state space of location $l_2|l_0|l_1$ plus the whole state space of location $l_4|l_0|l_1$. Transition α_1 is an unstable transition and transition α_2 is a stable transition. This means that $\mathcal{A}_s^1 = \{\alpha_2\}$ and $\mathcal{A}_u^1 = \{\alpha_1\}$. The guard of transition $\alpha_2 \circ \alpha_1$, from $l_2|l_0|l_1$ to $l_4|l_0|l_1$, is satisfied when x satisfies the guard of the *getmem*1 transition of X. The reset map of $\alpha_2 \circ \alpha_1$ is the identity reset map for x, m_1 and m_2 and stores m_1 and m_2 into $\tilde{m}_1$ and $\tilde{m}_2$. $\alpha_2 \circ \alpha_1$ is a stable transition and we get $\mathcal{A}_s^2 = \{\alpha_2 \circ \alpha_1\}$ and $\mathcal{A}_u^2 = \emptyset$. Now, according to Theorem 8.24, the corresponding PDP of $X|M_1|M_2$ is the one which has as state space the unguarded state space of $X|M_1|M_2$, which has the same continuous dynamics as $X|M_1|M_2$ and which has transition measure equal to the reset map of $\alpha_2 \circ \alpha_1$. Note that α_2 is not relevant for determining the corresponding PDP because location $l_3|l_0|l_1$ will never be reached since $\alpha_2 \circ \alpha_1$ will always jump directly to location $l_4|l_0|l_1$ and skips $l_3|l_0|l_1$. We could say that the relevance of α_2 is already incorporated in transition $\alpha_2 \circ \alpha_1$.

Example 8.28. We revisit the repair shop example. The repair shop was modelled

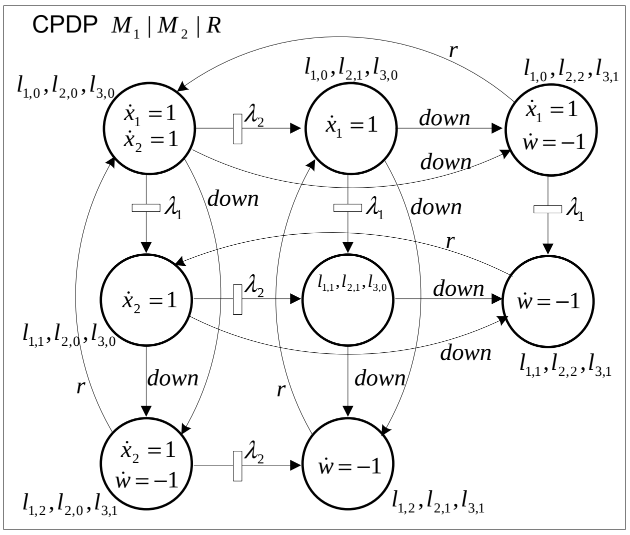

Figure 8.4: Composite CPDP of repair shop

as a PDP in Section 6.4 and was modelled as a composition of CPDPs in Example 7.12. The composed CPDP $[(M_1|^{\emptyset}_{\emptyset}|M_2)|^{\emptyset}_{down}|R]_{\bar{\Sigma}}$, i.e., the closed version of CPDP $(M_1|^{\emptyset}_{\emptyset}|M_2)|^{\emptyset}_{down}|R$, is pictured in Figure 8.4 as $M_1|M_2|R$, which is shorthand notation. We have closed the CPDP by the scope operator because it is complete, i.e., no more components will be added to the composition. $M_1|M_2|R$ has, because it is a closed CPDP now, no passive transitions as can be seen in Figure 8.4. The name of the middle location, which is an empty location, is written inside the location because of lack of space in the picture. Also, the reset maps and the guards are not pictured here. We have only drawn the eight locations of $M_1|M_2|R$ which are reachable from the initial joint location $l_{1,0}, l_{2,0}, l_{3,0}$. We will transform CPDP $M_1|M_2|R$ into a PDP.

We first explain why it is correct to execute CPDP $M_1|M_2|R$ under maximal progress. Any *down*-transition in $M_1|M_2|R$ reflects the synchronization of a *down*-transition of either M_1 or M_2 with a *down*-transition of R. If a *down*-transition in $M_1|M_2|R$ is enabled, it means that one machine needs to be repaired and that the repair shop is able to receive a machine to work on. In that case the machine is 'brought to the repair shop' immediately, which is indeed expressed by maximal progress because it takes care that as soon as the *down*-transition is enabled, it is immediately executed. Any r-transition in $M_1|M_2|R$ reflects the synchronization of a $\bar{r}$-transition of either M_1 or M_2 with an r-transition of R. If such an r-transition gets enabled, it means that the machine is repaired. Therefore the transition should be executed immediately such that the repair shop can work on another machine that might be waiting for repair work. This is again expressed by maximal progress.

The only state of $M_1|M_2|R$ that needs a scheduler is the state $\{x_1 = s_1, x_2 = s_2\}$

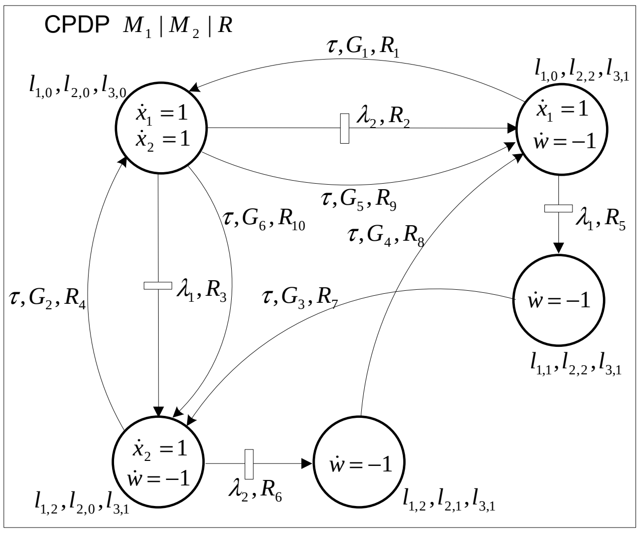

Figure 8.5: Transformed composite CPDP of repair shop

where two *down*-transitions are enabled. For this example we assume that this state is never reached and then CPDP $M_1|M_2|R$ does not need a scheduler to determine its FTS. Note that s_1 and s_2 may not be equal under this assumption. In Example 8.37 we drop this assumption and we determine a scheduler for $M_1|M_2|R$.

All transitions in $M_1|M_2|R$ are stable, therefore we could, according to Theorem 8.24, directly determine the corresponding PDP which then also has these eight locations. However, we first do another reduction step on the CPDP before we determine the corresponding PDP: if the λ_1 transition from $l_{1,0}, l_{2,0}, l_{3,0}$ to $l_{1,1}, l_{2,0}, l_{3,0}$ is executed at some time t, then from $l_{1,1}, l_{2,0}, l_{3,0}$ the *down*-transition to $l_{1,3}, l_{2,0}, l_{3,1}$ is executed immediately because of maximal progress. Therefore, this λ_1 transition can be replaced by the a λ_1 transition from $l_{1,0}, l_{2,0}, l_{3,0}$ directly to $l_{1,2}, l_{2,0}, l_{3,1}$, with as reset map the reset map of the *down*-transition from $l_{1,1}, l_{2,0}, l_{3,0}$ to $l_{1,2}, l_{2,0}, l_{3,1}$, without changing the CFSJS/FTS semantics up to equivalence. Similarly, we can replace the λ_2-transition from $l_{1,0}, l_{2,0}, l_{3,0}$ to $l_{1,0}, l_{2,1}, l_{3,0}$ with a λ_2-transition from $l_{1,0}, l_{2,0}, l_{3,0}$ directly to $l_{1,0}, l_{2,3}, l_{3,1}$. After these two transformations, the locations $l_{1,1}, l_{2,0}, l_{3,0}$, $l_{1,0}, l_{2,1}, l_{3,0}$ and $l_{1,1}, l_{2,1}, l_{3,0}$ have become unreachable and can therefore be deleted. The result is now pictured in Figure 8.5, where all active actions are replaced by τ-actions, because the action names play no role in determining the corresponding PDP. Now we have five locations left. These locations correspond exactly with the five locations of the PDP of Section 6.4 and it can be checked that the corresponding PDP of the CPDP of Figure 8.5, is exactly the PDP of Section 6.4.

The following example shows that if the algorithm does not terminate on a CPDP, there could still be a PDP with equivalent semantics.

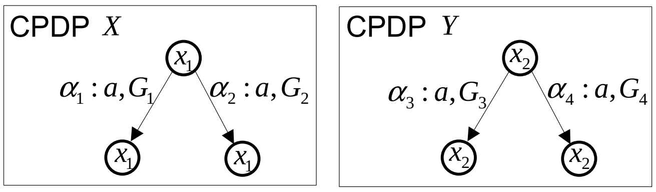

Figure 8.6: Scheduling composition of CPDPs

Example 8.29. Let CPDP X have one location, l_1. The state-space of l_1 is $[0,1]$, the continuous dynamics of l_1 is the clock dynamics $\dot{x} = 1$. From l_1 to l_1 there is one active transition with guard G and reset map R. $G = [\frac{1}{2}, 1]$. For $x \in G$, $R(\{0\}, x) = \frac{1}{2}$ and $R(A, x) = |A \cap [\frac{1}{2}, 1]|$ for $A \in \mathcal{B}([0,1]\backslash\{0\})$. This means that from an x in G, the reset map jumps to 0 with probability $\frac{1}{2}$ and jumps uniformly into $[\frac{1}{2}, 1]$ with probability $\frac{1}{2}$. It can easily be seen that for X we have that $T_u^n \neq \emptyset$ for all $n \in \mathbb{N}$. This means that the algorithm explained above does not terminate for this example. Still, according to Theorem 8.24, X expresses a PDP behavior, because for $x \in G$, $R([0,1], x) = \lim_{n\to\infty} R^n_{tot,s}([0,1], x) = \frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2} + \frac{1}{2}\cdot\frac{1}{2}\cdot\frac{1}{2} + \cdots = 1$.

8.3 Schedulers and composition

We could question whether it is possible to compose scheduled CPDPs (X, S_X) and (Y, S_Y) such that the result is the scheduled CPDP $(X|^P_A|Y, S)$, where S is some scheduler for CPDP $X|^P_A|Y$ that is derived from the schedulers S_X and S_Y. In general, it is not possible to do this in a 'natural' way. The main problem lies in the external scheduling part. In a composition, external scheduling for a component X is already partly done by the other components in the composition. For example, if X is composed with Y via $|^P_A|$, where $A = \{a\}$ and Y does not have a-transitions, then via this composition the action a of X is scheduled such that the probability of an a action is zero at any state of X. Now we can see that if we want to compose scheduled CPDPs, then the external parts of the schedulers might conflict with the scheduling done via the composition itself. In the above example: any external scheduler for X that assigns nonzero probability to a is conflicting with the composition with Y via $|^P_A|$. However, internal scheduling is not done by the composition itself. Therefore, internal schedulers may be composed to form an internal scheduler for the composed CPDP. Still, at certain states the internal schedulers of the components do not contain enough information to assess all the internal scheduling values for the transitions of the composed CPDP. If in such a state ξ more information is needed to schedule the σ-transitions, then we call the pair (ξ, σ) a *conflict pair*. The set of conflict pairs is defined as follows.

Definition 8.30. Let $X = (L_X, V_X, \nu_X, W_X, \omega_X, F_X, G_X, \Sigma, \mathcal{A}_X, \mathcal{P}_X, \mathcal{S}_X)$ and $Y = (L_Y, V_Y, \nu_Y, W_Y, \omega_Y, F_Y, G_Y, \Sigma, \mathcal{A}_Y, \mathcal{P}_Y, \mathcal{S}_Y)$ be two CPDPs. $CP(X, Y, |^P_A|)$, which we call the set of *conflict pairs* of X and Y under $|^P_A|$, is defined as

$$CP(X,Y,|^P_A|) := \{((\xi_1,\xi_2),\sigma) \in E_X \times E_Y \times (\Sigma \cup \bar{\Sigma}) | \sigma \in (\Sigma \backslash A \cup \bar{\Sigma} \backslash P), \xi_1 \xrightarrow{\sigma}, \xi_2 \xrightarrow{\sigma}\}.$$

If $a \in A$ and CPDP X has enabled a-transitions at state ξ_X and CPDP Y has enabled a-transitions at state ξ_Y, then there is no lack of scheduling information at joint state (ξ_X,ξ_Y) for the a-transitions because the internal scheduler of X may choose an a-transition for X and the internal scheduler of Y may choose an a-transition for Y and these two chosen a-transitions synchronize in the composition. Therefore, (ξ_X,ξ_Y,a) is not a conflict pair here. If $a \notin A$, there would be a conflict and this is illustrated in the following example.

Example 8.31. The set of conflict pairs of X and Y of Figure 8.6 under $|^P_A|$ with $A=\emptyset$ equals $\{(\xi_1,\xi_2,a)|\xi_1 \in G_1 \cup G_2, \xi_2 \in G_3 \cup G_4\}$. If (ξ_1,ξ_2,a) is a conflict pair, then internal scheduling of the four a-transitions of $X|^P_A|Y$ at state (ξ_1,ξ_2) cannot be derived from internal schedulers for X and Y. The internal scheduler of X provides at ξ_1 a probability distribution on $\{\alpha_1,\alpha_2\}$ and the internal scheduler of Y provides at ξ_2 a probability distribution on $\{\alpha_3,\alpha_4\}$. However, this does not naturally lead to a probability distribution on $\{\alpha_1,\alpha_2,\alpha_3,\alpha_4\}$ at state (ξ_1,ξ_2). To resolve this problem we introduce the notion of *priority function*.

Definition 8.32. Let X and Y be two CPDPs. A *priority function pri* assigns, under $|^P_A|$, to each conflict pair a probability measure on $\{X,Y\}$. We write $pri(\xi_1,\xi_2,\sigma)(X)$ (and $pri(\xi_1,\xi_2,\sigma)(Y)$) for the probability of X (Y) at conflict pair (ξ_1,ξ_2,σ).

Example 8.33. With this priority function, we maximally use all the information of the internal schedulers of the components. For example, consider CPDPs X and Y of Figure 8.6 with internal schedulers S^i_X and S^i_Y. Let $\xi_1 \in G_1 \cap G_2$, let $\xi_2 \in G_3 \cap G_4$ and let the priority function be such that $pri(\xi_1,\xi_2,a)(X) = \frac{2}{3}$ and $pri(\xi_1,\xi_2,a)(Y) = \frac{1}{3}$. Then the a-transitions of $X|^P_A|Y$ are internally scheduled at state (ξ_1,ξ_2) as: $(\alpha_1,*)$ gets probability $\frac{2}{3}S^i_X(\xi_1)(\alpha_1)$, $(\alpha_2,*)$ gets probability $\frac{2}{3}S^i_X(\xi_1)(\alpha_2)$, $(*,\alpha_3)$ gets probability $\frac{1}{3}S^i_Y(\xi_2)(\alpha_3)$ and $(*,\alpha_4)$ gets probability $\frac{1}{3}S^i_Y(\xi_2)(\alpha_4)$. Here $(\alpha_1,*)$ denotes the transition in $X|^P_A|Y$ where X executes α_1 and Y does not execute a transition, etc.

Composition of internally scheduled CPDPs with priority function *pri* is now formally defined as follows.

Definition 8.34. Let X and Y be CPDPs with internal schedulers S^i_X and S^i_Y. Let CP denote the set of conflict pairs of X and Y under $|^P_A|$. Let *pri* be a priority function for X and Y under $|^P_A|$. We define for each $\xi \in E_g$, $(S^i_X,S^i_Y,pri)(\xi)$ as a mapping from the set of active and passive transitions of $X|^P_A|Y$ to $[0,1]$. Let α and β be active transitions of X and Y respectively, let $\bar{\alpha}$ and $\bar{\beta}$ be passive transitions of X and Y respectively, and let (S^i_X,S^i_Y,pri) be denoted by S^i. With $(\alpha,*)$ we denote the transition of $X|^P_A|Y$ that reflects transition α of X, with $(*,\beta)$ we denote the transition of $X|^P_A|Y$ that reflects transition β of Y, with (α,β) we denote the synchronized transition of α and β, etc. With σ_α and l_α we denote the action and origin location of transition α. We define,

1.
$$S^i(\xi_1,\xi_2)(\alpha,*) :=$$
$$\begin{cases} S^i_X(\xi_1)(\alpha) & \text{if } \xi_1 \in G_\alpha \text{ and } (\xi_1,\xi_2,\sigma_\alpha) \notin CP, \\ pri(\xi_1,\xi_2,\sigma_\alpha)(X)S^i_X(\xi_1)(\alpha) & \text{if } \xi_1 \in G_\alpha \text{ and } (\xi_1,\xi_2,\sigma_\alpha) \in CP. \end{cases}$$

2.
$$S^i(\xi_1,\xi_2)(*,\beta) :=$$
$$\begin{cases} S^i_Y(\xi_2)(\beta) & \text{if } \xi_2 \in G_\beta \text{ and } (\xi_1,\xi_2,\sigma_\beta) \notin CP, \\ pri(\xi_1,\xi_2,\sigma_\beta)(Y)S^i_Y(\xi_2)(\beta) & \text{if } \xi_2 \in G_\beta \text{ and } (\xi_1,\xi_2,\sigma_\beta) \in CP. \end{cases}$$

3.
$$S^i(\xi_1,\xi_2)(\alpha,\beta) :=$$
$$S^i_X(\xi_1)(\alpha)S^i_Y(\xi_2)(\beta) \quad \text{if } (\xi_1,\xi_2) \in G_\alpha \times G_\beta$$

4.
$$S^i(\xi_1,\xi_2)(\alpha,\bar\beta) :=$$
$$\begin{cases} S^i_X(\xi_1)(\alpha)S^i_Y(\xi_2)(\bar\beta) & \text{if } \xi_1 \in G_\alpha, loc(\xi_2) = l_{\bar\beta} \text{ and} \\ & (\xi_1,\xi_2,\sigma_\alpha) \notin CP, \\ pri(\xi_1,\xi_2,\sigma_\alpha)(X)S^i_X(\xi_1)(\alpha)S^i_Y(\xi_2)(\bar\beta) & \text{if } \xi_1 \in G_\alpha, loc(\xi_2) = l_{\bar\beta} \text{ and} \\ & (\xi_1,\xi_2,\sigma_\alpha) \in CP \end{cases}$$

5.
$$S^i(\xi_1,\xi_2)(\bar\alpha,\beta) :=$$
$$\begin{cases} S^i_X(\xi_1)(\bar\alpha)S^i_Y(\xi_2)(\beta) & \text{if } \xi_2 \in G_\beta, loc(\xi_1) = l_{\bar\alpha} \text{ and} \\ & (\xi_1,\xi_2,\sigma_\beta) \notin CP, \\ pri(\xi_1,\xi_2,\sigma_\beta)(Y)S^i_X(\xi_1)(\bar\alpha)S^i_Y(\xi_2)(\beta) & \text{if } \xi_2 \in G_\beta, loc(\xi_1) = l_{\bar\alpha} \text{ and} \\ & (\xi_1,\xi_2,\sigma_\beta) \in CP \end{cases}$$

6.
$$S^i(\xi_1,\xi_2)(\bar\alpha,*) :=$$
$$\begin{cases} S^i_X(\xi_1)(\bar\alpha) & \text{if } loc(\xi_1) = l_{\bar\alpha} \text{ and } (\xi_1,\xi_2,\sigma_{\bar\alpha}) \notin CP, \\ pri(\xi_1,\xi_2,\sigma_{\bar\alpha})(X)S^i_X(\xi_1)(\bar\alpha) & \text{if } loc(\xi_1) = l_{\bar\alpha} \text{ and } (\xi_1,\xi_2,\sigma_{\bar\alpha}) \in CP. \end{cases}$$

7.
$$S^i(\xi_1,\xi_2)(*,\bar\beta) :=$$
$$\begin{cases} S^i_Y(\xi_2)(\bar\beta) & \text{if } loc(\xi_2) = l_{\bar\beta} \text{ and } (\xi_1,\xi_2,\sigma_{\bar\beta}) \notin CP, \\ pri(\xi_1,\xi_2,\sigma_{\bar\beta})(Y)S^i_Y(\xi_2)(\bar\beta) & \text{if } loc(\xi_2) = l_{\bar\beta} \text{ and } (\xi_1,\xi_2,\sigma_{\bar\beta}) \in CP. \end{cases}$$

8.
$$S^i(\xi_1,\xi_2)(\bar\alpha,\bar\beta) :=$$
$$S^i_X(\xi_1)(\bar\alpha)S^i_Y(\xi_2)(\bar\beta) \quad \text{if } loc(\xi_1) = l_{\bar\alpha} \text{ and } loc(\xi_2) = l_{\bar\beta}.$$

Theorem 8.35. *S^i in Definition 8.34 is a well-defined internal scheduler for CPDP $X|^P_A|Y$.*

Proof. To prove that S^i is an internal scheduler, we have to show that for all $\sigma \in \Sigma \cup \bar{\Sigma}$ and all states ξ such that $\mathcal{T}_{\xi \xrightarrow{\sigma}}$, which is the set of all ξ-enabled active or passive transitions, is not empty we have that $\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) = 1$.

We distinguish six cases: 1. $\sigma \in A$, 2. $\sigma \in \Sigma\backslash A$ and $(\xi_1, \xi_2, \sigma) \notin CP$, 3. $\sigma \in \Sigma\backslash A$ and $(\xi_1, \xi_2, \sigma) \in CP$, 4. $\sigma \in P$, 5. $\sigma \in \bar{\Sigma}\backslash P$ and $(\xi_1, \xi_2, \sigma) \notin CP$, 6. $\sigma \in \bar{\Sigma}\backslash P$ and $(\xi_1, \xi_2, \sigma) \in CP$. Given a state $\xi = (\xi_1, \xi_2)$ and an action σ such that there is a σ-transition enabled at state ξ, exactly one of the above six cases is true. We show that for all six cases we have $\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) = 1$.

1. $\sigma \in A$:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{3}{=} \sum_{(\alpha,\beta) \in \mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}} \times \mathcal{A}_{Y,\xi_1 \xrightarrow{\sigma}}} S^i_X(\xi_1)(\alpha) S^i_Y(\xi_2)(\beta) =$$

$$\sum_{\alpha \in \mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}}} (S^i_X(\xi_1)(\alpha) \sum_{\beta \in \mathcal{A}_{Y,\xi_2 \xrightarrow{\sigma}}} S^i_Y(\xi_2)(\beta)) = \sum_{\alpha \in \mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}}} S^i_X(\xi_1)(\alpha) = 1,$$

where $\overset{3}{=}$ means that this step follows from item 3 of Definition 8.34.

2. $\sigma \in \Sigma\backslash A$ and $(\xi_1, \xi_2, \sigma) \notin CP$: we distinguish four subcases: a. $\mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}} = \emptyset$ and $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} = \emptyset$, b. $\mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}} = \emptyset$ and $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} \neq \emptyset$, c. $\mathcal{A}_{Y,\xi_2 \xrightarrow{\sigma}} = \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} = \emptyset$, d. $\mathcal{A}_{Y,\xi_2 \xrightarrow{\sigma}} = \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} \neq \emptyset$.

That exactly one of these four cases is true follows from the fact that $(\xi_1, \xi_2, \sigma) \notin CP$. Case a:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{2}{=} \sum_{\beta \in \mathcal{A}_{Y,\xi_2 \xrightarrow{\sigma}}} S^i_Y(\xi_2)(\beta) = 1.$$

Case b: $\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{5}{=}$

$$\sum_{\beta \in \mathcal{A}_{Y,\xi_2 \xrightarrow{\sigma}}} (S^i_Y(\xi_2)(\beta) \sum_{\alpha \in \mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}}} S^i_X(\xi_1)(\alpha)) = \sum_{\beta \in \mathcal{A}_{Y,\xi_2 \xrightarrow{\sigma}}} S^i_Y(\xi_2)(\beta) = 1.$$

Case c:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{1}{=} \sum_{\alpha \in \mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}}} S^i_X(\xi_1)(\alpha) = 1.$$

Case d: $\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{4}{=}$

$$\sum_{\alpha \in \mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}}} (S^i_X(\xi_1)(\alpha) \sum_{\beta \in \mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}}} S^i_Y(\xi_2)(\beta)) = \sum_{\alpha \in \mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}}} S^i_X(\xi_1)(\alpha) = 1.$$

3. $\sigma \in \Sigma\backslash A$ and $(\xi_1, \xi_2, \sigma) \in CP$: We distinguish four subcases: a. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} = \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} = \emptyset$, b. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} \neq \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} = \emptyset$, c. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} = \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} \neq \emptyset$, d. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} \neq \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} \neq \emptyset$, Note that in all four cases we have $\mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}} \neq \emptyset$ and $\mathcal{A}_{Y,\xi_2 \xrightarrow{\sigma}} \neq \emptyset$, because $(\xi_1, \xi_2, \sigma) \in CP$.

Define

$$\tilde{A} := \sum_{\alpha \in \mathcal{A}_{X,\xi_1 \xrightarrow{\sigma}}} pri(\xi_1, \xi_2, \sigma)(X) S_X^i(\xi_1)(\alpha), \quad \tilde{B} := \sum_{\beta \in \mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}}} S_Y^i(\xi_2)(\beta),$$

$$\tilde{C} := \sum_{\beta \in \mathcal{A}_{Y,\xi_2 \xrightarrow{\sigma}}} pri(\xi_1, \xi_2, \sigma)(Y) S_Y^i(\xi_2)(\beta), \quad \tilde{D} := \sum_{\alpha \in \mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}}} S_X^i(\xi_1)(\alpha).$$

Then we get: Case a:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{1,2}{=} \tilde{A} + \tilde{C} = pri(\xi_1, \xi_2, \sigma)(X) + pri(\xi_1, \xi_2, \sigma)(Y) = 1.$$

Case b:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{1,5}{=} \tilde{A} + \tilde{C}\tilde{D} = pri(\xi_1, \xi_2, \sigma)(X) + pri(\xi_1, \xi_2, \sigma)(Y) = 1.$$

Case c:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{2,4}{=} \tilde{A}\tilde{B} + \tilde{C} = pri(\xi_1, \xi_2, \sigma)(X) + pri(\xi_1, \xi_2, \sigma)(Y) = 1.$$

Case d:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\xi)(\alpha) \overset{4,5}{=} \tilde{A}\tilde{B} + \tilde{C}\tilde{D} = pri(\xi_1, \xi_2, \sigma)(X) + pri(\xi_1, \xi_2, \sigma)(Y) = 1.$$

4. $\sigma \in P$: we distinguish three subcases: a. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} = \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} \neq \emptyset$, b. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} \neq \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} = \emptyset$, c. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} \neq \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} \neq \emptyset$.
Case a:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\bar{\sigma}}}} S^i(\xi)(\alpha) \overset{7}{=} \sum_{\beta \in \mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}}} S_Y^i(\xi_2)(\beta) = 1,$$

Case b:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\bar{\sigma}}}} S^i(\xi)(\alpha) \overset{6}{=} \sum_{\alpha \in \mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}}} S_X^i(\xi_1)(\alpha) = 1,$$

Case c:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\bar{\sigma}}}} S^i(\xi)(\alpha) \overset{8}{=} \sum_{\alpha \in \mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}}} S_X^i(\xi_1)(\alpha) \sum_{\beta \in \mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}}} S_Y^i(\xi_2)(\beta) = 1.$$

5. $\sigma \in \bar{\Sigma} \setminus P$ and $(\xi_1, \xi_2, \sigma) \notin CP$: we distinguish two subcases: a. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} \neq \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} = \emptyset$, b. $\mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}} = \emptyset$ and $\mathcal{P}_{Y,\xi_2 \xrightarrow{\bar{\sigma}}} \neq \emptyset$.
Case a:

$$\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\bar{\sigma}}}} S^i(\xi)(\alpha) \overset{6}{=} \sum_{\alpha \in \mathcal{P}_{X,\xi_1 \xrightarrow{\bar{\sigma}}}} S_X^i(\xi_1)(\alpha) = 1,$$

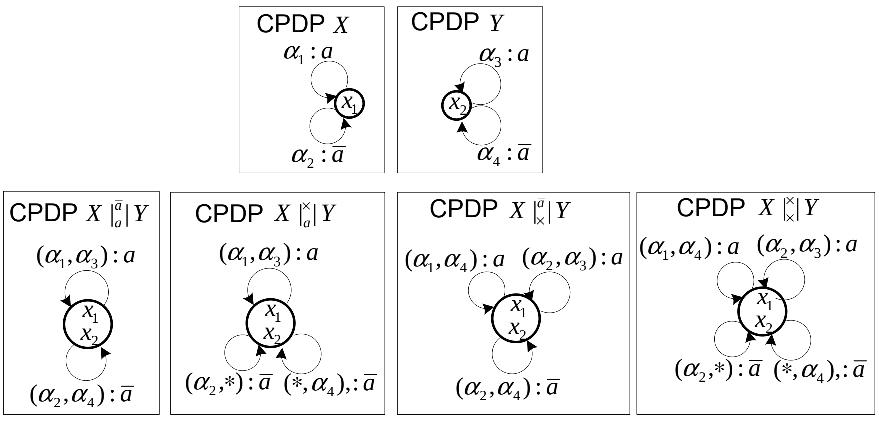
CPDP X
$\alpha_1 : a$
x_1
$\alpha_2 : \bar{a}$
CPDP Y
$\alpha_3 : a$
x_2
$\alpha_4 : \bar{a}$
CPDP $X\,|^{\bar{a}}_{a}|\,Y$
$(\alpha_1,\alpha_3) : a$
x_1 x_2
$(\alpha_2,\alpha_4) : \bar{a}$
CPDP $X\,|^{\times}_{a}|\,Y$
$(\alpha_1,\alpha_3) : a$
x_1 x_2
$(\alpha_2,*) : \bar{a}$ $(*,\alpha_4),: \bar{a}$
CPDP $X\,|^{\bar{a}}_{\times}|\,Y$
$(\alpha_1,\alpha_4) : a$ $(\alpha_2,\alpha_3) : a$
x_1 x_2
$(\alpha_2,\alpha_4) : \bar{a}$
CPDP $X\,|^{\times}_{\times}|\,Y$
$(\alpha_1,\alpha_4) : a$ $(\alpha_2,\alpha_3) : a$
x_1 x_2
$(\alpha_2,*) : \bar{a}$ $(*,\alpha_4),: \bar{a}$

Figure 8.7: Example of composition of internal schedulers

Case b:
$$\sum_{\alpha\in\mathcal{T}_{\xi\xrightarrow{\bar{\sigma}}}} S^i(\xi)(\alpha) \overset{7}{=} \sum_{\beta\in\mathcal{P}_{Y,\xi_2\xrightarrow{\bar{\sigma}}}} S^i_Y(\xi_2)(\beta) = 1.$$

6. $\sigma \in \bar{\Sigma}\backslash P$ and $(\xi_1,\xi_2,\sigma) \in CP$: there is only one case,

$$\sum_{\alpha\in\mathcal{T}_{\xi\xrightarrow{\bar{\sigma}}}} S^i(\xi)(\alpha) \overset{8}{=} \sum_{\alpha\in\mathcal{P}_{X,\xi_1\xrightarrow{\bar{\sigma}}}} pri(\xi_1,\xi_2,\bar{\sigma})(X)S^i_X(\xi_1)(\alpha) +$$

$$\sum_{\beta\in\mathcal{P}_{Y,\xi_2\xrightarrow{\bar{\sigma}}}} pri(\xi_1,\xi_2,\bar{\sigma})(Y)S^i_Y(\xi_2)(\beta) = pri(\xi_1,\xi_2,\bar{\sigma})(X) + pri(\xi_1,\xi_2,\bar{\sigma})(Y) = 1.$$

□

Example 8.36. To illustrate the composition rules of Definition 8.34 we compose CPDPs X and Y of Figure 8.7 in four different ways, i.e., with four different composition operators. The four composition operators are pictured in Figure 8.7, where $|^{\bar{a}}_{a}|$ stands for $|^{\{\bar{a}\}}_{\{a\}}|$, $|^{\times}_{a}|$ stands for $|^{\emptyset}_{\{a\}}|$, etc. The guards of the a-transitions of X and Y equal the whole state spaces of x_1 and x_2 respectively. We assume that a priority function pri is given. The internal schedulers S^i_X and S^i_Y for X and Y are trivial since for both a and $\bar{a}$ there is for both X and Y exactly one transition with that label. We now treat the four cases separately.

- $X|^{\bar{a}}_{a}|Y$. In this case, α_1 and α_3 must synchronize and α_2 and α_4 must synchronize. The set of conflict pairs is empty here. This means that the priority function is not needed in this case to do the internal scheduling. According to rule 3, the scheduling value for (α_1,α_3) at state (ξ_1,ξ_2) equals $S^i_X(\xi_1)(\alpha_1)S^i_Y(\xi_2)(\alpha_3) = 1$. Analogously, according to rule 8, the scheduling value for (α_2,α_4) also equals one for all (ξ_1,ξ_2).

- $X|_a^\times|Y$. In this case α_1 and α_3 must synchronize and α_2 and α_4 must interleave. The set of conflict pairs equals $\{(\xi_1, \xi_2, \bar{a}) | \xi_1 \in E_X, \xi_2 \in E_Y\}$, where E_X and E_Y denote the state spaces of X and Y. If $X|_a^\times|Y$ is composed with another CPDP Z via $|_A^P|$ with $a \notin A$, then an a-transition of Z synchronizes with either α_2 or α_4, but not with both. The choice between α_2 and α_4 is made by the priority function and is expressed by rules 6 and 7, which says that the probability that α_2 synchronizes at state (ξ_1, ξ_2) with the a-transition of Z equals $pri(\xi_1, \xi_2, \bar{a})(X)$ and the probability that α_4 synchronizes equals $pri(\xi_1, \xi_2, \bar{a})(Y)$.

- $X|_\times^{\bar{a}}|Y$. In this case, α_1 and α_3 must interleave and α_2 and α_4 must synchronize. The set of conflict pairs equals $\{(\xi_1, \xi_2, a) | \xi_1 \in E_X, \xi_2 \in E_Y\}$. The choice between α_1 and α_3 is made by the priority function and is expressed by rules 1 and 2.

- $X|_\times^\times|Y$. In this case, α_1 and α_3 must interleave and α_2 and α_4 must interleave. The set of conflict pairs equals $\{(\xi_1, \xi_2, \sigma) | \xi_1 \in E_X, \xi_2 \in E_Y, \sigma \in \{a, \bar{a}\}\}$. The choice between α_1 and α_3 and the choice between α_2 and α_4 is made by the priority function and is expressed by rules 1,2,6 and 7.

Example 8.37. For the last time we revisit the repair shop system of Section 6.4. Suppose that $s_1 = s_2$. This means that if both machines do not break down before their ages s_1 and s_2 are reached, then both machines should be brought to the repair shop at the same time. We want to model that both machines have equal probability to be brought to the repair shop first.

This is expressed by defining a priority function pri which assigns probability $\frac{1}{2}$ to both X and Y at conflict pair $((l_{1,0}, x_1 = s_1), (l_{2,0}, x_2 = s_2, down))$. CPDPs M_1, M_2 and R (see Figure 7.5), have trivial internal schedulers since at each location there is at most one active transition enabled, which then gets probability one of these trivial internal schedulers. Denote these trivial internal schedulers by $S^i_{M_1}$, $S^i_{M_2}$ and S^i_R. Then the internally scheduled composition of M_1 and M_2 results in internally scheduled CPDP $(M_1|_\emptyset^\emptyset|M_2, (S^i_{M_1}, S^i_{M_2}, pri))$, which can be composed with internally scheduled CPDP (R, S^i_R). Since external scheduling is not needed, the internal scheduler is a scheduler itself.

9

Bisimulation for CPDPs

As we already stated in the introduction chapter, it is well-known that the composition of multiple subsystems leads to state space explosion and one tool that has proved to be effective in dealing with the state space explosion problem is bisimulation. Bisimulation can be seen as a state space reduction technique: by bisimulation we can find systems with smaller state spaces, that still have the same external behavior. Two systems have the same external behavior if they cannot be distinguished in any composition context.

The notion of bisimulation was introduced by Milner in the context of discrete state processes (see for example [Mil89] for this notion of bisimulation). Since then, bisimulation has also been established in the context of probabilistic and stochastic automata [LS91, D'A97], continuous time interactive Markov chains (IMC) [Her02], continuous dynamical systems [Pap03, vdS04a, vdS04b] and general (non-stochastic) hybrid systems [LPS00, vdS04c]. A category theoretic approach to bisimulation for general stochastic hybrid systems can be found in [BLB05]. For an overview of bisimulation and its decidability for non-stochastic hybrid systems, we refer to [AHLP00].

In this chapter, we define bisimulation in the context of CPDPs. In some sense, this notion of bisimulation for CPDPs integrates the notions of bisimulation for IMC, stochastic automata and continuous/hybrid systems.

An important point is that under maximal progress scheduled CPDPs have a pure stochastic behavior, expressed by the semantical model TMS. We want to define bisimulation in such a way that the TMSs of bisimilar CPDPs are 'equivalent' as far as it concerns the output dynamics. This specific notion of 'equivalence' will be formalized in this chapter by introducing the concept of *quotient TMS*.

We show that the bisimulation substitutivity property holds for CPDPs. From an analysis point of view, we can then reduce the state space of a composite CPDP in a compositional way by substituting components by state-reduced bisimilar components.

We do not consider time abstracted notions of bisimulation. With a time-abstracted notion of bisimulation the amount of time before a state with certain properties is reached does not matter (i.e., if in one system this period is smaller than in another system, then they can still be bisimilar). Roughly said, in our notion of bisimulation, two bisimilar systems should reach 'equivalent' states in the same amount of time. Time abstracted bisimulation generally results in greater state space reduction than non-time abstracted bisimulation. In [AHLP00], time abstracted bisimulation might, because of the time abstraction, result in finite bisimulations (i.e. finite-amount-of-states systems) for systems that originally have infinite states, while our notion of bisimulation does not result in a finite state system for the same original systems. (Note that this does not mean that with our notion of bisimulation, finite state and infinite state systems can never be bisimilar. There are for example infinite state CPDPs that are bisimilar to finite state IMCs).

The organization of this chapter is as follows. In Section 9.1 we define bisimulation for CPDPs and we prove the substitutivity property. In Section 9.2 we do the same for internally scheduled CPDPs and we explain why bisimulation should not be defined on the level of (fully) scheduled CPDPs. In Section 9.3 we present an algorithm to automatically find bisimulations for CPDPs and for internally scheduled CPDPs and we discuss decidability conditions for this algorithm.

9.1 Bisimulation for CPDPs

As an introduction to the concept of *equivalent probability measures*, we give a simple example. Suppose we have a state space E with four states, ξ_1, ξ_2, ξ_3 and ξ_4. Suppose that we regard ξ_1 and ξ_2 as equivalent, while we regard ξ_1, ξ_3 and ξ_4 as different from each other. Then, this equivalence is expressed by equivalence relation $\mathcal{R} = \{\{\xi_1, \xi_2\}, \{\xi_3\}, \{\xi_4\}\}$. Suppose now we have two probability measures on E, Q_1 and Q_2. When do we regard Q_1 and Q_2 as being equivalent? The answer is: if they assign the same probabilities to each equivalence class of $\mathcal{R}$. Thus, if $Q_1(\xi_1) = 1$, $Q_1(\xi_2) = 0$, $Q_2(\xi_1) = 0$ and $Q_2(\xi_2) = 1$, while $Q_1(\xi_3) = Q_2(\xi_3)$ and $Q_1(\xi_4) = Q_2(\xi_4)$, then, although the probability measures are different, they are equivalent because they assign the same probabilities to all equivalence classes. We call the set of equivalence classes $\{[\xi_1], [\xi_3], [\xi_4]\}$ together with its measurable subsets (i.e., all subsets in this case) the quotient space of E. Q_1 and Q_2 can be defined on this quotient space as $\tilde{Q}_1$ and $\tilde{Q}_2$, where $\tilde{Q}_1([\xi_1]) = Q_1(\{\xi_1, \xi_2\})$, $\tilde{Q}_1([\xi_3]) = Q_1(\xi_3)$, $\tilde{Q}_1([\xi_4]) = Q_1(\xi_4)$ and the same for Q_2. Then the probability measures $\tilde{Q}_1$ and $\tilde{Q}_2$ are the same. In general we could therefore define two probability measures to be equivalent if they assign the same probabilities to the measurable subsets of the quotient space.

For a discrete set, the measurable subsets are all the subsets of that set. For (continuous) Borel spaces, the measurable sets are the Borel sets. If we look at the quotient space of a Borel space E, we have to take care that the measurable subsets of this quotient space correspond to Borel sets of E. We formalize this as follows.

Definition 9.1. Let $(E, \mathcal{E})$ be a Borel space and let $\mathcal{R}$ be an equivalence relation on E. Let $\mathcal{E}^*$ be the collection of all $\mathcal{R}$-saturated Borel sets of E, i.e., all $B \in \mathcal{E}$ such that any equivalence class of E is either totally contained or totally not contained in B. $\mathcal{E}^*$ is a σ-algebra:

Item 1 of Definition 2.2 if $A \in \mathcal{E}^*$, then A is a Borel set which can be written as a union of equivalence classes. Then $E \backslash A$ is also Borel and consists of the union of all equivalence classes that are not in A, therefore $E\backslash A \in \mathcal{E}^*$.

Item 2 of Definition 2.2 Let A_i, for i in some countable index set I, be in $\mathcal{E}^*$. Then, by definition of a σ-algebra (see Chapter 2), $\cup_{i\in I}A_i$ is also a Borel set. $\cup_{i\in I}A_i$ is clearly a union of equivalence classes since this is the case for each A_i. Therefore, $\cup_{i\in I}A_i \in \mathcal{E}^*$.

Let

$$\mathcal{E}^*/_{\mathcal{R}} := \{[A] | A \in \mathcal{E}^*\},$$

where $[A] := \{[a] | a \in A\}$ and $[a]$ denotes the equivalence class of a. Then $(E/_{\mathcal{R}}, \mathcal{E}^*/_{\mathcal{R}})$, which is a measurable space, is called the *quotient space* of E with respect to $\mathcal{R}$.

For our purposes it is desirable that the quotient space of a Borel space is again a Borel space. This property depends on the equivalence relation that determines the quotient space. Therefore, we define:

Definition 9.2. An equivalence relation $\mathcal{R}$ on a Borel space E is called *measurable* if the quotient space of E with respect to $\mathcal{R}$ is a Borel space.

In the definition of TMS (see Definition 3.1), we assume that the state space of the TMS is a Borel space. As we will see when we later compare the stochastic behavior of different TMSs, this assumption is needed to assure that the TMS represents a well-defined stochastic process. We will use bisimulation as a state reduction technique. The state space of a TMS can then be reduced to its quotient space, as we will see later. In order that this state reduced TMS is again a TMS, we need that the quotient space is a Borel space. This is why we need a measurable relation, as defined in Definition 9.2, for bisimulation.

Now we can define when two probability measures on a Borel space that is partitioned by a measurable relation, are equivalent.

Definition 9.3. Let $\mathcal{R}$ be a measurable relation on Borel space E and let m_1 and m_2 be measures on E. m_1 and m_2 are called *equivalent with respect to* $\mathcal{R}$ if for all $A \in \mathcal{E}^*$ $m_1(A) = m_2(A)$.

Note that this means that equivalent measures are equal on the quotient space.

We have now all ingredients to define the notion bisimulation in the context of stochastic hybrid systems. We start by defining bisimulation on the semantical level by defining bisimulation for TMSs. Then we rise one level in the semantical domains and we define bisimulation for CFSJSs and FTSs. Then we define bisimulation for NTSs and finally we define bisimulation for CPDPs.

Definition 9.4. Let $X = (E, \xi_0, \phi, (T, Q))$ be a TMS and let O be an output mapping from E to the output space E_O. A measurable relation $\mathcal{R}$ on E is called a *bisimulation* for (X, O) if: $(\xi_1, \xi_2) \in \mathcal{R}$ implies

1. $O(\xi_1) = O(\xi_2)$,
2. $(\phi(t, \xi_1), \phi(t, \xi_2)) \in \mathcal{R}$ for all $t > 0$,

3. $P(T(\xi_1) > t) = P(T(\xi_2) > t)$ for all $t > 0$,
4. the measures $Q(\xi_1)$ and $Q(\xi_2)$ are equivalent with respect to $\mathcal{R}$.

Let $\mathcal{R}$ be a bisimulation for (X, O). We call $X/_{\mathcal{R}} = (E/_{\mathcal{R}}, [\xi_0], \phi/_{\mathcal{R}}, (T/_{\mathcal{R}}, Q/_{\mathcal{R}}))$, where $E/_{\mathcal{R}}$ is the set of equivalence classes of E, $[\xi_0]$ is the equivalence class of ξ_0, $\phi/_{\mathcal{R}}([\xi]) := [\phi(\xi)]$, $T/_{\mathcal{R}}([\xi]) = T(\xi)$ and for all $A \in \mathcal{B}^*(E)$ $Q/_{\mathcal{R}}([\xi])([A]) = Q(\xi)(A)$, the *quotient TMS* of X under $\mathcal{R}$. The corresponding quotient output mapping is defined as $O/_{\mathcal{R}}([\xi]) = O(\xi)$.

Note that the quotient TMS is a well-defined TMS since $E/_{\mathcal{R}}$ is a Borel space according to Definition 9.2. If we generate an execution path $\xi : \mathbb{R}_+ \to E$ for a TMS X, then we call $O(\xi(t))$, $t > 0$, an output execution path for (X, O). We claim that if $\mathcal{R}$ is a bisimulation for X, then the quotient TMS $(X/_{\mathcal{R}}, O/_{\mathcal{R}})$ generates the same output execution paths (with the same probabilities) as (X, O). This can be seen as follows. The initial states ξ_0 and $[\xi_0]$ have the same output. The distributions of the first switching time, $T(\xi_0)$ and $T/_{\mathcal{R}}([\xi_0])$, are the same. The output trajectories till the first jump are the same. For any $t > 0$ we have that for all Borel sets $A \in \mathcal{E}^*$ and corresponding $[A]$ $Q(\phi(t, \xi_0))(A)$ and $Q_{\mathcal{R}}(\phi_{\mathcal{R}}(t, [\xi_0]))([A])$ are the same. Or, roughly said, the probabilities that $X_{\mathcal{R}}$ jumps to state $[\xi]$ and that X jumps to a state in the equivalence class of ξ are the same for all ξ, etc.

Example 9.5. Let $X = (E, \xi_0, \phi, (T, Q))$ be a TMS with $E = [0,1] \times [0,1]$, $\xi_0 = (0,0)$, $\phi(t, (\xi_1, \xi_2)) = (\xi_1 + t, \xi_2)$, $T(\xi_1, \xi_2)$ is a uniform distribution on $[0, 1 - \xi_1]$ and $Q(\xi_1, \xi_2) = Id_0 \times U[0,1]$, i.e., Q resets the first component to zero and resets the second component with a uniform distribution. We define an output mapping $O : E \to [0,1]$ as $O(\xi_1, \xi_2) = \xi_1$. It can be easily checked that $\mathcal{R} := \{\{r\} \times [0,1] | r \in [0,1]\}$ is a bisimulation for (X, O). The quotient TMS then equals $X/_{\mathcal{R}} = (E/_{\mathcal{R}}, [\xi_0], \phi/_{\mathcal{R}}, (T/_{\mathcal{R}}, Q/_{\mathcal{R}}))$, where $E/_{\mathcal{R}} = [0,1]$, $[\xi_0] = 0$, $\phi/_{\mathcal{R}}(t, \xi) = \xi + t$, $T/_{\mathcal{R}}(\xi) = U[0, 1 - \xi]$ and $Q/_{\mathcal{R}}(\xi) = Id_0$.

Definition 9.6. Let $X = (E, \xi_0, \phi, \lambda, Q)$ be a CFSJS. A measurable relation $\mathcal{R}$ on E is called a *bisimulation* for X if $(\xi_1, \xi_2) \in \mathcal{R}$ implies that

1. $\lambda(\xi_1) = \lambda(\xi_2)$,
2. $(\phi(t, \xi_1), \phi(t, \xi_2)) \in \mathcal{R}$ for all $t > 0$,
3. the measures $Q(\xi_1)$ and $Q(\xi_2)$ are equivalent with respect to $\mathcal{R}$.

Example 9.7. To illustrate bisimulation for CFSJSs, we adapt Example 9.5. Let $X = (E, \xi_0, \phi, \lambda, Q)$ be a CFSJS with $E = [0,1] \times [0,1]$, $\xi_0 = (0,0)$, $\phi(t, (\xi_1, \xi_2)) = (\xi_1 + t, \xi_2)$, $\lambda(\xi_1, \xi_2) = \tilde{\lambda}(\xi_1)$, where $\tilde{\lambda}$ is some jump rate function defined on $[0,1]$, and $Q(\xi_1, \xi_2) = Id_0 \times U[0,1]$. Then, $\mathcal{R} := \{\{r\} \times [0,1] | r \in [0,1]\}$ is a bisimulation for CFSJS X.

Next, we define bisimulation for FTS.

Definition 9.8. Let $X = (E, \mathcal{T})$ be an FTS. Let $O : E \to E_O$ be an output mapping. A measurable relation $\mathcal{R}$ on E is called a *bisimulation relation* for (X, O) if:

1. $(\xi_1, \xi_2) \in \mathcal{R}$ implies $O(\xi_1) = O(\xi_2)$,
2. $(\xi_1, \xi_2) \in \mathcal{R}$ and $(\xi_1, m_1) \in \mathcal{T}$ imply that there exists m_2 such that $(\xi_2, m_2) \in \mathcal{T}$ while m_1 and m_2 are equivalent with respect to $\mathcal{R}$.

Later we will use the following lemma for proving stochastic equivalence of bisimilar scheduled CPDPs.

Lemma 9.9. *Let X_C be a CFSJS with state space E, let X_F be an FTS with the same state space E and let $O : E \to E_O$ be an output mapping. If the measurable relation $\mathcal{R}$ on E is a bisimulation for both X_C and (X_F, O), then $\mathcal{R}$ is also a bisimulation for (X_T, O), where X_T is the TMS that corresponds to X_C and X_F.*

Proof. That items 1,2 and 4 of Definition 9.4 are satisfied follows directly from Definitions 9.6 and 9.8. Let $(\xi_1, \xi_2) \in \mathcal{R}$. Consider the following observations. 1. From items 1 and 2 of Definition 9.6 we find that the distribution of the time of a stochastic jump due to the CFSJS is the same at states ξ_1 and ξ_2, 2. Bisimilar states are either both enabled FTS states or both non-enabled FTS states, which, together with item 2 of Definition 9.6, makes the time until a forced jump happens the same at states ξ_1 and ξ_2. From these two observations we find directly that item 3 of Definition 9.4 is satisfied. □

Definition 9.10. Let $X = (E, \Sigma, \mathcal{T})$ be an NTS and let $O : E \to E_O$ be an output mapping. A measurable relation $\mathcal{R}$ on E is called a *bisimulation relation* for (X, O) if: $(\xi_1, \xi_2) \in \mathcal{R}$ implies for all $\sigma \in \Sigma$ that

1. $O(\xi_1) = O(\xi_2)$,
2. if $\xi_1 \xrightarrow{\sigma} m_1$ then there exists an m_2 such that $\xi_2 \xrightarrow{\sigma} m_2$, while m_1 and m_2 are equivalent measures with respect to $\mathcal{R}$.

Remark 9.11. The idea that for bisimilar states ξ_1 and ξ_2 an a-transition at ξ_1 with measure m_1 should have a corresponding a-transition at ξ_2 with equivalent measure m_2 is also present in the definition of bisimulation for probabilistic timed automata in [D'A97], but there measures of different transitions can be defined on different measurable spaces, where we consider the same Borel space for all measures. Equivalence of measures m_1 and m_2 is in [D'A97] defined as $m_1(A) = m_2(A)$ for each set A that is measurable in both the measurable spaces of m_1 and m_2.

Example 9.12. To illustrate bisimulation for NTSs, we give a simple example of an NTS with a discrete state space of three states. This NTS $X = (\{l_0, l_1, l_2\}, \{a, b\}, \mathcal{T})$ is pictured in Figure 9.1 $\mathcal{T}$ consists of three transitions: an a-transition from l_0, whose reset map assigns probability $\frac{1}{3}$ to l_1 and assigns probability $\frac{2}{3}$ to l_3, a b-transition from l_1 to itself (with probability one), and a b-transition from l_2 that returns to itself with probability $\frac{1}{2}$ and goes to l_1 with probability $\frac{1}{2}$. We consider output space $O = \{white, black\}$ and output mapping $O(l_0) = white$, $O(l_1) = O(l_2) = black$. We show that $\mathcal{R} = \{\{l_0\}, \{l_1, l_2\}\}$ is a bisimulation for

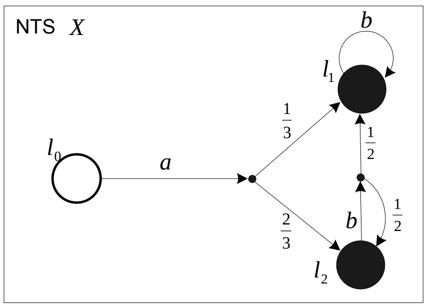

Figure 9.1: Bisimulation for NTSs

(X, O): Condition 1 is satisfied because $O(l_1) = O(l_2)$, condition 2 is satisfied because both the b-transition from l_1 and the b-transition from l_2 end up in equivalence class $\{l_1, l_2\}$ with probability one.

We now define bisimulation for CPDPs.

Definition 9.13. Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P} = \emptyset, \mathcal{S})$ be a CPDP with state space E and flow map ϕ. A measurable relation $\mathcal{R}$ on E is called a *bisimulation* for X if: $(\xi_1, \xi_2) \in \mathcal{R}$ implies

1. $\omega(loc(\xi_1)) = \omega(loc(\xi_2))$ and for all $w \in \omega(loc(\xi_1))$, $G(loc(\xi_1), w)(\xi_1) = G(loc(\xi_2), w)(\xi_2)$,
2. $(\phi(t, \xi_1), \phi(t, \xi_2)) \in \mathcal{R}$ for all $t > 0$,
3. $\sum_{\alpha \in \mathcal{S}_{loc(\xi_1)\rightarrow}} \lambda_\alpha(\xi_1) = \sum_{\alpha \in \mathcal{S}_{loc(\xi_2)\rightarrow}} \lambda_\alpha(\xi_2)$ and, if this total jump rate, which we denote by λ, is not equal to zero, then
$$\sum_{\alpha \in \mathcal{S}_{loc(\xi_1)\rightarrow}} \frac{\lambda_\alpha(\xi_1)}{\lambda(\xi_1)} rmap(\alpha)(\xi_1) \quad \text{and} \quad \sum_{\alpha \in \mathcal{S}_{loc(\xi_2)\rightarrow}} \frac{\lambda_\alpha(\xi_2)}{\lambda(\xi_2)} rmap(\alpha)(\xi_2)$$
are equivalent measures, and
4. for all $\sigma \in \Sigma \cup \bar{\Sigma}$ we have that if $\xi_1 \xrightarrow{\sigma} m_1$, i.e., if there exists a σ-transition enabled at ξ_1 with reset measure m_1, then there exists a reset measure m_2 such that 1. $\xi_2 \xrightarrow{\sigma} m_2$ and 2. m_1 and m_2 are equivalent measures.

Two states ξ_1 and ξ_2 in E are called *bisimilar* if there exists some bisimulation $\mathcal{R}$ for X such that $(\xi_1, \xi_2) \in \mathcal{R}$.

The conditions of Definition 9.13 can be interpreted as follows. Condition 1 assures that the outputs of bisimilar states are the same. Condition 2 assures that through continuous evolution, the states remain bisimilar. Condition 3 assures that the combined action of the spontaneous transitions is equivalent at bisimilar states

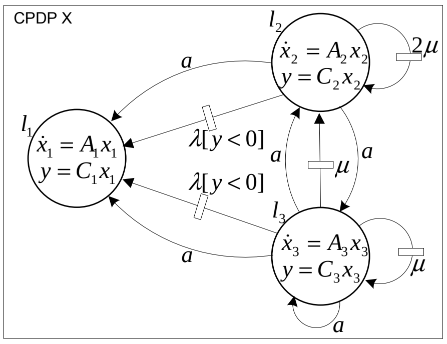

Figure 9.2: Bisimulation for CPDPs

(see Section 3.2.2 for combined action of spontaneous transitions). Condition 4 assures that for each active or passive σ-transition at state ξ_1, there is also a σ-transition at state ξ_2 with an equivalent reset measure.

Remark 9.14. It can be seen that the bisimulation notions for IMC and for CPDP coincide for all IMCs transformed to CPDP (see Section 7.1.2). In other words, if $\mathcal{R} \subset L \times L$ is a strong bisimulation for IMC X with discrete state space L, then $\{((l,0),(l',0)) | (l,l') \in \mathcal{R}\}$ is a bisimulation for CPDP X_C, where X_C, with state space $E = \cup_{l \in L}(l,0)$, denotes the transformation of X.

Bisimulation for CPDPs is reflected on the semantical level as follows.

Lemma 9.15. *$\mathcal{R}$ is a bisimulation for a CPDP X if and only if $\mathcal{R}$ is a bisimulation for both the CFSJS and the NTS of X.*

Proof. The following observations can be easily checked. Items 1 and 4 of Definition 9.13 are satisfied if and only if items 1 and 2 of Definition 9.10 are satisfied. Item 2 of Definition 9.13 is satisfied if and only if item 2 and 2 of Definition 9.6 is satisfied. Item 3 of Definition 9.13 is satisfied if and only if items 1 and 3 of Definition 9.6 are satisfied. The latter observation follows from the fact that the two measures of item 3 of Definition 9.13, which should be equivalent, are exactly the transition measures of the CFSJS at states ξ_1 and ξ_2. □

Example 9.16. Consider the CPDP X of Figure 9.2. We define three initial values $r_{0,1}$, $r_{0,2}$ and $r_{0,3}$. These values determine initial states for locations l_1, l_2 and l_3, i.e., the initial state of location l_i ($i = 1,2,3$) equals $\{x_i = r_{0,i}\}$. We use these initial states to specify the reset maps of all transitions as follows: any transition with target location l_i ($i = 1,2,3$) resets the state of l_i to $\{x_i = r_{0,i}\}$ with probability one. The guards of all four a-transitions are equal to the whole valuation space of the corresponding origin locations. The spontaneous transitions that have label $\lambda[y < 0]$ in Figure 9.2 are spontaneous transitions whose rates are piecewise constant: $\lambda[y < 0]$, which is shorthand notation, means that for states with output value smaller than zero, the rate is equal to the constant λ and for all other states the rate is equal to zero. The other two spontaneous transitions have constant rates μ and 2μ.

The locations l_1,l_2 and l_3 have linear time invariant state/output dynamics (see Figure 9.2). The dynamics of locations l_2 and l_3 are related as follows: $A_2 = TA_3T^{-1}$ and $C_2 = C_3T^{-1}$. In other words the dynamics of l_2 and l_3 can be transformed into each other via a state space transformation. Also, the initial states of locations l_2 and l_3 are related as: $r_{0,2} = Tr_{0,3}$.

We will now show that the equivalence relation

$$\mathcal{R} = \{\{(l_1, val)\}|val \in vs(l_1)\} \cup \{\{(l_2, x_2 = r_2), (l_3, x_3 = r_3)\}|r_2 = Tr_3\}$$

is a bisimulation for the CPDP X. First, the quotient space induced by $\mathcal{R}$ is Borel isomorphic to $(l_2, vs(l_2)) \cup (l_3, vs(l_3))$, where $(l_i, vs(l_i)) := \{(l_i, val)|val \in vs(l_i)\}$, which is clearly a Borel space and therefore the quotient space itself is a Borel space. Thus, $\mathcal{R}$ is a measurable relation. Now, we check the four conditions of Definition 9.13.

1. Follows directly from the fact that the state/output dynamics of location l_2 is a state space transformation of the state/output dynamics of location l_3 via matrix T.

2. Idem.

3. For both locations l_2 and l_3 the total rate equals $2\mu+\lambda$ for states where $y < 0$ and equals 2μ for states with $y \geq 0$. Thus, the total jump rates are the same. The spontaneous transitions to locations l_2 and l_3 have equivalent reset maps because $(l_2, \{x_2 = r_{0,2}\})$ and $(l_2, \{x_2 = r_{0,2}\})$ are bisimilar states. From this and from the fact that both λ transitions have the same reset map, we get that the second part of item 3 of Definition 9.13 holds.

4. Follows from the following two observations.

   - For all $val_2 \in vs(l_2)$ and all $val_3 \in vs(l_3)$ both the transitions $((l_2, val_2), Id_{(l_1,\{x_1=r_{0,1}\})})$ and $((l_3, val_3), Id_{(l_1,\{x_1=r_{0,1}\})})$ are present.
   - For all $val_2 \in vs(l_2)$ and all $val_3 \in vs(l_3)$ the transitions $((l_2, val_2), Id_{(l_3,\{x_1=r_{0,3}\})})$, $((l_3, val_3), Id_{(l_2,\{x_1=r_{0,2}\})})$ and $((l_3, val_3), Id_{(l_3,\{x_1=r_{0,3}\})})$ are present. Then, because the probability measures $Id_{(l_2,\{x_1=r_{0,2}\})}$ and $Id_{(l_3,\{x_1=r_{0,3}\})}$ are equivalent, we have that for each a-transition from a state with location l_2 there is a matching a-transition at its bisimilar state in location l_3 and vice versa.

In Section 9.3 we will see how this CPDP can be state reduced via bisimulation by using the algorithm we present in that section.

In this example we have seen that if the state/output dynamics of two locations can be transformed into each other via classical state space transformation, then this naturally leads to an equivalence relation on the hybrid state space such that item 1 and item 2 of Definition 9.13 hold. Later we will see that classical state space reduction also naturally leads to an equivalence relation such that item 1 and item 2 of Definition 9.13 are satisfied.

9.1.1 Substitutivity of bisimulation for CPDPs

An important property of bisimulation in the context of complex systems is that bisimilarity is substitutive. For CPDPs and composition defined by $|_A^P|$, this means that if for CPDP X the states $\xi_{X,1}$ and $\xi_{X,2}$ are bisimilar, then for any state ξ_Y of any CPDP Y we should have that the product states $(\xi_{X,1}, \xi_Y)$ and $(\xi_{X,2}, \xi_Y)$ are also bisimilar for $X|_A^P|Y$. We call this the *substitutivity property* of bisimulation with respect to $|_A^P|$. Before we prove that this is the case, we first need to prove that the trivial equivalence relation on the product space $E_1 \times E_2$ that is induced by a measurable equivalence relation on the space E_1, is again measurable.

Lemma 9.17. *If $\mathcal{R}$ is a measurable equivalence relation on Borel space E_1, then for all Borel spaces E_2*

$$\hat{\mathcal{R}} := \{((\xi_1, \xi_2), (\xi_1', \xi_2)) | (\xi_1, \xi_1') \in \mathcal{R}, \xi_2 \in E_2\}$$

is a measurable equivalence relation on the product Borel space $E_1 \times E_2$.

Proof. There exists a measurable $\psi : [0,1] \times E_1/_{\mathcal{R}} \to E_1$, such that $\psi([0,1] \times [\xi]) = \{\tilde{\xi} \in E_1 | [\tilde{\xi}] = [\xi]\}$. (This existence is proven in Theorem 3 of [SvdS05c], where Proposition 2.8 and Theorem 8.1 of [Par67] are used). We prove that $(E_1 \times E_2)/_{\mathcal{R}'} = E_1/_{\mathcal{R}} \times E_2$, which is indeed a Borel space.

Take $B \in \mathcal{B}^*(E_1 \times E_2)$ (i.e., B is Borel in $E_1 \times E_2$ and for any ξ_2 we have: if $(\xi_1, \xi_2) \in B$ and $[\xi_1] = [\tilde{\xi_1}]$ then $(\tilde{\xi_1}, \xi_2) \in B$). Now there exist Borel sets $B_i^{E_1}$ and $B_i^{E_2}$ such that

$$B = \cup_{i=1}^{\infty} B_i^{E_1} \times B_i^{E_2}.$$

Because ψ is measurable, we have that for all i that $\psi^{-1}(B_i^{E_1}) \in \mathcal{B}([0,1] \times E_1/_{\mathcal{R}})$. This means that there exist Borel sets $B_{i,j}^{E_1/_{\mathcal{R}}}$ and $B_{i,j}^{[0,1]}$, such that

$$\cup_{i=1}^{\infty} \psi^{-1}(B_i^{E_1}) \times B_i^{E_2} = \cup_{i,j=1}^{\infty} B_{i,j}^{[0,1]} \times B_{i,j}^{E_1/_{\mathcal{R}}} \times B_i^{E_2}$$

Because we have that if $(\xi_1, \xi_2) \in B$ and $[\xi_1] = [\tilde{\xi_1}]$ then $(\tilde{\xi_1}, \xi_2) \in B$, we can also write

$$\cup_{i=1}^{\infty} \psi^{-1}(B_i^{E_1}) \times B_i^{E_2} = \cup_{i,j=1}^{\infty} [0,1] \times B_{i,j}^{E_1/_{\mathcal{R}}} \times B_i^{E_2},$$

from which we can see that $\mathcal{R}'$ maps B to $\cup_{i,j=1}^{\infty} B_{i,j}^{E_1/_{\mathcal{R}}} \times B_i^{E_2}$, which is a Borel set in $E_1/_{\mathcal{R}} \times E_2$ and therefore $(E_1 \times E_2)/_{\mathcal{R}'}$ is a Borel space. □

Now we can prove the substitutivity property of bisimulation for composition. The result holds for both NTSs and CPDPs.

Theorem 9.18. *Let X be an NTS with state space E_1. Let $\mathcal{R}$ be a bisimulation for X. Then for all NTSs Y, with state space E_2, and all A and P, $\hat{\mathcal{R}}$, as defined in Lemma 9.17, is a bisimulation for NTS $X|_A^P|Y$. Furthermore, the result also holds if X and Y are CPDPs.*

Proof. We first prove the result for the NTS case. $\hat{\mathcal{R}}$ is a measurable relation according to Lemma 9.17. Let $(\xi_1, \xi_1') \in \mathcal{R}$. We prove that for any $\xi_2 \in E_2$, $((\xi_1, \xi_2), (\xi_1', \xi_2)) \in \hat{\mathcal{R}}$.

The output at (ξ_1, ξ_2) equals the union of outputs of ξ_1 and ξ_2, i.e., $O(\xi_1, \xi_2) = O(\xi_1) \cup O(\xi_2)$ which equals because of bisimulation $\mathcal{R}$ $O(\xi_1') \cup O(\xi_2) = O(\xi_1', \xi_2)$. Thus, item 1 of Definition 9.10 is satisfied,

Suppose for some $\sigma \in \Sigma \cup \bar{\Sigma}$ we have $(\xi_1, \xi_2) \xrightarrow{\sigma} m_1 \times m_2$. Then, this transition reflects transitions $\xi_1 \xrightarrow{\sigma_1} m_1$ and $\xi_2 \xrightarrow{\sigma_2} m_2$ of the components. Depending on the kind of transition, we could have $\sigma_1 = \sigma$ and $\sigma_2 = \bar{\sigma}$, or $\sigma_1 = \sigma$ and $\sigma_2 = *$, where $*$ means 'no transition for this component' and consequently for $\sigma_2 = *$ we have $m_2 = Id$, etc. Then, because of bisimulation $\mathcal{R}$, we also have $\xi_1' \xrightarrow{\sigma_1} m_1'$, with m_1 and m_1' equivalent. Consequently, $(\xi_1', \xi_2) \xrightarrow{\sigma} m_1' \times m_2$. $m_1(A_1) \times m_2(A_2) = m_1'(A_1) \times m_2(A_2)$ for all $A_1 \in \mathcal{B}^*(E_1)$ and all $A_2 \in \mathcal{B}(E_2)$. The σ-algebra of $\mathcal{B}^*(E_1 \times E_2)$ is generated by all A_1 and A_2 of this form and then we get that $m_1 \times m_2$ and $m_1' \times m_2$ agree on all sets in $\mathcal{B}^*(E_1 \times E_2)$, from which we conclude that item 2 of Definition 9.10 is satisfied.

Now we prove the CPDP case. Satisfaction of items 1 and 4 of Definition 9.13 follow directly from the NTS case. Item 2 of Definition 9.13 is satisfied because $(\phi(t, (\xi_1, \xi_2)), \phi(t, (\xi_1', \xi_2))) = ((\phi(t, \xi_1), \phi(t, \xi_2)), (\phi(t, \xi_1'), \phi(t, \xi_2))) \in \hat{\mathcal{R}}$ because $(\phi(t, \xi_1), \phi(t, \xi_1')) \in \mathcal{R}$. The measures from item 3 of Definition 9.13 are, for sets $A_1 \times A_2$, equal to

$$\frac{\lambda(\xi_1)}{\lambda(\xi_1) + \lambda(\xi_2)} Q_1(\xi_1)(A_1) + \frac{\lambda(\xi_2)}{\lambda(\xi_1) + \lambda(\xi_2)} Q_2(\xi_1)(A_2)$$

and

$$\frac{\lambda(\xi_1')}{\lambda(\xi_1') + \lambda(\xi_2)} Q_1'(\xi_1')(A_1) + \frac{\lambda(\xi_2)}{\lambda(\xi_1') + \lambda(\xi_2)} Q_2(\xi_1)(A_2)$$

respectively, where Q_1, Q_1' and Q_2 are the CFSJS transition measures at states ξ_1, ξ_1' and ξ_2. Because of bisimulation $\mathcal{R}$, these two measures agree on all sets $A_1 \times A_2$ with $A_1 \in \mathcal{B}^*(E_1)$ and $A_2 \in \mathcal{B}(E_2)$ and therefore, as described above, agree on all sets from $\mathcal{B}^*(E_1 \times E_2)$, from which we conclude that item 3 of Definition 9.13 is satisfied. □

To use bisimulation as a compositional model reduction technique, it is convenient to define when two CPDPs are bisimilar to each other. We first define the *union CPDP* of two CPDPs. With this concept of union CPDP the definition of bisimilarity of two CPDPs follows directly from Definition 9.13.

Definition 9.19. Let $X = (L_X, V_X, \nu_X, W, \omega_X, F_X, G_X, \Sigma, \mathcal{A}_X, \mathcal{P}_X, \mathcal{S}_X)$ and $Y = (L_Y, V_Y, \nu_Y, W, \omega_Y, F_Y, G_Y, \Sigma, \mathcal{A}_Y, \mathcal{P}_Y, \mathcal{S}_Y)$ be two CPDPs with state spaces E_X and E_Y. We define the *union CPDP* $X \cup Y$ as $(L_X \cup L_Y, V_X \cup V_Y, \nu, W, \omega, F, G, \Sigma, \mathcal{A}_X \cup \mathcal{A}_Y, \mathcal{P}_X \cup \mathcal{P}_Y, \mathcal{S}_X \cup \mathcal{S}_Y)$. Here $\nu(l)$ is defined as $\nu_X(l)$ if $l \in L_X$ and as $\nu_Y(l)$ if $l \in \nu_Y(l)$, $\omega(\xi)$ is defined as $\omega_X(\xi)$ if $\xi \in E_X$ and as $\omega_Y(\xi)$ if $\xi \in E_Y$. F_X and G_X are defined likewise. The reset maps of the transitions are here extended to the space $E_X \cup E_Y$, where a reset map from a transition from X assigns probability zero to E_Y, etc. CPDP $X \cup Y$ behaves as X if the initial state lies in E_X and behaves as Y if the initial state lies in E_Y.

Definition 9.20. CPDPs X and Y with state spaces E_X and E_Y are called *bisimilar* if there exists a bisimulation $\mathcal{R}$ for the CPDP $X \cup Y$ such that for all $A \in \mathcal{B}^*(E_X \cup E_Y)$ we have $A \cap E_X \neq \emptyset$ and $A \cap E_Y \neq \emptyset$.

Corollary 9.21. *With Definition 9.20, we can directly derive from Theorem 9.18 that if X and X' are bisimilar CPDPs (or NTSs) then for all CPDPs (or NTSs) Y we have that the CPDPs (or NTSs) $X|_A^P|Y$ and $X'|_A^P|Y$ are also bisimilar.*

This corollary shows how bisimulation can be used for compositional model reduction: each component in the composition may be replaced by a (reduced) bisimilar component without changing the composed system up to bisimilarity.

9.2 Bisimulation for internally scheduled CPDPs

We define bisimulation for internally scheduled CPDPs.

Definition 9.22. Let $X = (L, V, \nu, W, \omega, F, G, \Sigma, \mathcal{A}, \mathcal{P} = \emptyset, \mathcal{S})$ be a CPDP with state space E and flow map ϕ. Let S^i be an internal scheduler for X. A measurable relation $\mathcal{R}$ on E is called a *bisimulation* for (X, S^i) if $(\xi_1, \xi_2) \in \mathcal{R}$ implies that conditions 1,2 and 3 of Definition 9.13 are satisfied and for all $\sigma \in \Sigma \cup \bar{\Sigma}$ we have that if $\xi_1 \xrightarrow{\sigma}$ then $\xi_2 \xrightarrow{\sigma}$, while

$$\sum_{\alpha \in \mathcal{T}_{\xi_1 \xrightarrow{\sigma}}} S^i(\alpha)(\xi_1) rmap(\alpha)(\xi_1) \quad \text{and} \quad \sum_{\alpha \in \mathcal{T}_{\xi_2 \xrightarrow{\sigma}}} S^i(\alpha)(\xi_2) rmap(\alpha)(\xi_2) \tag{9.1}$$

are equivalent measures, where $\mathcal{T}_{\xi \xrightarrow{\sigma}}$ denotes the set of all ξ-enabled σ-transitions.

The last condition in Definition 9.22 assures that for each $\sigma \in \Sigma \cup \bar{\Sigma}$, the combined action of internal scheduler and all σ-transitions is equivalent for bisimilar states. In Chapter 8 we saw that the semantics of an internally scheduled CPDP can be captured as a CFSJS together with an NTS. Bisimulation for internally scheduled CPDPs is reflected on the semantical level as follows.

Lemma 9.23. *$\mathcal{R}$ is a bisimulation for a CPDP (X, S^i) if and only if $\mathcal{R}$ is a bisimulation for both the CFSJS and the NTS of (X, S^i).*

Proof. The measure $\sum_{\alpha \in \mathcal{T}_{\xi \xrightarrow{\sigma}}} S^i(\alpha)(\xi) rmap(\alpha)(\xi)$, as defined in Definition 9.22 for $\xi \in \{\xi_1, \xi_2\}$, is exactly the measure for the σ-transition of the NTS of (X, S^i) at state ξ. The result now follows directly from this observation, Definition 9.10 and Lemma 9.15. □

Example 9.24. Consider the CPDP X of Figure 9.2 with bisimilation relation $\mathcal{R}$ of Example 9.16. Remember that the states $(l_3, \{x_3 = r_3\})$ and $(l_2, \{x_2 = Tr_3\})$, where T is the state space transformation matrix, are contained in $\mathcal{R}$. Let S^i be an internal scheduler for X. S^i schedules the a-transitions at locations l_2 and l_3. When is $\mathcal{R}$ a bisimulation for (X, S^i)? The answer is: if for all appropriate dimensioned r_3 we have $S^i(l_3, \{x_3 = r_3\})(l_3 \xrightarrow{a} l_1) = S^i(l_2, \{x_2 = Tr_3\})(l_2 \xrightarrow{a} l_1)$. Note that this means that the scheduling value of $l_2 \xrightarrow{a} l_3$ equals the sum of the scheduling values of $l_3 \xrightarrow{a} l_2$ and $l_3 \xrightarrow{a} l_3$. This satisfies the bisimulation conditions because the reset probability measures used by the three transitions are all equivalent with respect to $\mathcal{R}$.

Also for internally scheduled CPDPs we define bisimilarity. For this purpose we first define the union of two internally scheduled CPDPs.

Definition 9.25. The union of the two internally scheduled CPDPs (X, S^i_X) and (Y, S^i_Y) is defined as follows. $(X, S^i_X) \cup (Y, S^i_Y) := (X \cup Y, S^i)$, where $S^i(\xi)$ equals $S^i_X(\xi)$ if $\xi \in E_X$ and equals $S^i_Y(\xi)$ if $\xi \in E_Y$.

Definition 9.26. The internally scheduled CPDPs (X, S^i_X) and (Y, S^i_Y) with state spaces E_X and E_Y are called *bisimilar* if there exists a bisimulation $\mathcal{R}$ for the CPDP $(X, S^i_X) \cup (Y, S^i_Y)$ such that for all saturated Borel sets $A \in \mathcal{B}^*(E_X \cup E_Y)$ we have $A \cap E_X \neq \emptyset$ and $A \cap E_Y \neq \emptyset$.

The following theorem shows the relation between bisimilar CPDPs and equivalence of stochastic output behavior.

Theorem 9.27. *Let the internally scheduled CPDPs (X, S^i_X) and (Y, S^i_Y) be bisimilar. Let $\mathcal{R}$ be a bisimulation for $X \cup Y$. Let S^e_X and S^e_Y be external schedulers for X and Y such that for all $\sigma \in \Sigma$ we have $S^e_X(\xi_1)(\sigma) = S^e_Y(\xi_2)(\sigma)$ if $(\xi_1, \xi_2) \in \mathcal{R}$. Then, the quotient TMS of (X, S_X) under $\mathcal{R}$ is isomorphic equivalent to the quotient TMS of (Y, S_Y) under $\mathcal{R}$, where $S_X = (S^i_X, S^e_X)$ and $S_Y = (S^i_Y, S^e_Y)$.*

Proof. Let $(\xi_1, \xi_2) \in \mathcal{R}$. The FTS reset measure at state ξ_1 is equal to $\sum_{\sigma \in \Sigma} S^e_X(\xi_1)(\sigma) m_\sigma(\xi_1)$, where $m_\sigma(\xi_1)$ is the reset measure for the σ-transition of the NTS of (X, S^i_x) at state ξ_1, and the FTS reset measure at state ξ_2 is equal to $\sum_{\sigma \in \Sigma} S^e_Y(\xi_2)(\sigma) m_\sigma(\xi_2)$. Because of bisimulation $\mathcal{R}$ and Lemma 9.23 $m_\sigma(\xi_1)$ and $m_\sigma(\xi_2)$ are equivalent measures and because we have $S^e_X(\xi_1)(\sigma) = S^e_Y(\xi_2)(\sigma)$ we get that the two FTS reset measures at ξ_1 and ξ_2 are equivalent. Therefore, $\mathcal{R}$ is a bisimulation for the FTS of $(X, S^i_X) \cup (Y, S^i_Y)$ with external scheduler S^e_X for the states of X and external scheduler S^e_Y for the states of Y. According to Lemma 9.9 we now have that $\mathcal{R}$ is a bisimulation for the TMS of $X \cup Y$. It can now easily be seen that the quotient TMS of (X, S_X) and the quotient TMS of (Y, S_Y) are isomorphic equivalent under the isomorphism that maps each equivalence class of (X, S_X), which is a single state in the quotient TMS of (X, S_X), to the corresponding equivalence class of (Y, S_Y), which is a single state in the quotient TMS of (Y, S_Y). $\square$

Corollary 9.28. *Bisimilar scheduled CPDPs, i.e., bisimilar internally scheduled CPDPs with external schedulers as in Theorem 9.27, generate the same output executions (with the same probabilities). This means that if we want to analyze the stochastic output behavior of a scheduled CPDP, then we can as well analyze the stochastic output behavior of a (reduced) bisimilar scheduled CPDP.*

9.2.1 Substitutivity of bisimulation for scheduled CPDPs

We saw that a bisimulation relation for one CPDP component in a composition naturally leads to a bisimulation relation for the composed CPDP. In the context of scheduled CPDPs there is another factor that we should take into account, the priority function. The priority function is a kind of high level scheduler that chooses probabilistically which component may execute a transition in cases where both components have transitions enabled. Bisimulation for scheduled CPDPs satisfies the substitutivity property only if the priority function treats bisimilar states in the same way. It is clear that when a priority function assigns different probabilities to the components for states that are bisimilar, then the behavior at these states will be different. We call a priority function that treats bisimilar states (with respect to $\mathcal{R}$) in the same way *$\mathcal{R}$-respecting*. This notion is defined as follows.

Definition 9.29. Let *pri* be a priority function for $X|_A^P|Y$ and let $\mathcal{R}$ be a bisimulation for X. *pri* is called *$\mathcal{R}$-respecting* if for all $\sigma \in \Sigma \cup \bar{\Sigma}$ we have: if $(\xi_X, \xi'_X) \in \mathcal{R}$ and for some $\xi_Y \in E_Y$ (ξ_X, ξ_Y, σ) and (ξ'_X, ξ_Y, σ) are conflict pairs, then $pri(\xi_X, \xi_Y, \sigma)(X) = pri(\xi'_X, \xi_Y, \sigma)(X)$.

We can now state the substitutivity result for bisimulation for internally scheduled CPDPs.

Theorem 9.30. *Let (X, S_X^i) and (Y, S_Y^i) be internally scheduled CPDPs. Let $\mathcal{R}$ be a bisimulation for (X, S_X^i). Let pri be an $\mathcal{R}$-respecting priority function for $X|_A^P|Y$. Then,*

$$\hat{\mathcal{R}} := \{((\xi_1, \xi_2), (\xi'_1, \xi_2)) | (\xi_1, \xi'_1) \in \mathcal{R}, \xi_2 \in E_Y\}$$

is a bisimulation for $(X|_A^P|Y, (S_X^i, S_Y^i, pri))$.

Proof. Let $(\xi_1, \xi_2), (\xi'_1, \xi_2)) \in \hat{\mathcal{R}}$. We first show that for all $\sigma \in \Sigma \cup \bar{\Sigma}$ the reset measures of the NTS σ-transitions at states (ξ_1, ξ_2) and (ξ'_1, ξ_2) are equivalent. Assume that $(\xi_1, \xi_2, \sigma) \in CP$, i.e., is a conflict pair. Then we have NTS transitions $\xi_1 \xrightarrow{\sigma} m_1$ and $\xi_2 \xrightarrow{\sigma} m_2$, which bring forth the NTS σ-transition $(\xi_1, \xi_2) \xrightarrow{\sigma} pri(\xi_1, \xi_2, \sigma)(X)m_1 + pri(\xi_1, \xi_2, \sigma)(Y)m_2$ in the composition $(X|_A^P|Y, (S_X^i, S_Y^i, pri))$. Because of bisimulation $\mathcal{R}$, we have $\xi'_1 \xrightarrow{\sigma} m'_1$, with m'_1 equivalent to m_1. This means that we get the NTS σ-transition

$$(\xi'_1, \xi_2) \xrightarrow{\sigma} pri(\xi'_1, \xi_2, \sigma)(X)m'_1 + pri(\xi'_1, \xi_2, \sigma)(Y)m_2$$

in the composition $(X|_A^P|Y, (S_X^i, S_Y^i, pri))$. $pri(\xi'_1, \xi_2, \sigma)(X)m'_1 + pri(\xi'_1, \xi_2, \sigma)(Y)m_2$ is equivalent to $pri(\xi_1, \xi_2, \sigma)(X)m_1 + pri(\xi_1, \xi_2, \sigma)(Y)m_2$ because m'_1 is equivalent to m_1 and *pri* is $\mathcal{R}$-respecting. Thus, we proved the NTS part for conflict pairs. The NTS part for non-conflict pairs, the CFSJS part, the output part and the flow map part can all be found in the proof of Theorem 9.18. □

In terms of bisimilar CPDPs, this result can be stated as in the following corollary.

Corollary 9.31. *Let (X, S^i_X), $(X', S^i_{X'})$ and (Y, S^i_Y) be internally scheduled CPDPs. Let (X, S^i_X) and $(X', S^i_{X'})$ be bisimilar, which means that there exists a bisimulation $\mathcal{R}$ for $(X, S^i_X) \cup (X', S^i_{X'})$ such that for all saturated Borel sets $A \in \mathcal{B}^*(E_X \cup E_{X'})$ we have $A \cap E_X \neq \emptyset$ and $A \cap E_{X'} \neq \emptyset$. Let pri be an $\mathcal{R}$-respecting priority function for $(X \cup X')|^P_A|Y$. Then, consequently, pri is an $\mathcal{R}$-respecting priority function for both $X|^P_A|Y$ and $X'|^P_A|Y$ and we have that $(X|^P_A|Y, (S^i_X, S^i_Y, pri))$ and $(X|^P_A|Y, (S^i_{X'}, S^i_Y, pri))$ are bisimilar.*

9.3 Bisimulation algorithm

In this section we give an algorithm to automatically determine bisimulation relations on the state space of a CPDP. First we give the bisimulation algorithm for internally scheduled CPDPs. After that, we show how the algorithm can be adjusted such that it computes a bisimulation for non-scheduled CPDPs. After this we also give another version of the algorithm, again for both internally scheduled and non-scheduled CPDPs, which is ‘improved’ in the sense that it might find coarser bisimulation partitions. The first algorithm can therefore refine too much with respect to the second algorithm, however, executing this algorithm is less elaborate than executing the second algorithm and might therefore in some cases be preferred.

The algorithms will only find bisimulations with the following property: two states can be in the bisimilation only if the corresponding locations of these states are bisimilar, where we define bisimilarity on the set of locations of a CPDP as follows.

Definition 9.32. We call two locations l_1 and l_2 of a CPDP X bisimilar if there exists a bisimulation relation $\mathcal{R}$ for X such that

1. $\{val_1 | \exists val_2 \in vs(l_2)$ such that $((l_1, val_1), (l_2, val_2) \in \mathcal{R}\}$ equals $vs(l_1)$, and

2. $\{val_2 | \exists val_1 \in vs(l_1)$ such that $((l_1, val_1), (l_2, val_2)) \in \mathcal{R}\}$ equals $vs(l_2)$.

We call two locations l_1 and l_2 of an internally scheduled CPDP (X, S^i_X) bisimilar if there exists a bisimulation relation $\mathcal{R}$ for (X, S^i_X) such that the above two conditions are satisfied.

In general this will not be the maximal bisimulation because it could for example be that two states are bisimilar while their corresponding locations are not bisimilar. We will see that this restriction, i.e., only looking for bisimilar locations, makes it possible to split up the process of finding a bisimulation in different parts: one for the continuous dynamics, one for the reset maps and one for the transitions and their guards.

Assumption 9.33. We assume throughout this section that guards and jump-rates can be defined on the output space. In other words, we assume that for a CPDP, with state space E, output space E_O and output mapping O,

1. for each active transition α with guard $\tilde{G}_\alpha$, there exists a $G_\alpha \subset E_O$, such that
$$\tilde{G}_\alpha = \{\xi \in E | O(\xi) \in G_\alpha\},$$
2. for each spontaneous transition α with jump rate $\tilde{\lambda}_\alpha$, there exists a $\lambda : E_O \to \mathbb{R}_+$ such that for all states ξ
$$\tilde{\lambda}(\xi) = \lambda(O(\xi)).$$

We also assume that all active, passive and spontaneous transitions we have that the reset maps of these transitions assign to each state the same reset measure. For the bisimulation algorithms that we will present, we restrict ourselves to CPDPs that satisfy these assumptions. From now on guards will be regarded as subsets of the output space and jump rates will be regarded as functions defined on the output space.

9.3.1 Partitioning the output space

In order to present the first algorithm, we need to define $P_s(l, \sigma, C_r)$ and $P_j(l, C_r)$, where $\sigma \in \Sigma \cup \bar{\Sigma}$. $P_s(l, \sigma, C_r)$ partitions the output space such that the sum of scheduling values of all σ-transitions with reset measure in C_r, where C_r is a set of reset measures, is the same at outputs y and y' if these outputs are in the same class. $P_j(l, C_r)$ partitions the output space such that for two outputs in the same class the sum of jump rates of all transitions with reset measure in C_r is the same.

First we define P_s for a single transition. The internal scheduler is denoted by S^i.
$$P_s(\alpha) = \bigcup_{p \in \mathbb{R}_+} \{(p, \{y \in G_\alpha | S^i(\alpha, y) = p\})\} - \bigcup_{p \in \mathbb{R}_+} \{(p, \emptyset)\}.$$
Note that for all scheduling values p the empty set is removed from P_s since this is not an element of a partition. In fact, P_s does not make a partition of the guard, but makes a partition of the guard where each class has a value p assigned to it.

We now define P_s for a set of transitions, but to do that we first have to define the scheduler S^i for a set of transitions:
$$S^i(\{\alpha_1, \alpha_2, \cdots, \alpha_n\}, y) = S^i(\alpha_1, y) + \cdots + S^i(\alpha_n, y).$$
Now we define P_s for a set of transitions $A = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$. $G_A = \cup_{\alpha \in A} G_\alpha$ denotes the union of guards.
$$P_s(A) = \bigcup_{p \in \mathbb{R}_+} \{(p, \{y \in G_A | S^i(A, y) = p\})\} - \bigcup_{p \in \mathbb{R}_+} \{(p, \emptyset)\}.$$
Now we define for all $\sigma \in \Sigma$ the partitioning function P_s in the form that we use in the algorithm.
$$P_s(l, \sigma, \{r_1, \cdots, r_n\}) := P_s(\{\alpha \in \mathcal{A}_{l \xrightarrow{\sigma}} | R_\alpha \in \{r_1, \cdots, r_n\}\}).$$

Let $C_r = \{r_1, \cdots, r_n\}$. Then $P_s(l, \sigma, C_r) = \cup_{i \in I}\{(p_i, G_i)\}$ and $\cup_{i \in I}\{G_i\}$ forms a partition of G_{σ, C_r}, where G_{σ, C_r} is the set of all outputs y where a σ-transition with (constant) reset map in C_r is enabled. $y \in G_i$ (for some $i \in I$) is then interpreted as: 'the probability that for action σ the internal scheduler chooses at output y a transition with reset map in C_r' equals p_i. In the algorithm, $P_s(l, \sigma, C_r)$, with C_r an equivalence class of reset measures, will be used to discriminate states as non-bisimilar.

In the same way we can define $P_s(l, \bar{\sigma}, C_r)$, with $\bar{\sigma} \in \bar{\Sigma}$ and $C_r = \{r_1, \cdots, r_n\}$, for passive transitions:

$$P_s(l, \bar{\sigma}, C_r) := \bigcup_{p \in \mathbb{R}_+} \{(p, \{y | S(\{\alpha \in \mathcal{P}_{l \xrightarrow{\bar{\sigma}}} | R_\alpha \in C_r\}, y) = p\})\} - \bigcup_{p \in \mathbb{R}_+} \{(p, \emptyset)\}.$$

Using the same idea, P_j makes a partition of the output space based on the jump-rates: for $C_r = \{r_1, \cdots, r_n\}$ we define

$$P_j(l, C_r) := \bigcup_{\lambda \in \mathbb{R}_+} \{(\lambda, \{y | \sum_{\alpha \in \mathcal{S}_{(l,y) \to C_r}} \lambda_\alpha(y) = \lambda\})\} - \bigcup_{\lambda \in \mathbb{R}_+} \{(\lambda, \emptyset)\},$$

where $\mathcal{S}_{(l,y) \to C_r}$ is the set of all spontaneous transitions leaving l that assign a reset measure from C_r to state y.

9.3.2 Bisimulation algorithm

To state the algorithms we need the following definition.

Definition 9.34. A measurable relation $\mathcal{R}$ on the state space E of a CPDP is called a *bisimulation with respect to the continuous dynamics* if it satisfies conditions 1 and 2 of Definition 9.13. Two states ξ and ξ' are called bisimilar with respect to the continuous dynamics if there exists such a relation $\mathcal{R}$ such that $(\xi, \xi') \in \mathcal{R}$.

Now, we can present the first algorithm that computes a bisimulation relation for a CPDP.

Algorithm 9.35. Let (X, S^i) be an internally scheduled CPDP with state space E that satisfies Assumption 9.33. Let L be the set of locations of X and let R be the set of all reset measures that are used by some transition of X. For all $k \in \mathcal{N}$, $\mathcal{R}_l^k$ and $\mathcal{R}_r^k$ are calculated as follows.

Step 1 Determine $\mathcal{R}_l^0 \subset L \times L$ such that $(l, l') \in \mathcal{R}_l^0$ if and only for all ξ with $loc(\xi) = l$ there exists a ξ' with $loc(\xi') = l'$ such that ξ and ξ' are bisimilar with respect to the continuous dynamics. Let, for each $(l, l') \in \mathcal{R}_l^0$, $\mathcal{R}_{l,l'}$ be the maximal bisimulation concerning the continuous dynamics, i.e., let $\mathcal{R}_{l,l'}$ be equal to $\{(\xi, \xi') | \xi, \xi' \in (l, vs(l)) \cup (l', vs(l'))$ and ξ and ξ' are bisimilar with respect to the continuous dynamics$\}$.

Step 2 Determine $\mathcal{R}_r^0 \subset R \times R$ such that $(r, r') \in \mathcal{R}_r^0$ if and only if r and r' are equivalent measures with respect to $\mathcal{R}_{l_r, l_{r'}}$, where l and l' are the target locations of r and r' respectively.

Step 3 Determine inductively for each $k \in \mathbb{N}$

$$\mathcal{R}_l^{k+1} = \mathcal{R}_l^k \cap \{(l, l') | (\forall \sigma \in (\Sigma \cup \bar{\Sigma}))(\forall C_r \in \mathcal{R}_r^k)$$

$$(P_s(l, \sigma, C_r) = P_s(l', \sigma, C_r) \wedge P_j(l, C_r) = P_j(l', C_r))\}$$

$$\mathcal{R}_r^{k+1} = \mathcal{R}_r^k \cap \{(r, r') | (\forall C_l \in \mathcal{R}_l^{k+1})(r(C_l) = r'(C_l))\},$$

where $r(C_l)$ equals one if the target location of r lies in C_l and equals zero otherwise.

We explain what happens in the three steps of this algorithm. In step 1, $\mathcal{R}_l^0$ partitions the location space such that l and l' are in the same class if their continuous dynamics are bisimilar with respect to the continuous dynamics. This gives a first partition on the locations space which is probably too coarse because it only concerns the continuous dynamics and does not discriminate yet on the transition dynamics and the jump rates. Therefore, this partition needs to be refined. These refinement steps are done in step 3. Per pair $(l, l') \in \mathcal{R}_l^0$, $\mathcal{R}_{l,l'}$ is determined as the maximal bisimulation with respect to the continuous dynamics on the part of the state space that has location l or l'.

In step 2, $\mathcal{R}_r^0$ partitions the set of reset measures such that equivalent reset measures with respect to $\mathcal{R}_{l,l'}$, from step 1, fall in the same class. Again, this partition needs to be refined in step 3 because $\mathcal{R}_{l,l'}$ is too coarse since it only concerns the continuous dynamics.

In step 3, $\mathcal{R}_l^0$ and $\mathcal{R}_r^0$ are refined into $\mathcal{R}_l^1$ and $\mathcal{R}_r^1$, which are then refined into $\mathcal{R}_l^2$ and $\mathcal{R}_r^2$, etc. Refinement goes on until a fixed point is achieved. We will see in the proof of Theorem 9.36, that for bisimilar locations l and l' we have under certain conditions that $P_s(l, \sigma, C_r) = P_s(l', \sigma, C_r)$ and $P_j(l, C_r) = P_j(l', C_r)$ for all $\sigma \in \Sigma \cup \bar{\Sigma}$ and for all equivalence classes C_r of the partition $\mathcal{R}_r$ of equivalent measures. (We come back on this 'under certain conditions'). In the refinement step for $\mathcal{R}_l$, $\mathcal{R}_l^k$ is refined into $\mathcal{R}_l^{k+1}$, such that we indeed have $P_s(l, \sigma, C_r) = P_s(l', \sigma, C_r)$ and $P_j(l, C_r) = P_j(l', C_r)$, but only for equivalence classes C_r from $\mathcal{R}_r^k$ and not from $\mathcal{R}_r$ (which do not know yet at this point of the algorithm). Because $\mathcal{R}_r^k$ might be too coarse, we might need another refinement step for $\mathcal{R}_l^k$. When we find in some step that $\mathcal{R}_l^{k+1}$ equals $\mathcal{R}_l^k$, then we found the fixed point and we know that further refinement will not happen in all following steps of the algorithm. In Theorem 9.36 we prove that this fixed point forms a bisimulation on the set of locations.

Once $\mathcal{R}_l^k$ is refined into $\mathcal{R}_l^{k+1}$, we might have found that l and l' are not bisimilar (while we did not know that yet in the previous step). This means that we then also know that reset measures with target locations l and l' are not equivalent. With this knowledge $\mathcal{R}_r^k$ is refined into $\mathcal{R}_r^{k+1}$.

The following theorem states that if Algorithm 9.35 reaches a fixed point, then this fixed point is a bisimulation on the set of locations.

Theorem 9.36. *If, in Algorithm 9.35, for some k we have $\mathcal{R}_l^{k+1} = \mathcal{R}_l^k$ (i.e., $\mathcal{R}_l^k$ is the fixed point), and consequently $\mathcal{R}_r^{k+1} = \mathcal{R}_r^k$, then $\mathcal{R}_l^k$ is a bisimulation on the set of locations (i.e., locations l and l' are bisimilar if $(l, l') \in \mathcal{R}_l^k$).*

Proof. We use the following notation: if $\mathcal{R}$ is an equivalence relation on a set E, then $Part(\mathcal{R})$ denotes the corresponding partition, i.e., $Part(\mathcal{R}) = \{C_1, C_2, \cdots, C_n\}$, where C_1 till C_n are the equivalence classes of E with respect to $\mathcal{R}$.

Define equivalence relation $\mathcal{R}$ on the hybrid state space E such that $((l_1, val_1), (l_2, val_2)) \in \mathcal{R}$ if and only if $(l_1, l_2) \in \mathcal{R}_l^k$ and (val_1, val_2) are bisimilar with respect to the continuous dynamics.

We now prove that, according to Definition 9.22, $\mathcal{R}$ is a bisimulation.

Suppose $((l_1, val_1), (l_2, val_2)) \in \mathcal{R}$, then

1. $\omega(l_1) = \omega(l_2)$ and $G(l_1, val_1, w) = G(l_2, val_2, w)$ follow from $(l_1, l_2) \in \mathcal{R}_l^0$.

2. Follows from $(l_1, l_2) \in \mathcal{R}_l^0$.

3. If we define for all $C_r \in Part(\mathcal{R}_r^k)$

$$\lambda_{C_r}(\xi) = \begin{cases} \tilde{\lambda} & (\exists(\tilde{\lambda}, G) \in P_j(l, C_r))(\xi \in G) \\ 0 & \text{otherwise.} \end{cases},$$

then we can write $\lambda(\xi) = \sum_{C_r \in Part(\mathcal{R}_r^k)} \lambda_{C_r}(\xi)$ and then $\lambda(\xi_1) = \lambda(\xi_2)$ follows from $P_j(l_1, C_r) = P_j(l_2, C_r)$.

Take any $C_l \in Part(\mathcal{R}_l^k)$ and any saturated Borel set B within the state space of C_l. Let $B_{/\mathcal{R}}$ denote the corresponding set of B in the quotient hybrid state space (with respect to $\mathcal{R}$) and let r_{C_r} denote the reset map on the level of this quotient space corresponding to the (equivalent) reset maps in C_r. Then it can be seen that

$$Q(l_1, val_1)(B) = \sum_{C_r \to C_l} \frac{\lambda_{C_r}(val_1)}{\lambda(val_1)} r_{C_r}(B_{/\mathcal{R}}),$$

where $C_r \to C_l$ denotes $C_r \in \{C \in \mathcal{R}_r^k |$ the target location of any $r \in C$ lies in $C_l\}$, and $Q(\xi)$ is the CFSJS reset measure of the CPDP at state ξ. Because $\lambda_{C_r}(val_1) = \lambda_{C_r}(val_2)$ we get $Q(l_1, val_1)(B) = Q(l_2, val_2)(B)$. It can now be easily seen that this result is also valid for any saturated Borel set of the hybrid state space.

4. The equivalence of the two reset measures of (9.1) can be proved analogously to 3. except that here we get for $\sigma \in \Sigma \cup \bar{\Sigma}$ that

$$Q(l_1, val_1, \sigma)(B) = \sum_{C_r : \to C_l} \frac{p_{C_r}(val_1, \sigma)}{\lambda(val_1)} r_{C_r}(B_{/\mathcal{R}}),$$

where $Q(\xi, \sigma)$ is the NTS reset measure of the σ transition at state ξ, and

$$p_{C_r}(val, \sigma) := \begin{cases} \tilde{p} & (\exists(\tilde{p}, G) \in P_s(l, \sigma, C_r))(val \in G) \\ 0 & \text{otherwise.} \end{cases}$$

Then because $P_s(l_1, \sigma, C_r) = P_s(l_2, \sigma, C_r)$ we get $p_{C_r}(val_1, \sigma) = p_{C_r}(val_2, \sigma)$ and consequently $Q(l_1, val_1, \sigma)(B) = Q(l_2, val_2, \sigma)(B)$. □

We define the *maximal bisimulation* on the set of locations as the equivalence relation that contains all pairs of locations that are bisimilar according to Definition 9.32. Now we question whether the algorithm above provides the *maximal bisimulation* on the set of locations. The answer is: not in all cases. We give an example where the bisimulation found with the algorithm is not maximal and we show why in cases like this the algorithm ends up in a partition that is too refined.

Example 9.37. Consider a CPDP with three locations, l_1, l_2 and l_3. From l_1 there are two a-transitions to l_3 with non-equivalent reset measures r_1 and r_2. From l_2 there is one a-transition to l_3 with reset measure $r_3 = \frac{1}{2}r_1 + \frac{1}{2}r_2$. The guards of all three transitions are equal to the whole valuation space of their origin locations. The internal scheduler S^i assigns at all states of location l_1 probability $\frac{1}{2}$ to both a-transitions. There are no more transitions in the CPDP and l_1 and l_2 are bisimilar with respect to the continuous dynamics. It is clear that the locations l_1 and l_2 are bisimilar because r_3 is equivalent to the combination of r_1 and r_2 of the two scheduled a-transitions. However, the algorithm does not check if one reset measure is equivalent to a weighted sum of other reset measures and because $P_l(l_1, a, \{r_1\}) = \{(\frac{1}{2}, E_O)\} \neq P_l(l_2, a, \{r_1\}) = \{(0, E_O)\}$, where E_O is the output space of the CPDP, the algorithm will refine $\mathcal{R}_l^0 = \{\{l_1, l_2, l_3\}\}$ into $\mathcal{R}_l^1 = \{\{l_1\}, \{l_2\}, \{l_3\}\}$ while l_1 and l_2 are bisimilar.

For spontaneous transitions we can give a similar example where the algorithm is refining too much. Then, the reset measure of a spontaneous transitions is equivalent to the weighted sum of reset measures of two other spontaneous transitions, while the jump rate of the spontaneous transition equals the sum of jump rates of the other two spontaneous transitions.

Non-scheduled CPDPs

The first algorithm can be simply adjusted in such a way that it works for non-scheduled CPDPs as well.

Theorem 9.38. *For a CPDP X, let the partition $P_j(l, C_r)$ be defined as in Section 9.3.1. Define $P_s(l, \sigma, C_r)$ as*

$$P_s(l, \sigma, C_r) := \cup_{\alpha \in \mathcal{T}_{l \xrightarrow{\sigma} C_r}} G_\alpha,$$

where $\mathcal{T}_{l \xrightarrow{\sigma} C_r}$ is the set of all active or passive σ-transitions from l with reset map in C_r and G_α is the guard (as output states) of α. (The guard of a passive transition equals the whole output space of the origin location). Then, Algorithm 9.35 computes a bisimulation relation on the set of locations of X.

Proof. Items 1,2 and 3 are the same as in the proof of Theorem 9.36. Item 4 is almost trivial now. For non-scheduled CPDPs we should have that for two bisimilar states ξ_1 and ξ_2 we have for all $\sigma \in \Sigma \cup \bar{\Sigma}$ that if $\xi_1 \xrightarrow{\sigma} m_1$, then there is an m_2 equivalent to m_1 such that $\xi_2 \xrightarrow{\sigma} m_2$. This is equivalent to saying: if ξ_1 is in the guard of a transition with reset map m_1, then ξ_2 should be in the guard of a transition with equivalent reset map. This is exactly stated by the new definition $P_s(l, \sigma, C_r) := \cup_{\alpha \in \mathcal{T}_{l \xrightarrow{\sigma} C_r}} G_\alpha$. (Note that here we obtain that two states in $\mathcal{R}$ have the same output and the guards of the transitions are defined as subsets of the output space). □

Improved algorithm

We give an improvement of Algorithm 9.35 in such a way that certain unnecessary refinements done by Algorithm 9.35 are not done anymore. For certain CPDPs, this improved algorithm will give a coarser partition of the location set, which is still a bisimulation. As we said before, the price to be paid is that this second algorithm is computationally more expensive.

The set of reset measures R, which is to be refined by the algorithm, should now not only contain all reset measures used by the transitions, but should also contain all weighted sums of reset measures used by the CPDP together with its internal scheduler. In other words, a weighted-sum reset measure

$$r_1 m_1 + r_2 m_2 + \cdots + r_n m_n$$

is contained in R if and only if there is a state ξ such that at least one of the following two conditions is true.

1. For some $\sigma \in \Sigma \cup \bar{\Sigma}$, there exist σ-transitions α_1 till α_n that have reset measures m_1 till m_n and have scheduling values r_1 till r_n at state ξ and $\sum_{i=1\cdots n} r_i = 1$.
2. There exist spontaneous transitions α_1 till α_n that have reset measures m_1 till m_n and have jump rates λ_1 till λ_n at state ξ and $r_i = \frac{\lambda_i}{\lambda}$ and $\sum_{i=1\cdots n} \lambda_i = \lambda$, where λ is the total jump rate at state ξ.

The partitioning functions P_s and P_j should now be defined differently because this 'weighted sum of reset measures' should be incorporated:

$$P_s(l, \sigma, C_r) :=$$

$$\bigcup_{p \in \mathbb{R}_+} \{(p, \{y | \sum_{\alpha \in \mathcal{T}_{l,y\xrightarrow{\sigma}}} S(y)(\alpha) = p, \sum_{\alpha \in \mathcal{T}_{l,y\xrightarrow{\sigma}}} S(y)(\alpha) rmap(\alpha) \in C_r\})\} - \bigcup_{p \in \mathbb{R}_+} \{(p, \emptyset)\},$$

where $\mathcal{T}_{l,y\xrightarrow{\sigma}}$ denotes the set of all σ-transitions leaving l that are enabled at y. (Remember that $rmap(\alpha)$ is constant, i.e., does not depend on y).

$$P_j(l, C_r) :=$$

$$\bigcup_{\lambda \in \mathbb{R}_+} \{(\lambda, \{y | \sum_{\alpha \in \mathcal{S}_{l\rightarrow}} \lambda_\alpha(y) = \lambda, \sum_{\alpha \in \mathcal{S}_{l\rightarrow}} \frac{\lambda_\alpha(y)}{\lambda} rmap(\alpha) \in C_r\})\} - \bigcup_{\lambda \in \mathbb{R}_+} \{(\lambda, \emptyset)\}.$$

In the case of internally scheduled CPDPs, S in the definition of P_s denotes an internal scheduler. In the case of non-scheduled CPDPs P_s is defined as in Theorem 9.38 (i.e., P_s does not change for the improved algorithm for non-scheduled CPDPs).

Define R as the set of all used weighted sum reset measures. The improved algorithm now consists of the following three steps.

Algorithm 9.39.

Step 1 Equals step 1 of Algorithm 9.35.

Step 2 Determine $\mathcal{R}_r^0 \subset R \times R$ such that $(r, r') \in \mathcal{R}_r$ if and only if r and r' are equivalent with respect to $\mathcal{R}^0 := \{((l_1, val_1), (l_2, val_2)) | (l_1, l_2) \in \mathcal{R}_l^0, (val_1, val_2) \in \mathcal{R}_{l_1,l_2}\}$.

Step 3 Determine inductively for each $k \in \mathbb{N}$

$$\mathcal{R}_l^{k+1} = \mathcal{R}_l^k \cap \{(l,l')|(\forall \sigma \in (\Sigma \cup \bar{\Sigma}))(\forall C_r \in \mathcal{R}_r^k)$$

$$(P_s(l,\sigma,C_r) = P_s(l',\sigma,C_r) \wedge P_j(l,C_r) = P_j(l',C_r))\}$$

$$\mathcal{R}_r^{k+1} = \mathcal{R}_r^k \cap \{(r,r')|(\forall C_l \in \mathcal{R}_l^{k+1})(r \sim r')\},$$

where $r \sim r'$ means that r and r' are equivalent with respect to $\mathcal{R}^{k+1} := \{((l_1, val_1),(l_2,val_2))|(l_1,l_2) \in \mathcal{R}_l^{k+1},(val_1,val_2) \in \mathcal{R}_{l_1,l_2}\}$.

Theorem 9.40. *If algorithm 9.39 reaches a fixed point $\mathcal{R}_l^k$ for some $k \in \mathbb{N}$, then this fixed point is a bisimulation on the set of locations. This result holds for internally scheduled CPDPs. If P_j is defined as in Theorem 9.38, the result also holds for non-scheduled CPDPs.*

Proof. The difference with Algorithm 9.35 is that now the set R of used reset measures in step 2 is larger and the initial partititon on the set of locations may be coarser. The updating steps in the algorithm are still the same, except that P_s and P_j are defined a bit differently and determining $\mathcal{R}_r^{k+1}$ is more elaborate now because checking equivalence of reset measures is more difficult since reset measures can have multiple target locations. In spite of these differences, the proof (which now works with a larger set of reset measures etc.) of Theorem 9.36 can be applied to this improved algorithm as well. For non-scheduled CPDPs the NTS part does not change for this new algorithm, because there is no scheduler here and therefore 'sum of reset measures' are not relevant here. From Theorem 9.38 we then find that the result also holds for non-scheduled CPDPs. □

Still we can not claim that the fixed point of our improved algorithm is the maximal bisimulation on the set of locations. Although CPDPs of the type of Example 9.37 are now taken care of because weighted sum reset measures are compared, there are still situations where a class of locations is refined by the algorithm while all locations in that class are bisimilar. This happens for example when two locations jump to bisimilar states of non-bisimilar locations. These locations do not fall in the same class by the algorithm because they have reset measures to non-bisimilar locations. However, still the locations might be bisimilar.

9.3.3 Decidability

Algorithms 9.35 and 9.39 are algorithms for general CPDP bisimulation. They will not be decidable in general, i.e., it can not be guaranteed that in general the algorithms terminate after a finite number of execution steps. In this section we pose some conditions under which the algorithms terminate at a fixed point and are decidable. The contents of this section applies to both algorithms. One of the conditions that we state for decidability is that the set of reset measures (used by the CPDP) is finite. From a compositionality point of view, the identity reset map (i.e., the reset map that leaves the state variables unchanged) is very important. However, because each continuous state has its own identity reset measure, the number of reset measures used by the identity reset map is infinite. At the end of this section we provide a decidable method which, under certain conditions, finds the fixed point of the algorithm while allowing the use of identity reset maps.

Decidability conditions

Each of the three steps of the algorithms, asks for its own decidability conditions.

In step 1, maximal continuous bisimulations should be computed among the dynamics of the different locations. This procedure is decidable if the number of locations is finite and the procedure to find the maximal bisimulation of two locations is decidable. An example of decidable continuous bisimulation is linear state/output systems, i.e., the continuous dynamics of each location can be written as

$$\dot{x} = Ax, y = Cx,$$

with A and C constant real matrices. Finding the maximal bisimulation between two linear state/output systems can be done by using the method of [vdS04a], where the algorithm consists of a finite number of standard matrix manipulations and is thus decidable.

In step 2, the set of all reset measures used by the CPDP should be partitioned into classes of equivalent measures (with respect to the results of step 1). This procedure is decidable if the number of used (sums of) reset measures is finite and the procedure to decide whether two reset measures are equivalent is decidable. In the case of linear state/output systems, checking whether two states are bisimilar can be done via a finite number of standard matrix operations and is therefore decidable. Consider the class of reset measures that assign a positive probability to a finite number of states. Two of these reset measures are equivalent if each state in the quotient space gets the same probability from both reset measures. It is decidable to determine which of the states (of positive probability) corresponds to the same quotient state and the number of quotient states that have to be checked are finite. Therefore, for this class of reset measures, the computation of the second step of the algorithm is decidable. (Note that the reset measures of identity reset maps fall within this class, but the number of them is for each identity reset map infinite).

In step 3, we have to split guards into parts with equal jump-rate and for each active or passive event into parts with equal scheduling value. Because transitions with the same action are combined in this step, we have to calculate intersections of guards. If two guards are both given as a set of linear inequalities, then the computation of their intersection is decidable. Therefore we have that if

- the guard of each transition can be given as a finite set of linear inequalities,
- the number of different scheduling values is for each transition finite and the 'guard-areas of equal scheduling value' can all be given as a finite set of linear inequalities,
- for each transition, the number of reset measures used by the reset map of the transition is finite and the guard area corresponding to each reset measure

can be given as a finite set of linear inequalities and

- for each spontaneous transition the number of different jump rates is finite and the area corresponding to each jump-rate can be given as a finite set of linear inequalities,

then for each $a \in (\Sigma \cup \bar{\Sigma})$ and each finite set of reset measures C_r, the computation of $P_s(l, a, C_r)$ and of $P_j(l, C_r)$ is decidable. Now it can be seen that if the number of locations and the number of transitions used is finite, then under the conditions listed above, the computation of step 3 is decidable.

Identity reset maps

The identity reset map for a state variable v is defined as $Id_v(\{(v = r)\}, (v = r)) = 1$, in other words, if variable v is reset with Id_v at the moment that its value equals r, then the probability that the value of v after reset equals r, equals one because the singleton Borel set $\{(v = r)\}$ gets measure one. If we have two CPDPs, X_1 and X_2, interacting with each other, and CPDP X_1 executes a transition (with reset map R) which does not influence X_2, then in the composite CPDP the variables of X_1 are reset with R while the variables of X_2 are reset with Id, which expresses that X_2 is not influenced by the transition of X_1. This is a common situation, which makes clear the importance of identity reset maps. Another situation where the identity reset map is used, is when a CPDP component wants to send, at a certain state, a signal a to another component, while its state should not be changed. This can be expressed via an active a-transition with identity reset map.

In order to use identity reset maps, while still allowing decidable computation of Algorithm 9.35, we need to syntactically add more structure to the relation between state and output variables of a CPDP location. Composing CPDPs naturally leads to different *compartments* of a joint location (i.e., a location of the composite CPDP), where each compartment contains the state and output variables of a specific component-CPDP. For example, consider the two CPDPs X and Y. X has one location, l_X, with state variable x and output variable y, and Y has one location, l_Y, with state variable x' and output variable y'. In the composition of X with Y there is one location, which is the product location of l_X and l_Y. This location has state variables x and x' and has output variables y and y'. Here x and y form one compartment, and x' and y' form another compartment. In general, for each compartment we have that the output variables of that compartment do not depend on state variables of other compartments. Instead of ν and ω, which select state and output variables for each location, we will now use γ, which assigns a set of compartments to each location. We define a compartment as a combination of a set of state variables and a set of output variables like $(\{v_1, v_2\}, \{w_1, w_2\})$ and we might have for example for some location l that $\gamma(l) = \{(\{v_1, v_2\}, \{w_1, w_2\}), (\{v_3, v_4\}, \{w_3\})$. Two compartments of one location have disjoint sets of state variables and have disjoint sets of output variables. Output variables may depend only on the state variables of its compartment.

Now we define the concept of *extended reset measure*: an extended reset measure on a location l assigns to each compartment of location l either a reset measure on the state variables of that compartment, or the symbol Id. Assigning the symbol Id to a compartment, expresses that this compartment is reset via the identity reset.

For Algorithm 9.35, instead of using the set of all reset measures used by the CPDP we now use the set of all *extended* reset measures used by the CPDP. Note that under the decidability conditions stated before, this set of extended reset measures is finite, while we can still express the identity reset action. We call the extended reset measures r_1 and r_2 *equivalent* if first for each compartment with output variables set W that is reset by r_1 there is a compartment with output variables set W that is reset by r_2 and vice versa, and secondly, two corresponding compartments (i.e., two compartments with the same output variables set) should be reset by equivalent reset measures or should both get assigned the symbol Id to it by r_1 and r_2. Now it can be seen that Algorithm 9.35, where now R is the set of extended reset measures and the phrase 'equivalent measures' in step 2 is changed to 'equivalent extended measures', determines a bisimulation for CPDPs with compartments and extended reset measures. A similar story applies to Algorithm 9.39.

9.3.4 Example

We give an example of finding a bisimulation relation for an internally scheduled CPDP. Consider the CPDP X in Figure 9.2. The exact specification of X can be found in Example 9.16. The internal scheduler S^i is defined on the five a-transitions as follows:

- $S^i((l_2, y), a, l_2 \xrightarrow{a} l_1)$ equals 1 for $y \leq 0$ and equals 0 for $y > 0$,
- $S^i((l_2, y), a, l_2 \xrightarrow{a} l_3)$ equals 0 for $y \leq 0$ and equals 1 for $y > 0$,
- $S^i((l_3, y), a, l_3 \xrightarrow{a} l_1)$ equals 1 for $y \leq 0$ and equals 0 for $y > 0$,
- $S^i((l_3, y), a, l_3 \xrightarrow{a} l_2)$ equals 0 for $y \leq 0$ and equals $\frac{1}{2}$ for $y > 0$,
- $S^i((l_3, y), a, l_3 \xrightarrow{a} l_3)$ equals 0 for $y \leq 0$ and equals $\frac{1}{2}$ for $y > 0$.

We execute Algorithm 9.35 for CPDP X.

Step 1 Via the algorithm of [vdS04a], we can find via matrix operations maximal bisimulations for the continuous dynamics of locations l_1, l_2 and l_3. We assume that according to this algorithm all three locations are bisimilar with respect to the continuous dynamics. Then we get $\mathcal{R}_l^0 = \{\{l_1, l_2, l_3\}\}$. Let the maximal state reduced versions of the continuous dynamics of l_1,l_2 and l_3, which can be computed with the same algorithm, be given by $\dot{\tilde{x}}_1 = \tilde{A}_1\tilde{x}_1$, $y = \tilde{C}_1\tilde{x}_1$ and $\dot{\tilde{x}}_2 = \tilde{A}_2\tilde{x}_2$, $y = \tilde{C}_2\tilde{x}_2$ and $\dot{\tilde{x}}_3 = \tilde{A}_3\tilde{x}_3$, $y = \tilde{C}_3\tilde{x}_3$. Remember that the initial states of l_1 and l_2, and consequently the initial states of $\hat{l}_1$ and $\hat{l}_2$, are bisimilar.

Step 2 From the results and assumptions of step 1 above, it is clear that we get $\mathcal{R}_r^0 = \{\{r_1, r_2, r_3\}\}$.

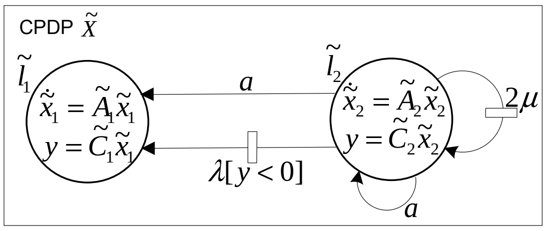
CPDP $\tilde{X}$
$\tilde{l}_1$
$\dot{\tilde{x}}_1 = \tilde{A}_1\tilde{x}_1$
$y = \tilde{C}_1\tilde{x}_1$
$\tilde{l}_2$
$\dot{\tilde{x}}_2 = \tilde{A}_2\tilde{x}_2$
$y = \tilde{C}_2\tilde{x}_2$
a
$\lambda[y < 0]$
2μ
a

Figure 9.3: Example of a bisimulation-reduced CPDP

Step 3 We can compute that for l_1 we get $P_s(l_1, a, \{r_1, r_2, r_3\}) = 0$, $P_j(l_1, \{r_1, r_2, r_3\}) = 0$, for l_2 we get $P_s(l_2, a, \{r_1, r_2, r_3\}) = \{(1, y \in \mathbb{R})\}$, $P_j(l_2, \{r_1, r_2, r_3\}) = \{(\lambda + 2\mu, y < 0), (2\mu, y \geq 0)\}$ and for l_3 we get $P_s(l_3, a, \{r_1, r_2, r_3\}) = \{(1, y \in \mathbb{R})\}$, $P_j(l_3, \{r_1, r_2, r_3\}) = \{(\lambda + 2\mu, y < 0), (2\mu, y \geq 0)\}$. This means that we find $\mathcal{R}_l^1 = \{\{l_1\}, \{l_2, l_3\}\}$ and $\mathcal{R}_r^1 = \{\{r_1\}, \{r_2, r_3\}\}$.

We continue with these new partitions and compute that for l_2 we get $P_s(l_2, a, \{r_1\}) = \{(1, y \leq 0)\}$, $P_s(l_2, a, \{r_2, r_3\}) = \{(1, y > 0)\}$, $P_j(l_2, \{r_1\}) = \{(\lambda, y < 0)\}$, $P_j(l_2, \{r_2, r_3\}) = \{(2\mu, y \geq 0)\}$ and for l_3 we get $P_s(l_3, a, \{r_1\}) = \{(1, y \leq 0)\}$, $P_s(l_3, a, \{r_2, r_3\}) = \{(1, y > 0)\}$, $P_j(l_3, \{r_1\}) = \{(\lambda, y < 0)\}$, $P_j(l_3, \{r_2, r_3\}) = \{(2\mu, y \geq 0)\}$. This means that we get $\mathcal{R}_l^2 = \{\{l_1\}, \{l_2, l_3\}\}$ and $\mathcal{R}_r^2 = \{\{r_1\}, \{r_2, r_3\}\}$, which is the fixed point of the algorithm.

From Theorem 9.36 we now know that l_2 and l_3 are bisimilar and l_1 is not bisimilar to l_2 and l_3. This means that CPDP $\tilde{X}$ of Figure 9.3 represents a bisimilar state reduced version of CPDP X. It can be checked that for this example the refinements done by the algorithm are all needed to find a bismilation relation. This means that the fixed point, where l_2 and l_3 are bisimilar, is the maximal bisimulation on the set of locations. It can also be seen that for each state ξ in $\tilde{l}_1$ and each state ξ' in $\tilde{l}_2$, xi and ξ' are not bisimilar. This means that CPDP $\tilde{X}$ is a maximal state reduced version of X (i.e., no two different states of $\tilde{X}$ are bisimilar).

10

CPDP example: ATM system

In Figure 10.1 we see a schematic representation of some entities and the interactions between them, that play a role in Air Traffic Management (ATM). It forms a part of a system that models the behavior of a pilot who is controlling a flying aircraft. This pilot is called the ‘pilot flying’. (Besides the ‘pilot flying’ there is also the co-pilot or ‘pilot not flying’ in the cockpit who we do not consider in our system). This example comes from [Obb05] and [EKBO04], where it is modelled as a Dynamically Coloured Petri Net (DCPN). There, the DCPN ‘pilot flying’ forms one of multiple components of a larger DCPN that models a ‘free flight’ situation. The DCPN ‘pilot flying’ is in itself already a large and complex model. In this chapter we model a simplified version of the ‘pilot flying’ system as a value-passing CPDP.

The chapter is organized as follows. In Section 10.1, we give a description of the ‘pilot flying’ system. In Section 10.2 we model the system as a CPDP and we apply the PDP conversion algorithm of Chapter 8 to this model. In Section 10.3 we give the DCPN model of the system. The DCPN framework has been developed for compositional modelling and analysis of PDP type systems. Illustrated by the CPDP and DCPN models of the ‘pilot flying’ system, we compare the CPDP and DCPN frameworks from a compositionality point of view.

10.1 ‘Pilot flying’ system description

We consider the following two components (see Figure 10.1).

Pilot flying Models the pilot executing different tasks during the flight.

Audio Alert Models a communication device that communicates to the pilot which tasks need to be executed at which times.

We compose ‘pilot flying’ from three subsystems: Current Goal, Task Performance and Memory. These three systems do not correspond to actual systems, but form a way to systematically model the behavior of the pilot. Task Performance is the part of ‘pilot flying’ that is actually executing a task. This system contains the information and structure needed to execute the different tasks. Current Goal is a control unit. It keeps track of which task is executed and which tasks need to be executed in the future. It forms the interface between Audio Alert, Memory and

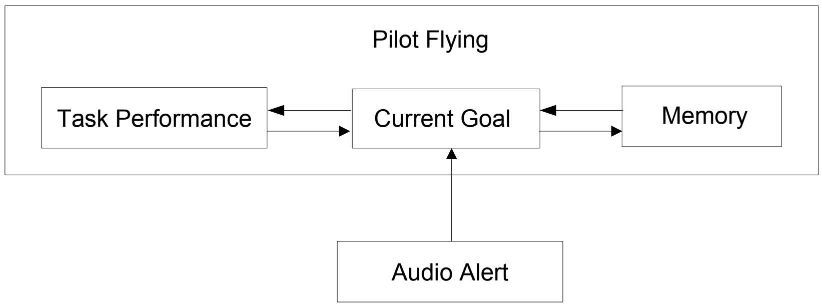

Figure 10.1: 'Pilot flying' system structure

Task Performance in the following sense: Audio Alert communicates to Current Goal that a task needs to be executed. Then, either Current Goal communicates to Task Performance to execute the task, or, if the pilot is still busy executing a task, stores the task into Memory such that when the pilot is not busy anymore, it can retrieve this information from the Memory and execute the task. Thus, Current Goal models in some sense the awareness of the pilot what he is doing and what he needs to do in the future. Memory models the memory (or a memory aid) of the pilot.

The communication structure is pictured in Figure 10.1. We now describe the system and the communication within it in more detail.

During the flight, there are seven distinguished tasks that might be executed by the pilot. These are

C1 Collision avoidance

C2 Emergency actions

C3 Conflict resolution

C4 Navigation vertical

C5 Navigation horizontal

C6 Preparation route change

C7 Miscellaneous

Task C2, emergency actions, is split up into six distinct tasks. Each C2-task corresponds to a specific kind of emergency action. Task C2.1 is executed in case of an engine failure, task C2.2 is executed in case of a navigation-system failure. Tasks C2.3 till C2.5 correspond to failures of other aircraft systems. Task C2.6 is titled 'other emergency'.

The ordering is an ordering of priority. This means that C1, collision avoidance, has higher priority than C2, emergency action, which has higher priority than C3, conflict resolution, etc. If the pilot is executing task C2 and Audio Alert communicates that task C1 needs to be done, then, because C1 has higher priority, finishing task C2 needs to be postponed and C1 is executed immediately. In this case 'executing task C2' should be put into Memory, such that the pilot 'knows' after executing task C1 that he still has to execute task C2. Vice versa, if the pilot is executing task C1 and Audio Alert communicates that task C2 needs to be executed, then the pilot should put 'execute C2' into memory and continue executing task C1. There is no ordering for the tasks C2.1 till C2.6. If the pilot starts executing task C2 while there are multiple failures, i.e., multiple tasks of the set of tasks C2 need to be executed, then a random choice is made about which C2 task is executed.

We will model this system in detail as far as it concerns the control unit Current Goal and its Memory. The number of details of the Task Performance system is large and the interactions happening within Task Performance are complex. We do not model these details and we use a very simple unrealistic model for the Task Performance. It is our goal to show that interactions between these subsystems can be modelled through composition of value passing CPDPs and in order to that we do not need to model all the details of the ATM system. Also, the Audio Alert system will be modelled in a simple unrealistic way, where we assume that the time that a task needs to be executed is exponentially distributed for each task.

10.2 The CPDP model

10.2.1 Specification

In Figure 10.2, the CPDPs $CurrentGoal$, $TaskPerformance$, $Memory$ and $AudioAlert$ model the 'pilot flying' system. $CurrentGoal$ and $Memory$ are the only parts of the system that are modelled in detail. We first describe the CPDP $AudioAlert$ which, in a simplified way, models the Audio Alert system.

CPDP *AudioAlert*

Location l_{10} of $AudioAlert$ models the situation where an alert signal might be 'generated'. l_{10} is an empty location, i.e., there are no continuous dynamics at this location. Constant λ is the parameter of the exponential distributed time indicating the time of an alert. If an alert signal is generated at time t, meaning that a task needs to be executed at time t, then $AudioAlert$ switches to location l_{11}. Reset map R_{15} resets the state variables k_a and q_a of location l_{11}. The value of k_a and q_a denote the task that needs to be executed. $k_a = i$ corresponds to task Ci and in case $k_a = 2$, $q_a = j$ corresponds to task C2.j. If $k_a \neq 2$, then q_a is irrelevant and equals zero.

Let the rate of occurrence that task C needs to be executed be equal to λ_C. Then we get

$$\lambda = \sum_{C \in Tasks} \lambda_C,$$

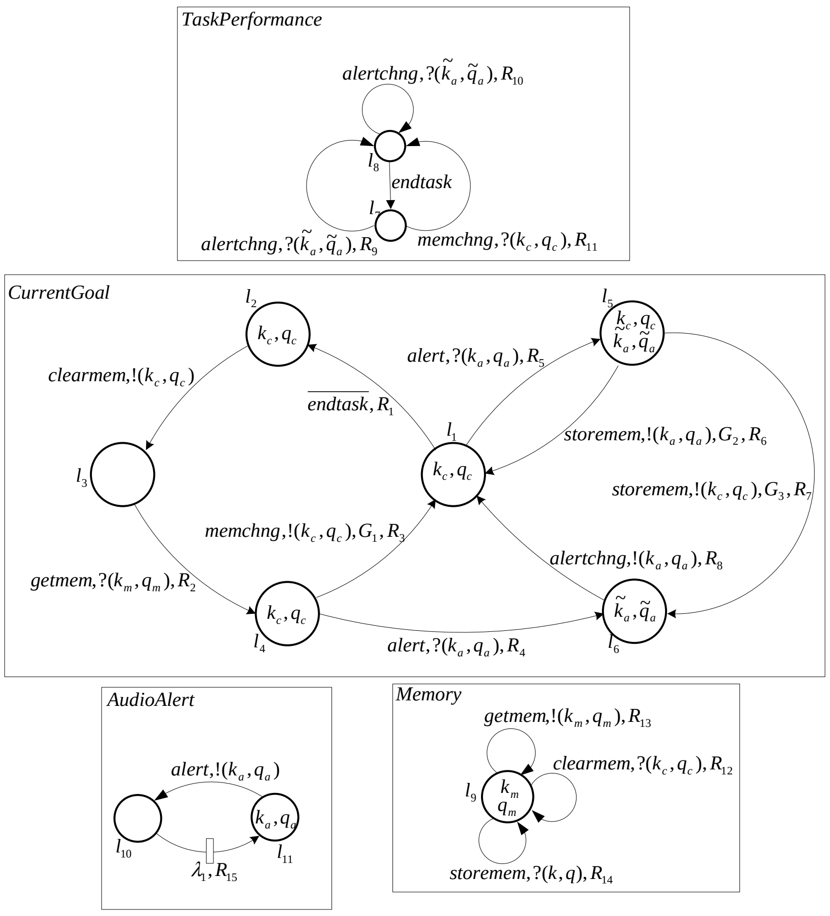

Figure 10.2: CPDP 'pilot flying' model

where $Tasks = \{C1, C2.1, C2.2, C2.3, C2.4, C2.5, C2.6, C3, C4, C5, C6\}$ and

$$R_{15} = \sum_{C \in Tasks} \frac{\lambda_C}{\lambda} R_C,$$

where R_{C1} resets $k_a := 1$ and $q_a := 0$, $R_{C2.1}$ resets $k_a := 2$ and $q_a := 1$, $R_{C2.2}$ resets $k_a := 2$ and $q_a := 2$, R_{C3} resets $k_a = 3$ and $q_a = 0$, etc. The *alert*-transition from l_{11} to l_{10} is executed immediately after the spontaneous transition, because there is no guard. (Technically, the guard equals the whole state space of location l_{11}). In this ATM example we do not distinguish between state and output variables. Formally, we model this by having for each state variable x a copy output variable y_x whose value equals the state value everywhere. The value passing part $!(k_a, q_a)$ expresses that the values of variables k_a and q_a are value-passed in a synchronization with component *CurrentGoal*, as we will see later. (Formally only output variables can be value-passed and the value passing part should therefore formally be equal to $!(y_{k_a}, y_{q_a})$.)

We see that location l_{11} is an intermediate location where no time is consumed, which only serves the value passing of the alert signal (k_a, q_a) via channel *alert*.

CPDP $TaskPerformance$

The empty location l_7 of CPDP $TaskPerformance$ denotes the situation where the pilot is not executing a task and is waiting for a new task to be executed. If a new task needs to be executed, $TaskPerformance$ switches to location l_8 which denotes the situation where a task is executed. The dynamics of 'executing a task' is large and complex in the 'pilot flying' system. We do not model this complexity and model it as one location, l_8. We unrealistically assume that the dynamics of the execution of a task is expressed by a differential equation $\dot{x} = f(x)$, where each value for x denotes a state somewhere in the execution of some task. We assume that for each task $C \in Tasks$ a state x_C exists such that evolution from x_C denotes starting and evolution of task C. We also assume that if x_C enters guard area G_4, then the task of execution is completed. If a task is completed, $TaskPerformance$ switches to l_7 via the active *endtask*-transition with guard G_4. The *endtask* signal will be received by CPDP *CurrentGoal*.

There are two ways in which a task execution is started. First, if CPDP *CurrentGoal* executes a *alertchng*-transition with value passing $!(k_a, q_a)$. Execution of this transition by *CurrentGoal* denotes the situation where, as we will see later in detail, *CurrentGoal* has received a (k_a, q_a) signal from *AudioAlert*. This value (k_a, q_a) is received by the *alertchng*, $?(k_a, q_a)$-transition from location l_7 of $TaskPerformance$. Then, the reset map R_9 resets state x of l_8 to value x_C, where C is the task that corresponds to the passed value (k_a, q_a). Note that the *alert*, $?(k_a, q_a)$ transition is formally specified as *alert*, $?U$ with $U = \{(1,0),(2,1),(2,2),(2,3),(2,4),(2,5),(2,6),(3,0),(4,0),(5,0),(6,0),(7,0)\}$, the set of all values corresponding to all tasks. If the pilot is execution a task, thus, if $TaskPerformance$ is in location l_8, and a task with a higher priority needs to be executed, then this switching of tasks is expressed by the *alertchng*-transition from l_8 to itself. Reset map R_{10} is equal to R_9.

The second situation where a task execution is started, is the one where *CurrentGoal* executes a *memchng*-transition with value passing $!(k_c, q_c)$. This expresses the situation where after the completion of a task a new to-be-executed-task is retrieved from the memory by *CurrentGoal* and stored in variables k_c and q_c, after which the *memchng*, $!(k_c, q_c)$-transition is executed. The value (k_c, q_c) is received by the *memchng*, $?(k_c, q_c)$-transition from location l_7 of $TaskPerformance$. Reset map R_{11} resets x to x_C, with C the task corresponding to received value (k_c, q_c).

CPDP *Memory*

CPDP *Memory* has one location l_9 with state variables k_m and q_m. There is no continuous dynamics, i.e., $\dot{k}_m = \dot{q}_m = 0$. k_m has seven components, i.e., $k_m = (k_{m,1}, k_{m,2}, \cdots, k_{m,7})$ and takes value in $\mathbb{R}^7$. q_m has six components, i.e., $q_m = (q_{m,1}, q_{m,2}, \cdots, q_{m,6})$ and takes value in $\mathbb{R}^6$. If at some time $k_{m,i} = 1$, then this means that task Ci needs to be executed because, as we will see, it is the consequence of *CurrentGoal* putting the value $k_{m,i} := 1$ to place task Ci on the stack. Similarly, $q_{m,i} = 1$ means that task C2.i is put on the stack. If a task Ci is not on stack, then $k_{m,i}$ equals zero. For example $k_m = (0, 1, 1, 0, 0, 0, 0)$ and $q_m = (1, 1, 0, 0, 0, 0)$ means that tasks C2.1, C2.2 and C3 needs are put on stack. This situation can happen in the unlucky situation where the pilot is executing task C1, collision avoidance, which has highest priority, while Audio Alert communicates that the engine and navigation systems switched to failure mode.

There are three transitions corresponding to three memory actions. The three memory actions are: retrieving the memory state, storing a new to-be-executed-task in the memory and clearing a to-be-executed-task from the memory as soon as this task has been completed by the pilot.

Retrieving the memory state is done via the $getmem, !(k_m, q_m)$ transition. The memory value (k_m, q_m) is then passed to CPDP *CurrentGoal*. Reset map R_{13} is the identity reset map, i.e., it does not change the state of the memory.

Storing a new task in memory is done via the $storemem, ?(k, q)$ transition. The value (k, q) is passed by *CurrentGoal* to *Memory* via this transition. Reset map R_{14} does not change the memory state except that for $i = k$ and $j = q$, $k_{m,i}$ and $q_{m,j}$ are reset to one.

Removing a task from the memory is done via the $clearmem, ?(k, q)$ transition. The value (k, q) is passed by *CurrentGoal* to *Memory* via this transition, indicating that the corresponding task is completed. Reset map R_{12} does not change the memory state except that for $i = k$ and $j = q$, $k_{m,i}$ and $q_{m,j}$ are reset to zero.

CPDP *CurrentGoal*

During the flight situation where the pilot is not working on a task and there are no tasks in memory, CPDP *CurrentGoal* is in location l_4 with $k_c = q_c = 0$. $k_c = 0$ indicates here that no task needs to be executed. If $k_c \neq 0$ in l_4, then, as we will see later, this would indicate that a task needs to be executed and then the *memchng*-transition to l_1 would be taken. Suppose that $k_c = q_c = 0$ in l_4. If an *alert* signal is executed by *AudioAlert*, then *CurrentGoal* switches to l_6 with value passing transition $alert, ?(k_a, q_a)$. Reset map R_4 copies the input values of the transition (k_a, q_a) to the variables $\tilde{k}_a$ and $\tilde{q}_a$. Thus, in l_4 $\tilde{k}_a$ and $\tilde{q}_a$ correspond to the task that needs to be executed according to *AudioAlert*. The guardless transition $alertchng, !(\tilde{k}_a, \tilde{q}_a)$ to l_1 is taken immediately, inducing the $alertchng, ?(\tilde{k}_a, \tilde{q}_a)$-transition in $TaskPerformance$, which means that the task corresponding to $(\tilde{k}_a, \tilde{q}_a)$ will be executed. Reset map R_5 puts the values of $\tilde{k}_a, \tilde{q}_a$ into the variables k_c and q_c at location l_1. Thus, at l_1, k_c and q_c correspond to the task that is currently worked on.

At l_1, two things can happen: 1. An *alert*-signal from *AudioAlert* is received, 2. An *endtask*-signal from $TaskPerformance$ is received.

In case 1, *CurrentGoal* executes the $alert, ?(k_a, q_a)$ transition and switches to location l_5. Reset map R_3 copies k_c and q_c at l_1 to k_c and q_c at l_5 and R_3 copies inputs k_a and q_a to $\tilde{k}_a$ and $\tilde{q}_a$ at l_5. At l_3 the current task (k_c, q_c) and the requested task $(\tilde{k}_a, \tilde{q}_a)$ need to be compared in order to decide which task has the highest priority. This 'comparing' is coded in the guards G_2 and G_3. G_2 contains all states of l_5 where task (k_c, q_c) has higher or equal priority, i.e.,

$$G_2 = \{(k_c, q_c, \tilde{k}_a, \tilde{q}_a) | (k_c, q_c) > (\tilde{k}_a, \tilde{q}_a)\},$$

where $(k_c, q_c) > (\tilde{k}_a, \tilde{q}_a)$ means that task (k_c, q_c) has higher or equal priority than task $(\tilde{k}_a, \tilde{q}_a)$. Equal priority only happens when both tasks are tasks of $C2$. (The tasks of C2 have no ordering). G_3 is the complement of G_2, i.e., G_3 contains all states where task (k_c, q_c) has lower priority. If at l_5, G_2 is satisfied, then the $storemem, !(\tilde{k}_a, \tilde{q}_a)$ transition to l_1 is executed, where R_6 copies k_c and q_c at l_5 to k_c and q_c at l_1. This transition induces the *storemem* transition of $Memory$. The cycle $l_1 \to l_5 \to l_1$, means that the requested task of *AudioAlert* is stored into memory and the task that is currently worked on at l_1 is not changed. If at l_5, G_3 is satisfied, then the $storemem, !(k_c, q_c)$ transition to l_6 is executed. This means that the task that was worked on at l_1 is now stored into memory. Reset map R_7 copies $(\tilde{k}_a, \tilde{q}_a)$ at l_5 to $(\tilde{k}_a, \tilde{q}_a)$ at l_6. At l_6 the requested task $(\tilde{k}_a, \tilde{q}_a)$ needs to be executed, and this happens via the *alertchng*-transition to l_1.

In case 2 at location l_1, *CurrentGoal* waits until the task that is currently worked on is completed. After completion, $TaskPerformance$ sends the *endtask* signal, which induced the $\overline{endtask}$ transition of $CurrentGoal$ to l_2. Reset map R_1 copies (k_c, q_c) at l_1 to (k_c, q_c) at l_2. Thus, at l_2 the state (k_c, q_c) corresponds to the task that has just been completed. This means that this task needs to be removed from the memory. This happens via the $clearmem, !(k_c, q_c)$ transition to l_3, which induced the *clearmem* transition of $Memory$ which removes task (k_c, q_c) from the memory. At l_3, the pilot needs to check the memory, whether there is a to-be-executed-task stored in the memory. The $getmem, ?(k_m, q_m), R_2$ transition to l_4 checks the memory and stores the new to-be-executed-task, or $(0, 0)$ in case there is no new to-be-executed-task, in (k_c, q_c) at l_4. Via this value passing transition, the memory state is passed to $CurrentGoal$. Reset map R_2 should then be defined as:

1. Let $i = \min(\{r | k_{m,r} = 1\} \cup \{0\})$.
2. If $i \neq 2$, then $j := 0$, otherwise, take j randomly from the set $\{r | q_{m,r} = 1\}$.
3. Reset $k_c := i$ and $q_c := j$ at l_4.

Note that R_2 resets $k_c = q_c = 0$ if there are no tasks in the memory, otherwise R_2 takes the task with the highest priority. At l_4 the task from the memory is executed via the *memchng* transition to l_1, or, in case $k_c = q_c = 0$, the pilot 'waits' at l_4 for an alert signal.

Composition

We have specified all four CPDPs. Now we specify the composed CPDP. The shared actions of $CurrentGoal$ and $TaskPerformance$ are $alertchng$ and $memchng$. Thus, if we compose $CurrentGoal$ with $TaskPerformance$ via $|_A^P|$, then $A = \{alertchng, memchng\}$. The shared actions of $CurrentGoal$ and $Memory$ are $getmem$, $clearmem$ and $storemem$. Thus, if we compose $CurrentGoal$ with $Memory$ via $|_A^P|$, then $A = \{getmem, clearmem, storemem\}$. The only shared action of $CurrentGoal$ and $AudioAlert$ is $alert$. Thus, if we compose $CurrentGoal$ with $AudioAlert$ via $|_A^P|$, then $A = \{alert\}$. Then, we get the following specification for the composition of $CurrentGoal$, $TaskPerformance$, $Memory$ and $AudioAlert$.

$$PilotFlying =$$
$$((CurrentGoal|_{A_1}^P|TaskPerformance)|_{A_2}^P|Memory)|_{A_3}^P|AudioAlert, \tag{10.1}$$

where $A_1 = \{alertchng, memchng\}$, $A_2 = \{getmem, clearmem, storemem\}$ and $A_3 = \{alert\}$. Note that P is not relevant, since there is only one passive transition in all four CPDPs. Also note that the order of composition can be changed in any way. Then, the composition operators should also be changed. For example, we can also compose as

$$PilotFlying = (AudioAlert||Memory||TaskPerformance)|_A^P|CurrentGoal,$$

where $||$ denotes the associative operator $|_\emptyset^{\bar{\Sigma}}|$, $A = A_1 \cup A_2 \cup A_3$ and P is not relevant.

10.2.2 CPDP to PDP conversion

We use the theory of Section 8.2 to show that the composite CPDP $PilotFlying$ of (10.1) exhibits a PDP behavior and to find the corresponding PDP of $PilotFlying$.

We consider $PilotFlying$ as a closed system, no more components will be added to the composition. Therefore we use CPDP $PF := [PilotFlying]_{\bar{\Sigma}}$, which equals $PilotFlying$ without the passive $\overline{endtask}$-transitions, for finding the PDP equivalent.

We take as initial situation the situation where no task is executed and where the memory is empty. This means that the initial location of PF is the joint location l_4, l_7, l_9, l_{10}. In Figure 10.3 CPDP PF is pictured. The locations that are not reachable from initial location l_4, l_7, l_9, l_{10} are not shown in Figure 10.3. This leaves nine locations and twelve transitions. It can be easily seen that locations l_1, l_8, l_9, l_{10} and l_4, l_7, l_9, l_{10} are the only locations that are not unstable, i.e., the only locations that contain unguarded states. This means, as we will see, that these two locations will be the only two locations of the PDP corresponding to PF.

We will now determine $\mathcal{A}_s^1$, $\mathcal{A}_u^1$, $\mathcal{A}_s^2$, $\mathcal{A}_u^2$, etc., until we find $\mathcal{A}_u^n = \emptyset$ for some $n \in \mathbb{N}$:

$$\mathcal{A}_s^1 = \{\alpha_3, \alpha_6, \alpha_8, \alpha_{11,s}, \alpha_{12}\}, \quad \mathcal{A}_u^1 = \{\alpha_2, \alpha_5, \alpha_7, \alpha_9, \alpha_{10}, \alpha_{11,u}\},$$

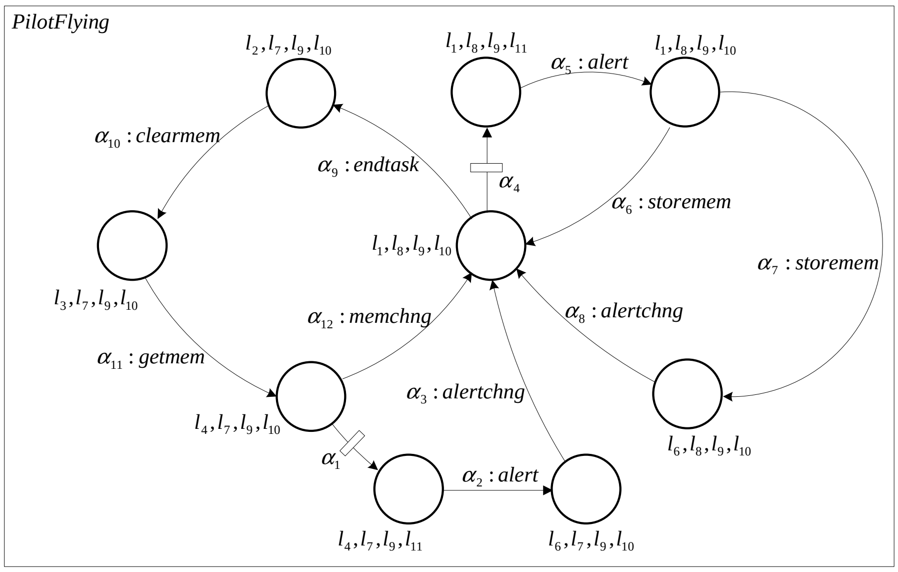

Figure 10.3: Composite CPDP 'pilot flying'

where $\alpha_{11,s}$ and $\alpha_{11,u}$ are the stable and unstable parts of α_{11} respectively.

$$\mathcal{A}_s^2 = \{\alpha_3 \circ \alpha_2, \alpha_6 \circ \alpha_5, \alpha_8 \circ \alpha_7, \alpha_{11,s} \circ \alpha_{10}, \alpha_{12} \circ \alpha_{11,u}\},$$
$$\mathcal{A}_u^2 = \{\alpha_7 \circ \alpha_5, \alpha_{10} \circ \alpha_9, \alpha_{11,u} \circ \alpha_{10}\},$$
$$\mathcal{A}_s^3 = \{\alpha_8 \circ \alpha_7 \circ \alpha_5, \alpha_{11,s} \circ \alpha_{10} \circ \alpha_9, \alpha_{12} \circ \alpha_{11,u} \circ \alpha_{10}\}, \quad \mathcal{A}_u^3 = \{\alpha_{11,u} \circ \alpha_{10} \circ \alpha_9\},$$
$$\mathcal{A}_s^3 = \{\alpha_{12} \circ \alpha_{11,u} \circ \alpha_{10} \circ \alpha_9\}, \quad \mathcal{A}_u^4 = \emptyset.$$

According to Corollary 8.26 CPDP PF has a corresponding PDP with the same TMS semantics. To find this PDP, we can now consider the CPDP PF, but now with the active transitions formed by $\mathcal{A}_s := \mathcal{A}_s^1 \cup \mathcal{A}_s^2 \cup \mathcal{A}_s^3 \cup \mathcal{A}_s^4$. It can be easily seen that $\alpha_{12} \circ \alpha_{11,u} \circ \alpha_{10} \circ \alpha_9$, $\alpha_8 \circ \alpha_7 \circ \alpha_5$, $\alpha_{11,s} \circ \alpha_{10} \circ \alpha_9$, α_6 AND α_3 are the only transitions from $\mathcal{A}_s$ that are reachable from the initial location l_4, l_7, l_9, l_{10}. Now, to find the PDP, we only have to consider the CPDP with the locations l_4, l_7, l_9, l_{10}, l_1, l_8, l_9, l_{10}, l_4, l_7, l_9, l_{11} and l_4, l_7, l_9, l_{11}, the five reachable active transitions and the two spontaneous transitions. This CPDP is pictured in 10.4, where now α_1 till α_5 are the five active transitions and α_6 and α_7 are the two spontaneous transitions. The corresponding PDP can now be found as described in Theorem 8.24.

10.2.3 Bisimulation

The number of locations of the CPDP PF cannot be reduced by bisimulation, i.e., each location has its own unique behavior and cannot be simulated by another location. However, for some locations of PF, the dimension of the continuous state

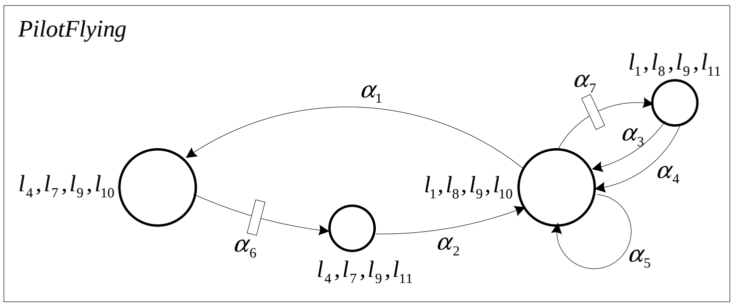

Figure 10.4: Converted CPDP 'pilot flying'

space can be reduced by bisimulation. We show where this state space reduction comes from. Component *CurrentGoal* stores a copy of the *Memory* state in location l_3. This means that in the joint locations with components l_3 and l_9, the state spaces of these locations contain both the original and the copy of the memory state. Because the transitions leaving these joint locations can directly have 'access' to the originals, the copy variables are superfluous. This means that there are state reduced CPDPs that are bisimilar to PF and that do not have these copy variables.

10.3 The DCPN model

Originally, the 'pilot flying' system was modelled as a Dynamically Colored Petri Net (DCPN). In Figure 10.5, we see the DCPN PF_D of our simplified 'pilot flying' system. This DCPN models the same system as the CPDP of Figure 10.2. We briefly explain the DCPN framework by explaining how PF_D models the 'pilot flying' system.

A DCPN has places and transitions. PF_D has ten places, p_1 till p_6 and ipn_1 till ipn_4. Each place may contain one or more tokens. The initial marking of PF_D (see Figure 10.5) has tokens in the places p_1, p_2, p_4 and p_5. A token in a DCPN corresponds to a continuously evolving process. The continuous dynamics within a place p can be characterized by an ordinary differential equation $\dot{x}_p = f_p(x_p)$, where x_p may have different dimensions for different places p. In our discussion here we only consider places with at most one token inside it.

PF_D has nine transitions, G_1 till G_5, D_1 and I_1, I_2 and I_3. Places and transitions are connected to each other via arcs. Every arc is attached to one transition and one place. Either it goes from a place to a transition or from a transition to a place. In this discussion we consider two types of arcs: ordinary arcs and enabling arcs. (The DCPN framework also considers inhibitor arcs.) Ordinary arcs are pictured as arrows, enabling arcs are pictured as curves with a little ball at the end (see for example the enabling arc from place p_4 to transition G_2 in Figure 10.5). Enabling arcs are always drawn from a place to a transition.

There are three types of transitions:

Immediate transition I_1, I_2 and I_3 are immediate transitions. An immediate transition I fires as soon as each input place of I, i.e., each place that is connected to an incoming ordinary or enabling arc of I, contains (at least) one token. If an immediate transition fires, then it removes a token from all of its input places that are connected to I via an ordinary arc and it adds a token to all of its output places. (In general DCPN it is possible that with certain probabilities certain output places do not get a token from the transition, but we do not consider this situation here.) Attached to each immediate transition I is a firing function F_I, that probabilistically determines the initial state for the token of each output place. The state value of a token is called the color of the token in Petri net terminology.

Guard transition G_1 till G_5 are guard transitions. The guard G of a guard transition assigns to each of its input places a guard space which is a subset of all possible color values of that place. A guard transition fires with a firing function F as soon as all input places contain a token that is in the guard area of that input place. When the transition fires, each output place gets a token with color determined by the firing function.

Delay transition D_1 is a delay transition. Attached to each delay transition is a jump rate function λ. λ is a function from the token color spaces of its input places to $\mathbb{R}_+$. The transition time of a delay transition with jump rate λ is (stochastically) determined as the transition time of a spontaneous CPDP transition with jump rate λ. The firing function determines the token colors for each of the output places.

We briefly describe what happens if at some time delay transition D_1 fires. By comparing this process with what happens if CPDP *AudioAlert* executes the spontaneous transition, the similarities between CPDP and DCPN can be clearly seen.

If D_1 fires, a token is placed in p_6. This token gets value $k_a = i_a$, $q_a = j_a$ which is probabilistically determined by the firing function of D_1. Now each of the input places of G_1 has a token. This means that G_1 fires immediately after the firing of D_1. The token value of p_1 is (initially) $k_c = 0$, $q_c = 0$. Guard G_1 compares the task (k_a, q_a) with the task (k_c, q_c) and places, via the firing function, the task with the highest priority into place p_1 and places the task with the lowest priority, unless this equals $(0, 0)$, into place ipn_3. If the token of p_1 has been changed by this transition, i.e., if a new task is started, the firing function also puts a token $(\tilde{k}_c, \tilde{q}_c)$, with as value this new task, into place ipn_2. Now the place of DCPN component *CurrentGoal* contains the new task, ipn_2 contains a token with this new task, and ipn_3 is still empty because the old value of p_1 was $(0, 0)$.

Now the input place of G_3 has a token and consequently this guard transition fires immediately. It removes the tokens from ipn_2 and p_2 and places a token in place p_3 with value x_C, where x_C corresponds to the beginning of the task that corresponds to the token removed from ipn_2 (see also the explanation of CPDP

$TaskPerformance$). DCPN component $TaskPerformance$ now started the new task in place p_3.

For the comparison between CPDP and DCPN of the next section, we do not have to explain the rest of this DCPN in detail. The rest can be explained informally as follows.

While the new task is executed in place p_3, transition D_1 can fire again. Suppose this new task from DCPN *AudioAlert* has lower priority than the task that is currently worked on in p_3. Then transition G_1 puts this new task in ipn_3. ipn_1 till ipn_4 are so called interaction petri nets and serve as buffers between the different DCPN components. The new task is stored into memory place p_4 (with token (k_m, q_m)) via transition I_2. If after some time the task of p_3 is completed, then the token from p_3 is removed by transition G_5 and a new token (without value) is placed into p_2. Transition G_2 now 'checks' if there is a to-be-executed-task in the memory. If so, transition G_2 puts this task into p_1, ipn_1 and ipn_4. Then G_4 puts a token in p_3 with value x_C corresponding to the task retrieved from memory (stored now in ipn_1). Transition I_3 removes this new task of p_3 from the memory.

If DCPN *AudioAlert* 'generates' a task with higher priority than the task currently worked on, then this new task is executed via transition G_6 and the task currently worked on is stored into memory via transition I_2.

10.4 Comparison of the CPDP and DCPN models

In this section we compare the CPDP model with the DCPN model. CPDP is an automaton model and DCPN is a Petri net model. Automata and Petri nets are essentially different models and hard to compare.

Because DCPN has been a source of inspiration for the development of CPDP and because in many specific details these two frameworks have some resemblance, we try to do some comparison here, although this is difficult as stated in the paragraph above. We concentrate the comparison around three important issues when it concerns the modelling of complex systems: the modelling process, model checking and analysis.

10.4.1 Modelling

Both CPDP and DCPN have been developed for compositional modelling of PDP-type systems. In modelling the 'pilot flying' system in both the CPDP and DCPN frameworks in this chapter, we have shown that the essential forms of communication are shared by both frameworks:

- Active CPDP transitions that enable passive transitions in other components has its counterpart in DCPN because one component can enable transitions in other components by means of enabling arcs.
- Synchronization of active CPDP transitions, where two components should satisfy certain guards in order to proceed, can be modelled in DCPN as a

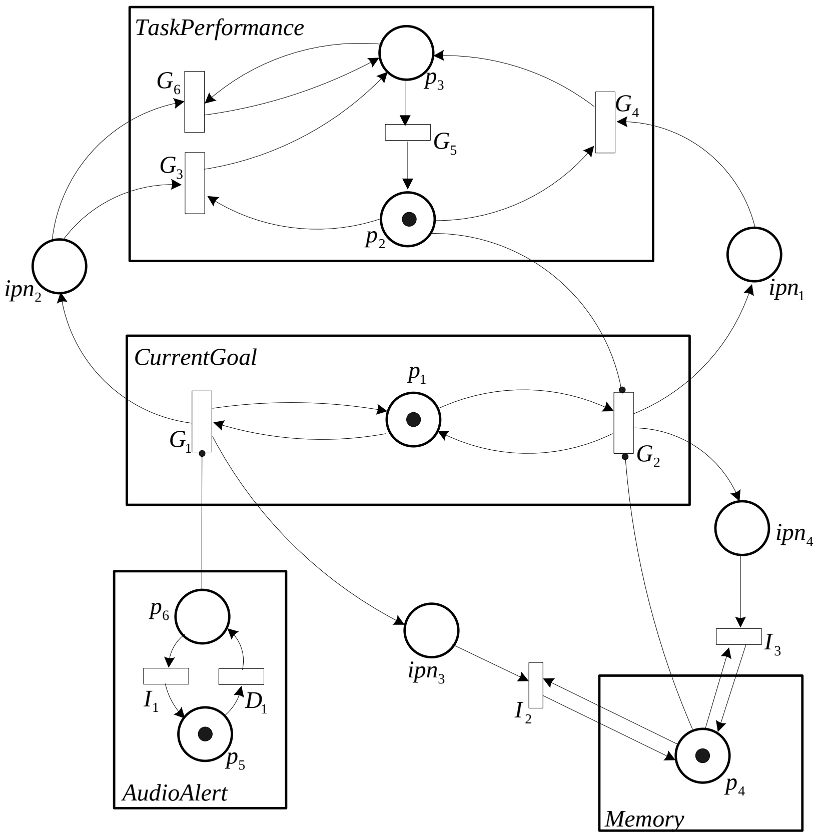

Figure 10.5: DCPN 'pilot flying' model

guard transition that has input places from both components and can guard both components therefore.

- CPDPs value passing is present in DCPN because via an enabling or ordinary arc, a transition of one component can read the tokens of other components.

Interaction/communication possibilities are for CPDPs restricted by the expressiveness of the $|^P_A|$ operator. We have shown that many interesting communication forms can be expressed by $|^P_A|$, but still this remains the restricting factor concerning interaction/communication. For DCPN this is different. There is no formal distinction between components and composition operators. In fact we could say that the composition operators for DCPN components, are formed by a part of the DCPN itself. The DCPN 'composition operator' may be simple, such as a single arrow from one component to another component, it may also be very complex by constructing the operator with a number of extra places, transitions, etc. This means that very complex composition structures can be built in DCPN and this can make compositional modelling easier in some cases. For example, the CPDP *CurrentGoal* had to copy the data received from CPDP *AudioAlert* into an extra pair of variables $(\tilde{k}_a, \tilde{q}_a)$. DCPN transitions have access to different tokens of different components at the same time. This makes it possible to model synchronized transitions in different components directly via one transition and the intermediate steps where one component (*CurrentGoal*) copies values of another component (*AudioAlert*) in order to send them to a third component ($TaskPerformance$) are not necessary then.

It probably depends on the kind of application whether CPDP or DCPN is the most appropriate modelling framework, but it seems that certain interactions can be modelled more directly with DCPN.

10.4.2 Model checking

The more complex the to-be-modelled systems are, the more important it gets to have means to check/test whether the model correctly expresses the behavior of the system. Modelling errors are easily made for complex systems. Similarly, large and complex specifications for programs/systems can easily have unwanted/unseen behavior and it is desirable to have means to test these specifications.

With the CPDP-to-PDP conversion algorithm of Chapter 8.2, we have at the same time given a means to test whether there are transition loops in the CPDP. For certain applications, this might be unwanted behavior that can be tested in this way. Whether similar algorithms can be developed for DCPN is not clear. At least it will be harder because DCPN is a true concurrency model, i.e., the algorithm should keep track of many tokens flowing through the net at the same time. This extra difficulty could be avoided if a clear transition semantics is determined for DCPN. Then the algorithm, which is less direct, could be executed on the semantical level.

Since CPDP can be seen as an extension of discrete event automata and IMC, we can expect that testing theory and model checking theory that exists for discrete event systems and for certain subclasses of IMCs, can be extended to simple-enough subclasses of CPDPs.

10.4.3 Analysis

Because we have an operational semantics for the composition operator $|_A^P|$ such that the composition of CPDPs is expressed by another CPDP, we can analyze a single (composite) CPDP if we want to analyze an interacting network of CPDPs. However, this type of composition suffers from the well-known state space explosion problem, which means that the state space of a composite CPDP grows exponentially with the state spaces of its components. Therefore, composite CPDPs can easily grow too large in order to analyze them.

To avoid this problem, compositional analysis techniques should be developed. One such a technique is bisimulation, which we defined for CPDPs. With bisimulation the state space of a composite CPDP can be reduced in a compositional way. Other compositional analysis techniques might be developed for (subclasses of) CPDPs. It seems that for development of general compositional analysis techniques, it is at least necessary that there is a formal description of what components are and how they interact. For CPDPs we have this formal distinction between components and composition operators. For DCPN, where the composition structure is a part of the DCPN itself, this distinction is not formalized. The advantage of the unrestricted composition structures for DCPN in the context of modelling, turns seemingly in a disadvantage in the context of analysis. In [EKBO04], a number of interaction types for DCPN are defined. Composition for DCPN becomes more formalized in that way and it might be that by restricting the DCPN composition structures by allowing only these interaction types, compositional theories can be developed.

11

Conclusions

In this chapter we summarize the main achievements made in this thesis. Afterwards, we point out directions for future research.

11.1 Achievements

The aim of our project was to develop an automata framework for compositional specification and analysis of stochastic hybrid systems of the PDP (Piecewise Deterministic Markov Process) type. This resulted in the CPDP (Communicating PDP) framework as described in Chapters 7,8 and 9 of this thesis. We developed a composition operator $|^P_A|$, which connects two CPDPs. The result of the composition operator is again a CPDP, whose behavior expresses the combined behavior of the component CPDPs. In other words, we managed to express communication/interaction between CPDPs syntactically in such a way that 1. the composition operation/operator can be defined via structural operational composition rules, and 2. the result of the composition falls again in the class of CPDPs. To make clear why and how this is possible, we distinguish three aspects.

Jump rate representation In the jump rate representation of the PDP process, each state ξ has a jump rate $\lambda(\xi)$ attached to it. By integrating the jump rate along the path of the state of the process, the distribution of the spontaneous jump time is determined. As a result of this representation we get that at each state the distribution of the jump time can be calculated without knowing the past of the process (i.e., the Markov property holds). For composition, this jump rate representation turns out to be very suitable: 1. the jump rate of the combination of two parallel processes turns out to be equal to the sum of jump rates of the individual processes, and 2. because the Markov property holds, a transition in one process from state ξ_1 to state ξ_1', while the other process is at state ξ_2, can be represented as a transition on the product space from (ξ_1, ξ_2) to (ξ_1', ξ_2), i.e., having an additional identity jump at state ξ_2 since the second process in the composition does not influence the behavior.

Borel spaces In an automata-like composition, the state space of the composed system equals the product space of the state spaces of the components. Borel sets/spaces behave nicely on product spaces. First, the (measure theoretic) product space of two Borel spaces is again a Borel space. Second, two independent probability measures on two Borel spaces bring forth a unique probability measure, the so called product probability measure, on the product Borel space.

Transitions Because of the jump-rate representation and the properties of Borel spaces, spontaneous transitions of one component CPDP can, by using identity reset measures, very easily be translated to spontaneous transitions of the composite CPDP. Synchronization of two active/passive transitions can be translated to the composite system as a single transition whose guard equals the 'intersection' of guards of the individual transitions (in case of active/active synchronization), and whose reset map equals the product probability measure of the individual reset maps.

Advantages of the fact that the composition of CPDPs is again a CPDP, are first that we do not need a new syntactical model to express the composition of CPDPs and second analyzing a network of interacting CPDPs can be done by using analysis tools that work for a single CPDP. For small scale systems that still have a compositional structure, this gives a means to find a CPDP representation for the composite system. Because of the equivalence between CPDPs and PDPs, as described in Chapter 8, PDP analysis tools can thus be used to analyze the composite CPDP.

For large scale systems with many components, the state space of the composite CPDP becomes very large since it is exponential in the sizes of the state spaces of the components. In other words, a CPDP grows easily too large to be analyzed. This asks for compositional analysis methods such as abstraction and bisimulation. Abstraction and bisimulation can reduce the state spaces of the components and therewith the state space of the composite system. In Chapter 9 we have shown how strong bisimulation can be defined for CPDPs as a generalization of strong bisimulation for IMCs.

For large scale PDP type systems it might be desirable to have more powerful interaction possibilities than can be expressed by $|^P_A|$ for CPDPs. In the Air Traffic Management system of Chapter 10, certain components need to know information about the continuous state of other components. In order to express such kind of communication, we extended the CPDP framework by using the concept of value passing. Value passing CPDPs are syntactically slightly different from CPDPs, but we have shown that the semantics can still be expressed as a combination of a CFSJS and an NTS. Also composition for value passing CPDPs can be defined in the same structural operational way.

11.2 Directions for future research

11.2.1 Diffusion

With CPDP we can model PDP processes in a compositional way. PDP forms a broad class of stochastic systems, however, it does not include stochastic systems which have diffusion terms. In the SHS (Stochastic Hybrid Systems [BL04, Blo03]) framework, PDP systems with diffusion terms can be modelled. Since diffusion terms can be composed, i.e., the superposition of diffusion terms forms again a diffusion terms, it seems to be possible to add diffusion to the CPDP framework, such that composition can still be defined. Extending the CPDP framework with diffusion is interesting since the class of stochastic hybrid systems that can be modelled then becomes very large.

11.2.2 Compositional modelling power

Interaction between CPDPs happens through synchronized transitions. This means that interaction is expressed through the discrete part, i.e., the transitions which are executed at time instants from a discrete set, of the CPDP. Another direction of extending the CPDP framework is to allow interaction through continuous variables (as in [JSvdS03] for example). Then, hybrid control systems, where the controller component continuously reads and controls the process component, could be specified compositionally in this extended CPDP framework. Although this extension would give powerful and expressive composition, we think that compositional analysis becomes much more difficult, since the moments of interaction then fall in a continuous set instead of a discrete set as is the case for CPDPs now.

11.2.3 Model checking

Model checking has proven to be a very effective tool for analyzing discrete event processes. Also for certain subclasses of IMCs an effective model checking theory has been developed [HKMKS00]. CPDPs are a generalization of IMCs and it is to be expected that model checking theory (including developing a suitable logic) for IMC can be generalized to certain 'simple enough' subclasses of CPDPs. We expect that for the class of CPDPs that is discussed in Section 9.3.3, and for which the bisimulation algorithm is decidable, a model checking theory can be developed.

11.2.4 Compositional analysis

With the CPDP framework we can specify complex PDP type systems in a compositional way. Concerning the analysis part, we can transfer small scale composite systems to the realm of PDP because of the equivalence result between CPDPs and PDPs. However, for large scale system analysis, compositional methods are needed.

One such method is bisimulation. We have shown that for a certain subclass of CPDPs, bisimulations can be calculated algorithmically and the computation of the algorithm is decidable. Since bisimulation has proven to be a very effective tool, it might be worthwile to investigate whether other, broader, subclasses of CPDPs allow decidable computation of (maximal) bisimulations.

Other compositional analysis tools are based on assume-guarantee rules [AH97, dAHJ01]. Here, under assumptions on the components, the composition of the components is guaranteed to behave in a certain way. These tools are developed mainly for discrete event processes. However, the essence of composition for CPDPs, which lies in the synchronization of transitions, is the same as for discrete event concurrent processes and therefore we can hope that these theories allow some generalization to (subclasses of) CPDPs. Connected to these assume-guarantee theories are certain transformation tools that transform components in a composition such that the behavior of the composite system does not change while the composition structure of the transformed system is less complex and a less complex composition structure allows better/easier compositional analysis [Bol96]. Again, these theories could possibly be generalized to the level of CPDPs.

11.2.5 Abstraction and weak bisimulation

By abstracting (internalizing) actions of IMC transitions, sequences of internal transitions can be replaced by a single transition without changing the system up to weak bisimilarity (see Chapter 5). Strong bisimilarity for IMC can naturally be generalized to strong bisimilarity for CPDPs (as was shown in Chapter 9). For weak bisimilarity this can not so easily be done. Defining internal τ-transitions for CPDPs forms no problem. However, defining weak bisimulation for CPDPs can seemingly not be done in a simple and elegant way (as it is done for IMCs) because of the probabilistic reset maps that are involved in a chain of transitions. Developing a theory for weak bisimulation for CPDPs would therefore take quite some more efforts than is the case for strong bisimulation. However, for analysis of large scale systems it is generally believed that a theory of abstraction is essential. This belief seems to hold even stronger for (stochastic) hybrid systems and makes abstraction/weak bisimulation for CPDPs therefore a natural and interesting research direction.

11.2.6 Stochastic analysis

Effective analysis tools have been developed for PDPs. Since the components of composite CPDPs are 'PDP-like', a research direction would be to investigate whether these stochastic analysis tools can be applied to CPDP components as well and how such results for the components can induce results for the composite CPDPs. In other words, can these stochastic analysis tools be used for CPDPs in a compositional way? For example, in [Dav93], formulas are given such that expectations, concerning a PDP, can be calculated in an effective way. Can these formula's play a role in a compositional theory? In other words, can we calculate expectations on the level of components and derive results from that concerning expectations of the composite system?

Bibliography

[ACH$^+$95] R. Alur, C. Coucoubetis, N. Halbwachs, T. Henzinger, P. Ho, X. Nicolin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[AD94] R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AH97] R. Alur and T. A. Henzinger. Modularity for timed and hybrid systems. In *Proceedings of the Eighth International Conference on Concurrency Theory (CONCUR)*, pages 74–88, 1997.

[AHLP00] R. Alur, T.A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. *Proceedings of the IEEE*, 88(7):971–984, July 2000.

[BB87] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Comp. Networks and ISDN Systems*, 14:25–59, 1987.

[BKU05] E. Brinksma, T. Krilavičius, and Y.S. Usenko. Process algebraic approach to hybrid systems. In *Proc. of 16th IFAC World Congress*, 2005.

[BL04] M.L. Bujorianu and L. Lygeros. General stochastic hybrid systems: Modelling and optimal control. In *Proc. of 43rd Conference in Decision and Control*, 2004.

[BLB05] M.L. Bujorianu, J. Lygeros, and M.C. Bujorianu. Different approaches on bisimulation for stochastic hybrid systems. In L. Thiele and M. Morari, editors, *Hybrid Systems: Computation and Control*, number 3414 in LNCS, pages 198–214. Springer-Verlag, Berlin, 2005.

[Blo03] H. A. P. Blom. Stochastic hybrid processes with hybrid jumps. In *Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03*, pages 361–366, 2003.

[Bol96] T. Bolognesi. Regrouping parallel processes. *Formal Methods in System Design 9*, pages 263–302, 1996.

[BSA04] M. Bernadsky, R. Sharykin, and R. Alur. Structured modelling of concurrent stochastic hybrid systems. In *Joint Conference on Formal Modeling and Analysis of Timed Systems (FORMATS) and Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT)*, Grenoble, France, September 22-24 2004.

[Buj04] M.L. Bujorianu. Extended stochastic hybrid systems and their reachability problem. In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 234–249. Springer, 2004.

[CL99] C. G. Cassandras and S. Lafortune. *Introduction to discrete event systems.* Kluwer Academic Publishers, 1999.

[CLSV05] L. Cheung, N. A. Lynch, R. Segala, and F. W. Vaandrager. Switched probabilistic i/o automata. In *Theoretical Aspects of Computing*, volume 3407 of *Lecture Notes in Computer Science*, pages 494–510. Springer, 2005.

[CS02] S. Cattani and R. Segala. Decision algorithms for probabilistic bisimulation. In *13th International Conference on Concurrency Theory*, volume 2421 of *Lecture Notes in Computer Science*, pages 371–385. Springer, 2002.

[Cui04] P.J.L. Cuijpers. *Hybrid Process Algebra.* PhD thesis, Technical University Eindhoven, 2004.

[D'A97] P. R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems.* PhD thesis, University of Twente, 1997.

[DA98] R. David and H. Alla. Continuous and hybrid petri nets. *Journal of Circuits, Systems and Computers*, 8(1):159–188, 1998.

[dAHJ01] L. de Alfaro, T. A. Henzinger, and R. Jhala. Compositional methods for probabilistic systems. *Lecture Notes in Computer Science*, 2154:351+, 2001.

[Dav84] M. H. A. Davis. Piecewise Deterministic Markov Processes: a general class of non-diffusion stochastic models. *Journal Royal Statistical Soc. (B)*, 46:353–388, 1984.

[Dav93] M. H. A. Davis. *Markov Models and Optimization.* Chapman & Hall, London, 1993.

[DHS04] S. Derisavi, H. Hermanns, and W. H. Sanders. Optimal state-space lumping in Markov chains. *Information Processing Letters*, 87(6):309–315, 2004.

[EB03] M. H. C. Everdij and H. A. P. Blom. Petri-nets and hybrid-state Markov processes in a power-hierarchy of dependability models. In *Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03*, pages 355–360, 2003.

[EKBO04] M.H.C. Everdij, M.B. Klompstra, H.A.P. Blom, and B. Klein Obbink. Compositional specification of a multi-agent system by dynamically coloured petri nets. Technical report, National Aerospace Laboratory NLR, 2004. Public Deliverable D9.2 of HYBRIDGE project.

[GAM93] M.K. Ghosh, A. Arapostathis, and S.I. Marcus. Optimal control of switching diffusions with application to flexible manufacturing systems. *SIAM Journal on Control Optimization*, 31(5):1183–1204, September 1993.

[GAM97] M.K. Ghosh, A. Arapostathis, and S.I. Marcus. Ergodic control of switching diffusions. *SIAM Journal on Control Optimization*, 35(6):1952–1988, November 1997.

[GB04] M. K. Ghosh and A. Bagchi. Modelling stochastic hybrid systems. In *Preprint of paper in IFIP TC-7 book*, 2004.

[Gro91] J. F. Groote. *Process Algebra and Structured Operational Semantics.* PhD thesis, University of Amsterdam, 1991.

[Hen88] M. Hennesy. *Algebraic Theory of Processes.* MIT Press, 1988.

[Hen96] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS)*, pages 265–292. IEEE Computer Society Press, 1996.

[Her02] H. Hermanns. *Interactive Markov Chains*, volume 2428 of *Lecture Notes in Computer Science.* Springer, 2002.

[HKMKS00] H. Hermanns, J.-P Katoen, J. Meyer-Kayser, and M. Siegle. A markov chain model checker. In *TACAS/ETAPS 2000*, volume 1785 of *Lecture Notes in Computer Science*, pages 347–362. Springer, 2000.

[HLS00] J. Hu, J. Lygeros, and S.S. Sastry. Towards a theory of stochastic hybrid systems. In Nancy Lynch and Bruce H. Krogh, editors, *Hybrid Systems: Computation and Control*, number 1790 in LNCS, pages 160–173. Springer-Verlag, Berlin, 2000.

[Hoa85] C.A.R. Hoare. *Communicating Sequential Processes.* Prentice-Hall, 1985.

[JSvdS03] A. A. Julius, S. N. Strubbe, and A. J. van der Schaft. Control of hybrid behavioral automata by interconnection. In *Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03*, pages 135–140, 2003.

[Jul05] A.A. Julius. *On Interconnection and Equivalence of Continuous and Discrete Systems.* PhD thesis, University of Twente, 2005.

[Kin93] J.F.C. Kingman. *Poisson Processes.* Oxford Clarendon Press, 1993.

[KS60] J. G. Kemeny and J. L. Snell. *Finite Markov Chains.* Van Nostrand, New York, 1960.

[Lan92] R. Langerak. *Transformations and Semantics for LOTOS.* PhD thesis, Universiteit Twente, 1992.

[LL92] M. Haj-Hussein L. Logrippo, M. Faci. An introduction to lotos: Learning by examples. *Comp. Networks and ISDN Systems*, 23(5):325–342, 1992.

[LPS00] G. Lafferriere, G.J. Pappas, and S.S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, March 2000.

[LS91] K. G. Larsen and A. Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94:1–28, 1991.

[LSV01] N. A. Lynch, R. Segala, and F. W. Vaandrager. Hybrid I/O automata revisited. *Lecture Notes in Computer Science*, 2034:403–414, 2001.

[LSV03] N. A. Lynch, R. Segala, and F. W. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, 2003.

[LT88] N. A. Lynch and M. R. Tuttle. An introduction to input/output automata. *CWI Quarterly*, 2(3):219–246, 1988.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Obb05] B. Klein Obbink. Description of advanced operation: Free flight. Technical report, National Aerospace Laboratory NLR, 2005. Public Deliverable D9.1 of HYBRIDGE project.

[Pap03] G.J. Pappas. Bisimilar linear systems. *Automatica*, 39:2035–2047, 2003.

[Par67] K.R. Parthasarathy. *Probability Measures on Metric Spaces.* Academic Press, 1967.

[PBL03] G. Pola, M. L. Bujorianu, and J. Lygeros. Stochastic hybrid models: an overview. In *Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03*, pages 52–57, 2003.

[Plo81] G. D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI FN-19, Aarhus University, Comp. Sci. Dept., 1981.

[Pra91] K. Prasad. A Calculus of Broadcasting Systems. In *Proc. 16th Colloquium on Trees in Algebra and Programming*, volume 493, pages 338–358, 1991.

[PW98] J.W. Polderman and J.C. Willems. *Introduction to Mathematical System Theory : A Behavioral Approach.* Springer, New York, 1998.

[RW89] P.J. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[SJvdS03] S. N. Strubbe, A. A. Julius, and A. J. van der Schaft. Communicating Piecewise Deterministic Markov Processes. In *Preprints Conference on Analysis and Design of Hybrid Systems ADHS 03*, pages 349–354, 2003.

[SL05] S. N. Strubbe and R. Langerak. A composition operator with active and passive actions. Accepted for 25th IFIP WG 6.1 International Conference on Formal Techniques for Networked and Distributed Systems, Taipei, 2005.

[SvdS05a] S. N. Strubbe and A. J. van der Schaft. Algorithmic bisimulation for Communicating Piecewise Deterministic Markov Processes. Accepted for Conf. Decision and Control, Seville, 2005, 2005.

[SvdS05b] S. N. Strubbe and A. J. van der Schaft. Application of communicating pdps to air traffic management. Submitted to American Control Conference 2006, 2005.

[SvdS05c] S. N. Strubbe and A. J. van der Schaft. Bisimulation for Communicating Piecewise Deterministic Markov Processes (cpdps). In *Hybrid Systems: Computation and Control*, volume 3414 of *Lecture Notes in Computer Science*, pages 623–639. Springer, 2005.

[SvdS05d] S. N. Strubbe and A. J. van der Schaft. Stochastic equivalence of CPDP-automata and Piecewise Deterministic Markov Processes. In *Proc. of IFAC World Congress 2005*, 2005.

[SvdS05e] S. N. Strubbe and A. J. van der Schaft. Stochastic semantics and value-passing for communicating piecewise deterministic markov processes. Accepted for Conf. Decision and Control, Seville, 2005, 2005.

[vdS04a] A.J. van der Schaft. Bisimulation of dynamical systems. In *Hybrid Systems: Computation and Control*, volume 2993 of *Lecture Notes in Computer Science*, pages 555–569. Springer, 2004.

[vdS04b] A.J. van der Schaft. Equivalence of dynamical systems by bisimulation. *IEEE Transactions on Automatic Control*, 49:2160–2172, 2004.

[vdS04c] A.J. van der Schaft. Equivalence of hybrid dynamical systems. In *Proceedings of the 16th international symposium on Mathematical Theory of Networks and Systems*, 2004.

[vdSS00] A. J. van der Schaft and J. M. Schumacher. *An Introduction to Hybrid Dynamical Systems.* Springer London, 2000.

[Wil91] D. Williams. *Probability with Martingales.* Cambridge University Press, 1991.

[Wil97] J. C. Willems. On interconnections, control and feedback. *IEEE Transactions on Automatic Control*, 42:326–339, 1997.

Index

Summary

Since nowadays hybrid systems, like software systems, telecommunication systems or air traffic management systems, often have a very complex structure, modelling these systems without compositional modelling methods is nearly impossible. In this thesis the modelling framework CPDP (Communicating PDP or Communicating Piecewise Deterministic Markov Process) is presented. CPDP is a compositional framework for the modelling of stochastic hybrid systems of the PDP-type.

A parallel composition operator, denoted by $|^P_A|$, is presented. With this operator multiple CPDPs can be connected to form a single complex CPDP. Because a distinction is made between *active* and *passive* transitions, multiple types of communication/interaction between CPDPs can be expressed through $|^P_A|$, among which *blocking-interaction*, where one component can block the execution of transitions in other components, and *broadcasting-interaction*, where one component can observe signals sent by other components. By using CPDPs together with the operator $|^P_A|$, complex stochastic hybrid systems can be modelled in a stepwise way.

An extended framework, called value-passing CPDP, is presented. In this framework richer interaction structures can be built. The main extension is that values concerning the continuous variables can now be communicated from one value-passing CPDP to the other.

It is shown that by using schedulers, which resolve the non-determinism of a CPDP, a scheduled CPDP can be transformed into a PDP without changing the stochastic behavior. With this result it becomes possible that PDP analysis techniques are applied to complex CPDP processes.

The concept of bisimulation is defined for CPDPs. It is shown that bisimilar CPDPs have equivalent stochastic behavior. An algorithm is presented, which can for CPDPs satisfying certain conditions, automatically determine a state reduced bisimilar CPDP. Since the state reduced version has equivalent stochastic behavior, analysis can be applied to the state reduced version instead of to the more complex original CPDP.

Samenvatting

Hybride systemen zoals softwaresystemen, telecommunicatiesystemen en air-traffic-managementsystemen, hebben tegenwoordig vaak een zeer complexe structuur. Zonder compositionele modelleertechnieken wordt het modelleren van deze systemen haast onmogelijk. In dit proefschrift wordt het modelleringskader CPDP (Communicating PDP or Communicating Piecewise Deterministic Markov Process) gepresenteerd. CPDP vormt een compositioneel kader voor het modelleren van stochastische hybride systemen van het PDP type.

Een parallelle compositie-operator $|{}^{P}_{A}|$ wordt gepresenteerd. Met deze operator kunnen meerdere CPDPs aan elkaar gekoppeld worden, met als resultaat een nieuwe complexe CPDP. Omdat een distinctie wordt gemaakt tussen *actieve* en *passieve* transities, kunnen verscheidene vormen van communicatie tussen CPDPs worden gemodelleerd, waaronder *blocking-interaction*, waar één component de executie van transities van andere componenten kan blokkeren, en *broadcasting-interaction*, waar één component signalen van andere componenten kan observeren. Door CPDPs samen met de operator $|{}^{P}_{A}|$ te gebruiken, kunnen complexe stochastische hybride systemen op een stapsgewijze manier gemodelleerd worden.

Een uitgebreide versie van het CPDP kader, genoemd value-passing CPDP, wordt gepresenteerd. In deze versie kunnen rijkere interactiestructuren worden gemodelleerd. De uitbreiding komt er voornamelijk op neer dat waarden van de continue variabelen kunnen worden gecommuniceerd van de ene value-passing CPDP naar de andere.

Er wordt aangtoond dat door *schedulers*, die het non-determinisme van een CPDP oplossen, te gebruiken, een *scheduled* CPDP kan worden getransformeerd tot een PDP, zonder het stochastische gedrag te veranderen. Door dit resultaat wordt het mogelijk om analyse technieken voor PDPs toe te passen op complexe CPDP processen.

De notie van bisimulatie wordt gedefinieerd voor CPDPs. Er wordt aangetoond dat bisimulaire CPDPs equivalent stochastisch gedrag vertonen. Een algoritme wordt gegeven, waarmee voor CPDPs die aan bepaalde voorwaarden voldoen, automatisch een toestandsgereduceerde bisimulaire CPDP kan worden bepaald. Omdat deze toestandsgereduceerde versie equivalent stochastisch gedrag vertoont, kan analyse toegepast worden op deze toestandsgereduceerde versie in plaats van op de meer complexe originele CPDP.

Dankwoord

In afsluiting van dit proefschrift wil ik graag een aantal mensen bedanken die of voor de ontwikkeling van dit proefschrift of voor de ontwikkeling van mijzelf in de jaren dat ik aan dit project gewerkt heb, belangrijke personen zijn geweest.

Als eerste wil ik mijn begeleider en promotor Arjan van der Schaft bedanken. In de besprekingen die we hebben gehad over mijn onderzoek reageerde hij (bijna) altijd enthousiast op de ideeën die ik had. Redelijk vaak is het voorgekomen dat, terwijl mijn motivatie gedaald en mijn onzekerheid over mijn ideeën gestegen was, ik weer een motivatie-impuls kreeg vanwege de positieve houding en reacties van Arjan. Hierdoor, en vanwege het feit dat hij nooit probeert zijn ideeën tegen mijn zin/smaak op te dringen, heb ik vrijheid en motivatie ervaren om dat te doen wat geresulteerd heeft in dit proefschrift.

Ook wil ik de leden van de promotie commissie, te weten prof.dr. Bagchi, dr.ir. Blom, dr.ir. Vellekoop, prof.dr. Katoen, dr.ir. Langerak, prof.dr. Lygeros, prof.dr.ir. Mouthaan, prof.dr. van der Schaft en prof.dr. van Schuppen, bedanken voor het lezen en becommentariseren van het proefschrift. Hun correcties en suggesties hebben bijgedragen tot een verbetering van het proefschrift.

Ik bedank de Europese Commissie en het Hybridge project voor het bieden van de mogelijkheid om mijn project uit te voeren.

In de eerste jaren van mijn onderzoek was er een clubje genaamd miniCASH, dat bestond uit in-hybride-systemen-geïnteresserde wiskundigen en informatici. Dit waren Agung Julius, Rom Langerak, Tomas Krilavicius en mijzelf. Later heeft ook Raj Kakumani zich bij onze club gevoegd. Ik wil hen bedanken voor de leuke en interessante lunchbijeenkomsten en discussies die we hebben gehad. Ik heb hier veel van geleerd, vooral omdat het voor mij de toegang tot de theoretische informatica wereld heeft mogelijk gemaakt. In het bijzonder wil ik Rom bedanken voor zijn grote vriendelijkheid, voor de leuke samenwerking die we hebben gehad en voor zijn goede ideeën die hun uitwerking niet gemist hebben in het proefschrift. Ook wil ik hier in het bijzonder Agung bedanken. Hij is jarenlang een leuke, vriendelijke en humorvolle *office-mate* geweest en we hebben samen goede en belangrijke inhoudelijke discussies gehad. Ook zijn invloed is terug te vinden in het proefschrift.

Ik wil de AIO's en de leden van de vakgroep SSB bedanken voor de prettige en inspirerende omgeving die ze hebben gevormd en waarin ik een tijd heb mogen verkeren. In het bijzonder bedank ik Agoes, Ram, Javier, Jaroslav, Vishy en Agung voor de goede momenten die we hebben beleefd. Verder wil ik Marja bedanken voor haar vriendelijkheid en voor de administratieve ondersteuning.

Christiaan, mijn broer, wil ik bedanken voor het realiseren en meehelpen bedenken van de omslag van het proefschrift, en ook voor zijn grote behulpzaamheid. Hij stond altijd direct voor me klaar als ik weer eens problemen had met de computer.

De afgelopen jaren zijn Arnaud de Callafon en Dirk-Jan van Vliet mijn beste vrienden geweest. Met grote vreugde kijk ik terug op de vele mooie momenten die we hebben beleefd en de 'dingen' die we hebben gedeeld. Hun invloed op mijn ontwikkeling van de afgelopen jaren is zeer groot en zo is ook mijn dank voor onze vriendschap. Ik hoop dat deze nog lang en met een zelfde diepte zal mogen voortduren.

Een andere sleutelpersoon in mijn leven is Mark Verwoerd. Vanaf de middelbare school tot en met de studententijd waren wij zo'n beetje onafscheidelijk. Zijn warme persoonlijkheid en de vele dingen waarin we elkaar herkennen zijn zeer inspirerend en invloedrijk geweest voor mij. Ik ben dankbaar voor onze vriendschap en kameraadschap, en ook al wonen we niet meer in hetzelfde huis, niet meer in dezelfde stad en niet meer in hetzelfde land, ik hoop en vertrouw erop dat deze vriendschap zal voortduren.

Verder wil ik al mijn andere vrienden in de Ronde Venen, Utrecht, Amsterdam, etc., bedanken voor de vriendschap en de wijsheden die ze mij de afgelopen jaren gegeven hebben.

Tenslotte wil ik mijn familie in Wilnis bedanken voor de grote warmte en liefde die ze mij mijn hele leven hebben gegeven en die ik nog altijd ervaar als ik 'naar huis kom'.

Enschede, November 2005

Stefan Strubbe