Equivalence Notions for Concurrent Systems and Refinement of Actions

(Extended Abstract)

Rob van Glabbeek
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Ursula Goltz
Gesellschaft für Mathematik und Datenverarbeitung
Postfach 1240, D-5205 Sankt Augustin 1, German Federal Republic

Abstract We investigate equivalence notions for concurrent systems. We consider "linear time" approaches where the system behaviour is characterised as the set of possible runs as well as "branching time" approaches where the conflict structure of systems is taken into account. We show that the usual interleaving equivalences, and also the equivalences based on *steps* (multisets of concurrently executed actions) are not preserved by refinement of atomic actions. We prove that "linear time" partial order semantics, where causality in runs is explicit, is invariant under refinement. Finally, we consider various bisimulation equivalences based on partial orders and show that the strongest one of them is preserved by refinement whereas the others are not.

Introduction

A large body of research is devoted to equivalence notions for concurrent systems. Most of the equivalence notions currently being considered are based on a semantics where concurrency is modelled by arbitrary interleaving of atomic actions. In [Pratt] and in [CDP] it is pointed out that this approach has a severe drawback. It leads to complications when changing the level of atomicity of events; "... we would like a theory of processes to be just as usable for events having a duration or structure, where a single event can be atomic from one point of view and compound from another" ([Pratt]). In [CDP], an example is given, showing that the usual interleaving equivalence is not invariant under refinement of actions when this is simply modelled by textual replacement. Both [Pratt] and [CDP] claim that modelling concurrency by expressing causal dependencies explicitly using partial orders could help to solve this problem. However, the two systems considered in [CDP] can already be distinguished by considering interleavings of "steps" (multisets of concurrently executable actions). So their example does not show that it is indeed necessary to consider partially ordered executions. Furthermore, their proof of the claim that partial order equivalence is preserved by refinement is only valid for "linear time" partial order semantics, where the set of all possible executions of a system is considered, without taking into account where conflicts are resolved. This is also the model considered by Pratt.

In this paper, we will consider various equivalence notions based on steps and on partial orders. We will discuss "linear time" semantics, but we will also take the conflict structure of systems into account by considering various forms of bisimulation ("branching time" semantics). We will show that the known equivalences based on steps are not invariant under action refinement. We will rephrase in our framework the proof of [CDP], showing that "linear time" partial order semantics is indeed robust against changing the level of atomicity. Then we consider several equivalence notions based on "branching time" partial order semantics. We give examples, showing that pomset bisimulation equivalence [BC] and also the NMS partial ordering equivalence suggested in [DDM], are not preserved by refinement of atomic actions. An equivalence notion for Petri nets which coincides

with the notion of NMS partial ordering equivalence was suggested in [Devillers] where the refinement problem has also been discussed. We also show that NMS partial ordering equivalence does not imply pomset bisimulation (and vice versa); hence these notions are incomparable. Finally we show that a stronger equivalence notion, first suggested in [TRH] under the name BS-bisimulation, is indeed preserved by refinement. This equivalence does respect pomset bisimulation.

We do not intend to advocate any particular equivalence notion here, the purpose of this investigation is to find out about the consequences of the different approaches. There will certainly be a tradeoff between simplicity and distinguishing power. We just want to illustrate that the appropriate notion has to be chosen carefully with regard to the questions considered.

1 Concurrent systems

In this paper we consider systems that are capable of performing actions from a given set Act of action names. As our model for this kind of systems we have chosen event structures here (prime event structures with a binary conflict relation as introduced in [NPW]); we could have chosen other models like Petri nets or behaviour structures [TRH] as well. We will frequently give CCSP–expressions for our examples, to make them easier to understand: $+$ will denote choice (as in CCS), $|$ will denote parallel composition (without communication), $a.P$ performs action a and then behaves like P and nil denotes the empty process; a abbreviates $a.nil$ and the unary prefixing operator binds stronger than the binary ones, as usual. Dots in expressions $a.P$ will be omitted. However, this notation is only used for intuition; formally our results are established for event structures. We will not distinguish external and internal actions here, we do not consider abstraction by hiding of actions.

Definition A *(labelled) event structure* (over an alphabet Act) is a 4–tuple

$\mathcal{E} = (E, \leq, \#, l)$ where

- E is a set of *events*,
- $\leq \subseteq E \times E$ is a partial order (the *causality relation*) satisfying the *principle of finite causes*: $\forall e \in E : \{e' \in E | e' \leq e\}$ is finite.
- $\# \subseteq E \times E$ is an irreflexive, symmetric relation (the *conflict relation*) satisfying the principle of *conflict heredity*: $\forall e_1, e_2, e_3 \in E : e_1 \leq e_2 \wedge e_1 \# e_3 \Rightarrow e_2 \# e_3$.
- $l : E \rightarrow Act$ is a *labelling function*.

The components of an event structure $\mathcal{E}$ will be denoted by $E_\mathcal{E}, \leq_\mathcal{E}, \#_\mathcal{E}$ and $l_\mathcal{E}$. If clear from the context, the index $\mathcal{E}$ will be omitted. As usual, we write $e < e'$ for $e \leq e' \wedge e \neq e'$, etc.

An event structure represents a concurrent system in the following way: action names $a \in Act$ represent actions the system might perform, an event $e \in E$ labelled with a represents an occurrence of a during a possible run of the system, $e' < e$ means that e' is a prerequisite for e and $e \# e'$ means that e and e' cannot happen both in the same run. We will later assume that in a finite period the system performs only finitely many actions.

Throughout the paper, we assume a fixed set Act of action names as labelling set. Let $\mathbb{E}$ denote the domain of event structures labelled over Act.

Causal independence (*concurrency*) of events is expressed by the derived relation $co \subseteq E \times E$: $e \; co \; e'$ iff $\neg(e < e' \vee e' < e \vee e \# e')$. By definition, $<, >, \#$ and co form a partition of $E \times E$.

An event structure $\mathcal{E}$ is *finite* if $E_\mathcal{E}$ is finite; $\mathcal{E}$ is *conflict-free* if $\#_\mathcal{E} = \emptyset$.
O denotes the empty event structure $(\emptyset, \emptyset, \emptyset, \emptyset)$.

For $X \subseteq E_\mathcal{E}$, the *restriction of* $\mathcal{E}$ *to* X is defined as $\mathcal{E}\lceil X = (X, \leq \cap (X \times X), \# \cap (X \times X), l\lceil X)$.

Two event structures $\mathcal{E}$ and $\mathcal{F}$ are *isomorphic* ($\mathcal{E} \cong \mathcal{F}$) iff there exists a bijection between their sets of events preserving $\leq, \#$ and labelling. Generally, we will not distinguish isomorphic event structures.

Isomorphism classes of conflict-free event structures are called *pomsets* [Pratt]. They have also been considered under the name *partial words* in [Grabowski]. Pomsets generated by certain subsets of events may be considered as possible "executions" of the system represented by the event structure. The partial order between action occurrences then represents causal dependencies in the execution. Subsets of events representing executions (called *configurations*) have to be conflict-free; furthermore they must be left-closed with respect to $\leq$ (all prerequisites for any event occurring in the "execution" must also occur). We will consider only finite executions when comparing the behaviour of systems. It is assumed that in a finite period only finitely many actions are performed.

Definition

i. A subset $X \subseteq E_\mathcal{E}$ of events in an event structure $\mathcal{E}$ is *left-closed* in $\mathcal{E}$ iff, for all $e, e' \in E_\mathcal{E}$, $e \in X \wedge e' \leq e \Rightarrow e' \in X$.
X is *conflict-free* in $\mathcal{E}$ iff $\mathcal{E}\lceil X$ is conflict-free.

ii. A subset $C \subseteq E_\mathcal{E}$ will be called a *(finite) configuration* of an event structure $\mathcal{E}$ iff C is finite[1], left-closed and conflict-free in $\mathcal{E}$. $\mathcal{C}(\mathcal{E})$ denotes the set of all configurations of $\mathcal{E}$.
A configuration $C \in \mathcal{C}(\mathcal{E})$ is called *complete* iff C is a maximal conflict-free set of events in $\mathcal{E}$.

Configurations may be considered as possible states of the system; they determine the remaining behaviour of the system as being the set of all events which have not yet occurred and are not excluded because of conflicts.

Example 1.1

Let us consider the event structure $\mathcal{E}$ corresponding to the expression $a|b + ab$.

In graphical representations, only immediate conflicts - not the inherited conflicts - are indicated. The $\leq$-relation is represented by arcs, omitting those derivable by transitivity. Furthermore, instead of events only their labels are displayed; if a label occurs twice it represents two different events. Thus these pictures determine event structures only up to isomorphism.

Following these conventions, $\mathcal{E}$ is represented as $\begin{matrix} a \\ \# \\ a \rightarrow b \\ \# \\ b \end{matrix}$. The possible executions of $\mathcal{E}$ are represented by the pomsets O (the empty pomset), a , b , $\begin{matrix} a \\ b \end{matrix}$ and $a \rightarrow b$. The latter two correspond to complete configurations.

[1] [Winskel] does not require configurations to be finite.

We may now ask which actions may occur in a configuration and which configuration is then obtained.

Definition Let $\mathcal{E}$ be an event structure,

i. $C \longrightarrow_{\mathcal{E}} C'$ if $C, C' \in \mathcal{C}(\mathcal{E})$ and $C \subseteq C'$.

ii. $C \stackrel{a}{\rightarrow} C'$ iff $a \in Act$, $C \longrightarrow_{\mathcal{E}} C'$ and $C' \setminus C = \{e\}$ with $l(e) = a$.

Note that $C \longrightarrow_{\mathcal{E}} C'$ implies that $\mathcal{E}\lceil(C' \setminus C)$ is finite and conflict-free.

Here $C \xrightarrow{a} C'$ says that if $\mathcal{E}$ is in the state represented by C, then it may perform an action a and reach a state represented by C'. Likewise, $C \rightarrow_{\mathcal{E}} C'$ says that $\mathcal{E}$ may evolve from C to C'.

Considering transitions $C \xrightarrow{a} C'$ only, one can define the usual *interleaving semantics*. The simplest form is that of comparing just the possible sequences of action occurrences.

Definition $w = a_1 \cdots a_n \in Act^*$ is a *(sequential) trace* of an event structure $\mathcal{E}$ iff there exist configurations $C_o, \cdots, C_n$ of $\mathcal{E}$ such that $C_o = \emptyset$ and $C_{i-1} \stackrel{a_i}{\rightarrow} C_i$ $(i = 1, \cdots, n)$.
SeqTraces $(\mathcal{E})$ denotes the set of all sequential traces of an event structure $\mathcal{E}$.
Two event structures $\mathcal{E}, \mathcal{F}$ are called *interleaving trace equivalent* $(\mathcal{E} \approx_{it} \mathcal{F})$ iff *SeqTraces* $(\mathcal{E})$ = *SeqTraces* $(\mathcal{F})$.

With the concept of labelled transition systems, we obtain a stronger equivalence notion based on the idea of bisimulation [Park, Milner]. For example, the systems $a(b+c)$ and $ab+ac$ have the same traces but are distinguished by bisimulation equivalence.

Definition Let $\mathcal{E}, \mathcal{F}$ be event structures.

A relation $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F})$ is called an *interleaving bisimulation between* $\mathcal{E}$ *and* $\mathcal{F}$ iff
$(\emptyset, \emptyset) \in R$ and if $(C, D) \in R$ then
- $C \stackrel{a}{\rightarrow} C' \Rightarrow \exists D'$ with $D \stackrel{a}{\rightarrow} D'$ and $(C', D') \in R$,
- $D \stackrel{a}{\rightarrow} D' \Rightarrow \exists C'$ with $C \stackrel{a}{\rightarrow} C'$ and $(C', D') \in R$.

$\mathcal{E}$ and $\mathcal{F}$ are *interleaving bisimulation equivalent* $(\mathcal{E} \approx_{ib} \mathcal{F})$ iff there exists an interleaving bisimulation between $\mathcal{E}$ and $\mathcal{F}$.

Clearly, $\mathcal{E} \approx_{ib} \mathcal{F}$ implies $\mathcal{E} \approx_{it} \mathcal{F}$.

In the following section, after introducing the notion of refinement, we will recall the example of [CDP], showing that both $\approx_{it}$ and $\approx_{ib}$ are not preserved by refinement.

2 Refinement of actions

In [CDP] it is shown that equivalence notions based on interleaving are not preserved when replacing an action in a system by a sequence of two actions. We consider here a more general version of this operation: replacing actions by "computations", finite conflict-free event structures. Replacing actions by infinite computations could in general invalidate the principle of finite causes for event structures. Replacing actions by event structures containing conflicts would require a more sophisticated notion of refinement or, alternatively, a more general form of event structures where the axiom of conflict heredity is dropped, e.g. flow event structures [BCa]. (Consider $\mathcal{E} = \begin{smallmatrix} a \\ \downarrow \\ b \end{smallmatrix}$. Replacing a by $c \# d$ violates the axiom of conflict heredity, as long as the event labelled by b is not duplicated in some

way.) For sake of simplicity, we will also not allow to replace actions by the empty event structure. In the conclusion,we will discuss possible extensions of our results to these cases.

A refinement will be a function r specifiying, for each action a, an event structure $r(a)$ which is to be substituted for a. Interesting refinements (and also the refinements in our examples) will mostly refine only certain actions, hence replace most actions by themselves. However, for uniformity (and for simplicity in proofs) we consider all actions to be refined.

Given an event structure $\mathcal{E}$ and a refinement function r, we construct the refined event structure $r(\mathcal{E})$ as follows. Each event e labelled by a is replaced by a disjoint copy, $\mathcal{E}_e$, of $r(a)$. The causality and conflict structure is inherited from $\mathcal{E}$: every event which was causally before e will be causally before all events of $\mathcal{E}_e$, all events which causally followed e will causally follow all the events of $\mathcal{E}_e$, and all events in conflict with e will be in conflict with all the events of $\mathcal{E}_e$.

Definition A *refinement* $r : Act \to \mathbb{E} - \{O\}$ is a function that takes any action

$a \in Act$ into a finite, conflict-free, non-empty event structure $r(a) \in \mathbb{E}$.
For an event structure $\mathcal{E}$ and a refinement r, the event structure $r(\mathcal{E})$ is defined by
- $E_{r(\mathcal{E})} = \{(e, e') | e \in E_{\mathcal{E}}, e' \in E_{r(l_{\mathcal{E}}(e))}\}$,
- $(d, d') \leq_{r(\mathcal{E})} (e, e')$ iff $d <_{\mathcal{E}} e$ or $(d = e \wedge d' \leq_{r(l_{\mathcal{E}}(d))} e')$,
- $(d, d') \#_{r(\mathcal{E})} (e, e')$ iff $d \#_{\mathcal{E}} e$,
- $l_{r(\mathcal{E})}(e, e') = l_{r(l_{\mathcal{E}}(e))}(e')$.

As one can easily check, $r(\mathcal{E})$ is an event structure indeed.

Since we do not distinguish isomorphic event structures, replacing actions by conflict-free event structures corresponds to replacing actions by *pomsets*. It is easy to see that replacing actions by different representatives of a pomset leads to isormophic refined event structures.
As the lemma below will show, the behaviour of a refined event structure $r(\mathcal{E})$ may be deduced from the behaviour of $\mathcal{E}$ and from the behaviour of the event structures which are substituted for actions. On the other hand, we may derive information about the behaviour of $\mathcal{E}$ from the behaviour of $r(\mathcal{E})$.

Lemma Let $\mathcal{E}$ be an event structure, r a refinement.

i. $\tilde{C} \subseteq E_{r(\mathcal{E})}$ is a configuration of $r(\mathcal{E})$ iff
$\tilde{C} = \{(e, e') | e \in C, e' \in C_e\}$ where
C is a configuration of $\mathcal{E}$,
C_e is a configuration of $r(l_{\mathcal{E}}(e))$ for $e \in C$,
C_e is complete if e not maximal in C with respect to $\leq_{\mathcal{E}}$.

ii. If $\tilde{C} \to_{r(\mathcal{E})} \tilde{C'}$ then $pr_1(\tilde{C}) \to_{\mathcal{E}} pr_1(\tilde{C'})$ (pr_1 denotes projection to the first component).

Proof straightforward (see the full version of this paper [GG]). ∎

Example 2.1

We now recall the example of [CDP]. They considered the two systems $P = a|b$ and $Q = ab + ba$, representable by the following event structures.

$$\mathcal{E}_P = \quad a \quad b \quad , \quad \mathcal{E}_Q = \quad \begin{array}{ccc} a & \# & b \\ \downarrow & & \downarrow \\ b & & a \end{array}$$

In all known interleaving semantics, P and Q are considered equivalent; we have $\mathcal{E}_P \approx_{ib} \mathcal{E}_Q$. However, if we allow to refine the action a into the pomset $a_1 \rightarrow a_2$, this gives rise to the two systems

$$\mathcal{E}_{P'} = \begin{array}{c} a_1 \\ \downarrow \\ a_2 \end{array} \quad b \quad , \quad \mathcal{E}_{Q'} = \begin{array}{ccc} a_1 & \# & b \\ \downarrow & & \downarrow \\ a_2 & & a_1 \\ \downarrow & & \downarrow \\ b & & a_2 \end{array}$$

and they are not interleaving equivalent; indeed they are not even interleaving trace equivalent: $\mathcal{E}_{P'}$ allows for the sequence $a_1\, b\, a_2$ whereas $\mathcal{E}_{Q'}$ doesn't.

This shows that both interleaving trace equivalence and interleaving bisimulation equivalence are not preserved by action refinement. Even more, the same can be said for all equivalences identifying P and Q and respecting interleaving trace equivalence, e.g. failure equivalence [BHR], testing equivalence [DH].

An event structure equivalence which is indeed preserved by refinement is event structure isomorphism. However, the main purpose of introducing an equivalence notion is to abstract from certain details in a system representation. For example, we would like to express that the processes a and $a + a$ exhibit the same behaviour. Furthermore, we would like to identify processes like $(a|(b + c)) + (a|b) + ((a + c)|b)$ and $(a|(b + c)) + ((a + c)|b)$ (absorption law, see [BC]). This is not possible when using event structure isomorphism. Hence, in the sequel we will consider various equivalence notions in between these two extremes (interleaving trace equivalence and event structure isomorphism), taking into account the concurrency and the conflict structure ("branching-time" semantics) in more and more detail.

3 Step semantics

A more discriminating view of concurrent systems than that offered by interleaving semantics is obtained by modelling concurrency as either arbitrary interleaving or simultaneous execution. This view is taken in calculi like SCCS [Milner a], CIRCAL [Milne] and MEIJE [AB]. In [TV], this idea is applied to give a non-interleaving semantics to theoretical CSP, called *step failure semantics*. The word *step* originates from Petri net theory where it denotes a set (or multiset) of concurrently executable transitions. Recently, a step semantics for CCS has been defined [DDMa], inspired by [AB]. Step semantics give a more precise account of concurrency than interleaving semantics, e.g. the systems $a|b$ and $ab + ba$ are distinguished. This means that the example given in [CDP] constitutes an argument against interleaving semantics but not against step semantics. We will formalise some step equivalence notions and then discuss an example which shows that even these equivalences are not preserved by refinement.

Step semantics are defined by generalising the single action transitions $C \xrightarrow{a} C'$ from section 1 to transitions of the form $C \xrightarrow{A} C'$ where A is a multiset over Act, representing actions occurring concurrently. In particular, we allow actions to occur concurrently with themselves ("autoconcurrency"). Using this new kind of transitions, *step trace equivalence* and *step bisimulation equivalence* are straightforward generalisations of the corresponding interleaving equivalences, see e.g. [Pomello].

Definition Let $\mathcal{E}$ be an event structure.

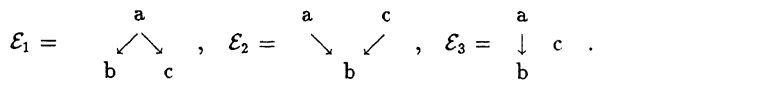
$C \xrightarrow{A} C'$ iff $A \in I\!N^{Act}$ (A is a multiset over Act), $C \rightarrow_{\mathcal{E}} C', C' \setminus C = G$ such that $\forall e, e' \in G \; e \; co \; e'$ and $l(G) = A$ where $l(G)(a) = |\{e \in G | l(e) = a\}|$.

Using this form of transitions, we obtain *step trace equivalence*, $\approx_{st}$, and *step bisimulation equivalence*, $\approx_{sb}$, exactly as the corresponding interleaving equivalences in section 1. Like for interleaving, $\mathcal{E} \approx_{sb} \mathcal{F}$ implies $\mathcal{E} \approx_{st} \mathcal{F}$. Moreover (as far as we know) all other interesting step equivalence notions are positioned somewhere in between (recall that we do not consider abstraction from internal actions).

Considering the two systems $P = a|b$ and $Q = ab + ba$ from [CDP], represented as event structures $\mathcal{E}_P$ and $\mathcal{E}_Q$ in example 2.1, we find that $\mathcal{E}_P$ and $\mathcal{E}_Q$ are not equivalent in step semantics. The step $\{a, b\}$ is possible in $\mathcal{E}_P$ but not in $\mathcal{E}_Q$. So the example in [CDP] is not adequate for step semantics. Here we give an example showing that both $\approx_{st}$ and $\approx_{sb}$ are not invariant under refinement of actions, as well as all equivalences included between them, e.g. step failure equivalence (for both $\approx_{st}$ and $\approx_{sb}$ there exist even simpler examples [GG]).

Example 3.1

First consider the following three systems:



Now we consider the two composed systems $\mathcal{E} = \mathcal{E}_1 + \mathcal{E}_2$ and $\mathcal{F} = \mathcal{E}_1 + \mathcal{E}_2 + \mathcal{E}_3$.
The +-sign is supposed to indicate that a system behaves alternatively like one of its components. It may easily be "implemented" by indicating that all events in one component are in conflict with all events in the others. (For representing $\mathcal{E}$ and $\mathcal{F}$ as terms, we would need to use a sequential composition operator or a TCSP-like parallel composition.)

We have $\mathcal{E} \approx_{sb} \mathcal{F}$ [GV]. However, when refining c into $c_1 \rightarrow c_2$ only the refinement of $\mathcal{F}$ may perform the sequence of actions $c_1 \; a \; b \; c_2$. The resulting systems $\mathcal{E}'$ and $\mathcal{F}'$ are not even interleaving trace equivalent.
So let $\approx$ be an equivalence included between $\approx_{st}$ and $\approx_{sb}$, then also $\mathcal{E} \approx \mathcal{F}$, but $\mathcal{E}' \not\approx \mathcal{F}'$.

Thus we have shown that all the currently known versions of step equivalence are not preserved by refinement.

4 Partial order semantics

In [CDP] it was claimed that equivalence based on considering partially ordered executions is preserved by refinement. In this section we will make this claim more precise.

In "linear time" semantics, when considering only the sets of all possible executions of systems, the claim is indeed true. As explained in section 1, the set of possible executions of an event structure is represented as the set of all pomsets derivable from its configurations. We call two event structures $\mathcal{E}$ and $\mathcal{F}$ *pomset trace equivalent* ($\mathcal{E} \approx_{pt} \mathcal{F}$) if their sets of pomsets are equal. The refinement theorem for $\approx_{pt}$ then follows from the lemma about the behaviour of refined event structures in section 2 ([GG], formalising the proof sketch from [CDP]).

Next, we discuss several suggestions to define equivalence notions based on partial orders and recording where choices are made. We show that most of these fail in general to be preserved by refinement. Finally we show that the last and strongest notion is indeed invariant with respect to refinement.

In [BC] it was suggested to generalise the idea of bisimulation by considering transitions labelled by pomsets. So we consider now transitions $C \xrightarrow{u} C'$ where u is a pomset over Act.

Definition Let $\mathcal{E}$ be an event structure.

$C \xrightarrow{u} C'$ iff $C \rightarrow_{\mathcal{E}} C'$ and u is the isomorphism class of $\mathcal{E}\lceil(C' \setminus C)$.

Using this kind of transitions, *pomset bisimulation equivalence*, $\approx_{pb}$, is obtained exactly as $\approx_{ib}$.

This equivalence notion is clearly stronger than both step bisimulation equivalence and pomset trace equivalence: $\mathcal{E} \approx_{pb} \mathcal{F}$ implies $\mathcal{E} \approx_{sb} \mathcal{F}$ and $\mathcal{E} \approx_{pt} \mathcal{F}$; moreover, the processes $a|b$ and $(a|b) + ab$ are *sb*–equivalent but not *pb*–equivalent; $a(b + c)$ and $ab + ac$ are pomset trace equivalent but not *pb*–equivalent.

However, *pb*–equivalence is not preserved by refinement.

Example 4.1

Consider $a(b + c) + (a|b)$ and $a(b + c) + (a|b) + ab$. We have $P \approx_{pb} Q$. However, when refining a into $a_1 \rightarrow a_2$ and executing a_1, we may arrive in a situation in the second system where a_2 and b may be only executed sequentially and where c is excluded. This is not possible in the first system.

In [GG] we also showed that the *generalised pomset bisimulation equivalence* of [GV] is not preserved by refinement.

Another equivalence notion based on the idea of bisimulation with partial orders that might be preserved by refinement was suggested in [Devillers]. It turned out that this notion coincides with the NMS partial ordering equivalence suggested earlier in [DDM]. We rephrase the definition here in terms of event structures as follows.

Definition Let $\mathcal{E}, \mathcal{F}$ be event structures.

A relation $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F})$ is called a *weak history preserving bisimulation between $\mathcal{E}$ and $\mathcal{F}$* iff $(\emptyset, \emptyset) \in R$ and if $(C, D) \in R$ then
- $\mathcal{E}\lceil C$ and $\mathcal{F}\lceil D$ are isomorphic,
- $C \rightarrow_{\mathcal{E}} C' \Rightarrow \exists D'$ with $D \rightarrow_{\mathcal{F}} D'$ and $(C', D') \in R$,
- $D \rightarrow_{\mathcal{F}} D' \Rightarrow \exists C'$ with $C \rightarrow_{\mathcal{E}} C'$ and $(C', D') \in R$.

$\mathcal{E}$ and $\mathcal{F}$ are *weakly history preserving equivalent* ($\mathcal{E} \approx_{wh} \mathcal{F}$) iff there exists a weak history preserving bisimulation between $\mathcal{E}$ and $\mathcal{F}$.

Note that the isomorphism requirement guarantees that the labels of the events in $C' \setminus C$ and $D' \setminus D$ correspond as well.

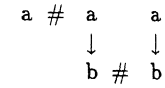
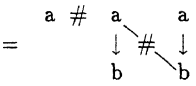
As observed in [Devillers], it is sufficient to consider only those transitions $C \rightarrow_{\mathcal{E}} C'$, (resp. $D \rightarrow_{\mathcal{F}} D'$) where $C'(D')$ is obtained from $C(D)$ by executing exactly one event.

The two systems considered in example 4.1 are pomset bisimulation equivalent but not weakly history preserving equivalent. However, *wh*-equivalence is not stronger than pomset bisimulation, as

shown by the following example; the two notions are in general incomparable. We will show later that wh-equivalence does respect pomset bisimulation for systems without autoconcurrency.

The following example will also show that wh-equivalence is in general not preserved by refinement. This example was suggested to us by Rabinovich. He used it for showing that $\approx_{wh}$ is not a congruence with respect to a TCSP-like parallel composition.

Example 4.2

Let $\mathcal{E} =$  and $\mathcal{F} =$  .

It is straightforward to check that $\mathcal{E} \approx_{wh} \mathcal{F}$. However, $\mathcal{E}$ and $\mathcal{F}$ are not pomset bisimulation equivalent. After executing a, it is always possible to execute $a \rightarrow b$ in $\mathcal{E}$, in $\mathcal{F}$ it may be impossible to execute $a \rightarrow b$ after a. When refining a into $a_1 \rightarrow a_2$, the resulting systems are no longer wh-equivalent, not even interleaving bisimulation equivalent. This can be proven by providing a formula in Hennessy-Milner logic [HM] that is satisfied by the refinement of $\mathcal{F}$, but not by the refinement of $\mathcal{E}$. Such a formula is:

$\langle a_1 \rangle \langle a_2 \rangle (\langle b \rangle T \wedge \langle a_1 \rangle \neg \langle b \rangle T)$.

An equivalence respecting both pomset bisimulation and wh-equivalence may be considered by extending the definition of pomset bisimulation with the requirement that, for any $(C, D) \in \mathcal{R}$, $\mathcal{E}\lceil C$ and $\mathcal{F}\lceil D$ should be isomorphic. However, in [GG] we showed that also this equivalence would not preserve refinement.

We finally define a stronger version of history perserving equivalence which will respect pomset bisimulation. This notion was first suggested in [TRH] in terms of behaviour structures. We will show that this equivalence is preserved by refinement. For systems without autoconcurrency, this equivalence coincides with $\approx_{wh}$. This will imply the result that $\approx_{wh}$ is invariant against refinement for systems without autoconcurrency.

Definition Let $\mathcal{E}, \mathcal{F}$ be event structures.

A relation $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F}) \times \mathcal{P}(E_{\mathcal{E}} \times E_{\mathcal{F}})$ is called a *history preserving bisimulation* between $\mathcal{E}$ and $\mathcal{F}$ if $(\emptyset, \emptyset, \emptyset) \in R$ and whenever $(C, D, f) \in R$ then

- $f : C \rightarrow D$ is an isomorphism between $\mathcal{E}\lceil C$ and $\mathcal{F}\lceil D$,
- $C \rightarrow_{\mathcal{E}} C' \Rightarrow \exists D', f'$ with $D \rightarrow_{\mathcal{F}} D'$, $(C', D', f') \in R$ and $f'\lceil C = f$,
- $D \rightarrow_{\mathcal{F}} D' \Rightarrow \exists C', f'$ with $C \rightarrow_{\mathcal{E}} C'$, $(C', D', f') \in R$ and $f'\lceil C = f$.

$\mathcal{E}$ and $\mathcal{F}$ are *history preserving equivalent* ($\mathcal{E} \approx_h \mathcal{F}$) iff there exists a history preserving bisimulation between $\mathcal{E}$ and $\mathcal{F}$.

Again it is sufficient to consider only those transitions $C \rightarrow_{\mathcal{E}} C'$ (resp. $D \rightarrow_{\mathcal{F}} D'$) where C' (resp. D') is obtained from C (resp. D) by executing exactly one event, as can be proved straightforwardly.

Clearly, we have $\mathcal{E} \approx_h \mathcal{F} \Rightarrow \mathcal{E} \approx_{wh} \mathcal{F}$. However the two systems of example 4.2 are not h-equivalent.

Proposition $\mathcal{E} \approx_h \mathcal{F} \Rightarrow \mathcal{E} \approx_{pb} \mathcal{F}$.

Proof

We show that any history preserving bisimulation between $\mathcal{E}$ and $\mathcal{F}$ is also a pomset bisimulation

between $\mathcal{E}$ and $\mathcal{F}$(after leaving out the isomorphism component). Let R be a h-bisimulation, and suppose $(C,D,f) \in R$ and $C \xrightarrow{u} C'$. Then $C \rightarrow_{\mathcal{E}} C'$, thus $\exists D', f'$ with $D \rightarrow_{\mathcal{F}} D', (C', D', f') \in R$ and $f' \lceil C = f$. Since f' is an isomorphism and $f' \lceil C = f$, *range* $(f' \lceil (C' \setminus C)) =$ *range* $(f') \setminus$ *range* $(f) = D' \setminus D$, so $f' \lceil (C' \setminus C)$ is an isomorphism between $C' \setminus C$ and $D' \setminus D$. Hence $D \xrightarrow{u} D'$, so R satisfies the first clause of a pomset bisimulation. The second clause follows by symmetry. ■

From this proof we learn that h-bisimulation not only respects pomset bisimulation but even the previous proposal combining weak history preserving equivalence and pomset bisimulation. Thus $\approx_h$ is the strongest equivalence considered so far (except for event structure isomorphism of course). Nevertheless it is possible to abstract from certain details in a system representation: we have $a \approx_h a + a$ and $(a|(b+c)) + (a|b) + ((a+c)|b) \approx_h (a|(b+c)) + ((a+c)|b)$ (absorption law).

We now show that $\approx_h$ is preserved by refinement.

Theorem Let $\mathcal{E}, \mathcal{F}$ be event structures, let r be a refinement.

Then $\mathcal{E} \approx_h \mathcal{F} \Rightarrow r(\mathcal{E}) \approx_h r(\mathcal{F})$.

Sketch of Proof
Let $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F}) \times \mathcal{P}(E_{\mathcal{E}} \times E_{\mathcal{F}})$ be a history preserving bisimulation between $\mathcal{E}$ and $\mathcal{F}$.

Let $\tilde{\mathcal{R}} = \{(\tilde{C}, \tilde{D}, \tilde{f}) \in \mathcal{C}(r(\mathcal{E})) \times \mathcal{C}(r(\mathcal{F})) \times \mathcal{P}(E_{r(\mathcal{E})} \times E_{r(\mathcal{F})}) |$
$\exists (C,D,f) \in R$ such that $pr_1(\tilde{C}) = C$, $pr_1(\tilde{D}) = D$
and $\tilde{f}: \tilde{C} \rightarrow \tilde{D}$ is a bijection, satisfying $\tilde{f}\ (e,e') = (f(e), e')\}$

Using again the Lemma about the behaviour of refined event structures from section 2, it may be shown that $\tilde{R}$ is a history preserving bisimulation between $r(\mathcal{E})$ and $r(\mathcal{F})$ [GG]. ■

Finally we show that $\approx_{wh}$ and $\approx_h$ coincide for event structures where concurrent events may not carry the same label. As a corrollary we then have that also $\approx_{wh}$ is preserved by refinement in this case and respects pomset bisimulation.

Definition $\mathcal{E}$ is an event structure *without autoconcurrency* iff

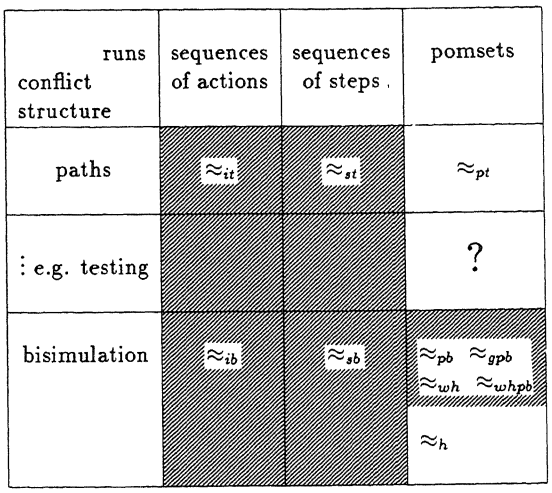
$\forall d, e \in E_{\mathcal{E}} : d \ co\ e$ and $l(d) = l(e) \Rightarrow d = e$.

Theorem For event structures $\mathcal{E}, \mathcal{F}$ without autoconcurrency, $\mathcal{E} \approx_{wh} \mathcal{F} \Leftrightarrow \mathcal{E} \approx_h \mathcal{F}$.

Proof see [GG]. Hint: For configurations $C \in \mathcal{C}(\mathcal{E})$ and $D \in \mathcal{C}(\mathcal{F})$ there can be at most one isomorphism between $\mathcal{E} \lceil C$ and $\mathcal{F} \lceil D$. ■

Conclusion

In this paper we have shown that equivalences based on interleaving of atomic actions or of steps (multisets of concurrently executable actions) are not preserved when changing the level of atomicity of actions. However, we could show that certain equivalences based on modelling causal relations explicitly by partial orders are indeed preserved by refinement of actions. We considered "linear time" approaches, where the behaviour of a system is equated to the set of possible runs, and "branching time" approaches, where the conflict structure of systems is taken into account. We could show the negative results about the interleaving approaches regardless of the level of detail in modelling the conflict behaviour. However, for the positive results about the partial order approaches, the conflict structure turned out to be crucial. An interesting topic for further research would be to investigate testing equivalences based on partial orders, taking the conflict structure in a weaker form into account. For an overview consider the following diagram:



means: not preserved by refinement

This diagram is not at all complete. A naturally arising question is to what extent it is actually necessary to move to partial orders to achieve invariance of equivalence under refinement (here we have only shown that steps are not sufficient). In fact, also ST-bisimulation [GV] is preserved by refinement. ST-bisimulation does not respect pomset trace equivalence. Another equivalence being preserved by refinement was proposed by Hennessy [AH]; however it is defined on a syntactic level and is not applicable to such a wide class of systems as considered here, e.g. it is not possible to treat full CCS.

The refinement operation we have considered replaced actions by non-empty, conflict-free event structures. It is debatable whether one should consider refinements where replacing actions by the empty event structure is allowed (*forgetful refinements*). Such refinements can drastically change the structure of processes, they can not be explained by a change in the level of abstraction at which processes are regarded. Nevertheless, our results hold also for forgetful refinements (with slightly more complicated proofs). On the other hand, relaxing the condition that replacements have to be conflict-free seems very natural and fits nicely with the concept of flow event structures as introduced in [BCa]. We expect that our results may then be generalised.

Acknowledgements

This paper was initiated by a dicussion with Albert Meyer and Ernst-Rüdiger Olderog at ICALP 87 in Karlsruhe. Alex Rabinovich helped us by supplying the two systems considered in example 4.2. Many thanks also to Ilaria Castellani, Rocco De Nicola and Frits Vaandrager for helpful discussions and comments, and to Gertrud Jacobs for her patience and careful preparation of the manuscript.

References

[**AB**] D. Austry, G. Boudol: *Algèbre de processus et synchronisations*, Theoretical Computer Science, Vol. 30, pp. 91-131, 1984

[**AH**] L. Aceto, M. Hennessy: *Towards Action-Refinement in Process Algebras*, Report 3/88, Computer Science, University of Sussex, Brighton, 1988

[**BC**] G. Boudol, I. Castellani: *On the Semantics of Concurrency: Partial Orders and Transition Systems*, Proc. TAPSOFT 87, Vol. I, LNCS 249, Springer-Verlag, pp 123-137, 1987

[**BCa**] G. Boudol, I. Castellani: *Permutation of Transitions: An Event Structure Semantics for CCS and SCCS*, handout at the REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout, The Netherlands, May 30 - June 3, 1988

[**BHR**] S.D. Brookes, C.A.R. Hoare, A.W. Roscoe: *A Theory of Communicating Sequential Processes*, Journal of the ACM, Vol. 31, No. 3, pp 560-599, 1984

[**CDP**] L. Castellano, G. De Michelis, L. Pomello: *Concurrency vs Interleaving: An Instructive Example*, Bulletin of the EATCS 31, pp 12-15, 1987

[**DDM**] P. Degano, R. De Nicola, U. Montanari: *Observational Equivalences for Concurrency Models*, in : Formal Description of Programming Concepts - III, Proc. of the third IFIP WG 2.2 working conference, ed. M. Wirsing, Elsevier Science Publishers B.V. (North Holland), pp 105-129, 1987

[**DDMa**] P. Degano, R. De Nicola, U. Montanari: *A Distributed Operational Semantics for CCS Based on Condition/Event Systems*, Acta Informatica Vol. 26, pp. 59-91, 1988

[**Devillers**] R. Devillers: *On the Definition of a Bisimulation Notion Based on Partial Words*, Petri Net Newsletter 29, pp 16-19, April 1988

[**DH**] R. De Nicola, M. Hennessy: *Testing Equivalences for Processes*, Theoretical Computer Science, Vol. 34, pp. 83-133, 1984

[**GG**] R.J. van Glabbeek, U. Goltz: *Equivalence Notions for Concurrent Systems and Refinement of Actions*, Arbeitspapiere der GMD 366, February 1989

[**Grabowski**] J. Grabowski: *On Partial Languages*, Fundamenta Informatica IV.2, pp 427-498, 1981

[**GV**] R.J. van Glabbeek, F.W. Vaandrager: *Petri Net Models for Algebraic Theories of Concurrency*, Proc. PARLE, Vol. II, LNCS 259, Springer-Verlag, pp 224-242, 1987

[**HM**] M. Hennessy, R. Milner: *Algebraic Laws for Nondeterminism and Cocurrency*, Journal of the ACM, pp 137-161, 1985

[**Milne**] G.J. Milne: *CIRCAL and the Representation of Communication, Concurrency and Time*, Transactions on Programming Languages and Systems (ACM), Vol. 7, No. 2, pp 270-298, 1985

[**Milner**] R. Milner: *A Calculus of Communicating Systems*, LNCS 92, Springer-Verlag, 1980

[**Milner a**] R. Milner: *Calculi for Synchrony and Asynchrony*, Theoretical Computer Science, Vol. 25, No. 3, pp 267-310, 1983

[**NPW**] M. Nielsen, G.D. Plotkin, G. Winskel: *Petri Nets, Event Structures and Domains, Part I*, Theoretical Computer Science, Vol. 13, No. 1, pp 85-108, 1981

[**Park**] D. Park: *Concurrency and Automata on Infinite Sequences*, Theoretical Computer Science (5th GI-Conference), LNCS 104, Springer-Verlag, pp 167-183, 1981

[**Pomello**] L. Pomello: *Some Equivalence Notions for Concurrent Systems. An Overview*, in: Advances in Petri Nets 1985, LNCS 222, Springer-Verlag, pp 381-400, 1986

[**Pratt**] V.R. Pratt: *Modelling Concurrency with Partial Orders*, International Journal of Parallel Programming, Vol. 15, No. 1, pp 33-71, 1986

[**TRH**] B.A. Trakktenbrot, A. Rabinovich, J. Hirschfeld: *Nets of Processes*, Technical Report 97/88, Tel Aviv Univ., 1988

[**TV**] D. Taubner, W. Vogler: *The Step Failure Semantics*, Proc. STACS 87, LNCS 247, Springer-Verlag, pp 348-359, 1987

[**Winskel**] G. Winskel: *Event Structures*, in: Petri Nets: Applications and Relationships to Other Models of Concurrency, LNCS 255, Springer-Verlag, pp 325-392, 1987