The Refinement Theorem for ST-bisimulation Semantics

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

In this paper I prove that ST-bisimulation equivalence, as introduced in [8], is preserved under refinement of actions. This implies that it is possible to abstract from the causal structure of concurrent systems without assuming action atomicity.

1980 Mathematics Subject Classification (Zentralblatt für Mathematik): 68B10.
1985 Mathematics Subject Classification (Mathematical Reviews): 68Q55.
1987 CR Classification scheme (Computing Reviews): F.3.2.
Key Words & Phrases: Concurrency, Semantic equivalences, Action refinement, Event structures, Bisimulation, Interleaving vs. partial orders.
Note: To appear in: Proceedings IFIP Working Conference on Programming Concepts and Methods, Sea of Galilee, Israel 1990 (M. Broy & C.B. Jones, eds.), Elsevier Science Publishers B.V. (North-Holland), 1990.

INTRODUCTION

Virtually all semantic equivalences employed in theories of concurrency are defined in terms of *actions* that concurrent systems may perform (cf [1-18]). Mostly, these actions are taken to be *atomic*, meaning that they are considered not to be divisible into smaller parts. In this case, the defined equivalences are said to be based on *action atomicity*.

However, in the top-down design of distributed systems it might be fruitful to model processes at different levels of abstraction. The actions on an abstract level then turn out to represent complex processes on a more concrete level. This methodology does not seem to be compatible with non-divisibility of actions and for this reason PRATT [15], LAMPORT [11] and others plead for the use of semantic equivalences that are not based on action atomicity.

As indicated in CASTELLANO, DE MICHELIS & POMELLO [4], the concept of action atomicity can be formalized by means of the notion of *refinement of actions*. A semantic equivalence is *preserved under action refinement* if two equivalent processes remain equivalent after replacing all occurrences of an action a by a more complicated process $r(a)$. In particular, $r(a)$ may be a sequence of two actions a_1 and a_2. An equivalence is strictly based on action atomicity if it is not preserved under refinement.

Most semantic equivalences can be positioned in a two dimensional classification diagram, such as the one of Figure 1. On the x-axis equivalences are ordered with respect to the preserved level of detail of runs of processes. Three well-known points on this axis are *interleaving semantics*, where runs are represented by sequences of action occurrences, *step semantics*, where runs are represented by sequences of multisets of action occurrences - the multisets (or *steps*) representing simultaneous occurrences - and *partial order semantics*, in which all causal dependencies between action occurrences in runs of processes are preserved. On the y-axis the equivalences are ordered with respect to the preserved level of detail of the branching structure of these runs. Two well-known points on this axis are *trace semantics*, where a process is fully determined by the set of its possible (partial) runs, thereby

completely neglecting the branching structure of processes, and *bisimulation semantics*, where also the information is preserved where two different courses of action diverge (although branching of identical courses of action is still neglected). In between there are several *decorated trace semantics*, where part of the branching structure is taken into account. Mostly these are motivated by the observable behaviour of processes, according to some testing scenario. In Figure 1 the equivalences become finer, or more discriminating, when moving upwards or to the right.

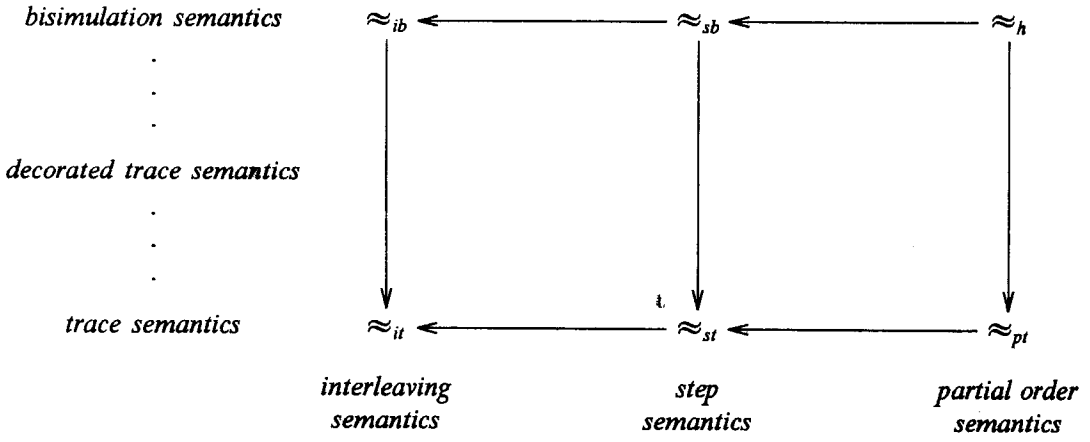


FIGURE 1. *Semantic equivalences*

In [4], CASTELLANO, DE MICHELIS & POMELLO show by means of a simple example that none of the interleaving equivalences - not even bisimulation - is preserved under action refinement. Furthermore they claim that 'on the other hand, the approaches based on partial order are not constrained to the assumption of atomicity'. Therefore they conclude that 'interleaving semantics is adequate only if the abstraction level at which the atomic actions are defined is fixed. Otherwise, partial order semantics should be considered'.

In [6], URSULA GOLTZ & I elaborated on this argument by providing examples, showing that also none of the step equivalences is preserved under refinement, and by formalizing the proof sketch of [4] that trace equivalence based on partial orders *is* invariant under refinement. We also wanted to prove this for bisimulation equivalence based on partial orders, but surprisingly we found that none of the partial order bisimulation equivalences proposed before publication of [4] is preserved under action refinement. However, we *did* prove a refinement theorem for a new notion of bisimulation equivalence based on partial orders, proposed recently by Hirshfeld, RABINOVICH & TRAKHTENBROT [16]. We chose to call this equivalence *history preserving bisimulation equivalence*, notation $\approx_h$. Hence, even in bisimulation semantics, the requirements of preservation under action refinement and capturing causal dependencies in processes by means of partial orders can be conciliated. But of course, this still does not show that in case preservation under refinement is required, it is *necessary* to employ partial order semantics. In this paper I will show that it is not.

Event structures and Petri nets have been established as suitable domains for modelling (both branching and causal aspects of) concurrent systems. Usually a state of a concurrent system is represented by a *configuration* of the associated event structure, or by a *marking* of the associated net. In this paper I argue that when events or transitions are considered to have a duration or structure, configurations or markings do not properly represent all the states of concurrent systems. Instead I propose to use so-called *ST-configurations* or *ST-markings*. The idea to model a state in a safe

labelled marked net as the set of places (*Stellen*) containing a token, together with the set of transitions (*Transitionen*) which are currently firing (an *ST-marking*) originates from VAN GLABBEEK & VAANDRAGER [8]. In this paper I translate this idea to the realm of event structures by introducing *ST-configurations*.

All interleaving, step and partial order equivalences on event structures or Petri nets considered so far, have been defined in terms of configurations or markings. If the constructions from interleaving semantics are applied on ST-configurations instead of ordinary configurations two new points on the x-axis of Figure 1 emerge. *Split-semantics* is just interleaving semantics, but based on interleaving of beginnings and ends of events, instead of entire events; *ST-semantics* is a refinement of split semantics where in addition a link is required between the beginning and the end of any event. Split semantics is more discriminating than step semantics, whereas ST-semantics is as least as discriminating as split semantics. Furthermore ST-trace semantics is less discriminating than trace semantics based on partial orders and ST-bisimulation semantics is less discriminating than history preserving bisimulation semantics (but incomparable with the other bisimulation semantics based on partial orders proposed so far). Hence the situation is as indicated in Figure 2.

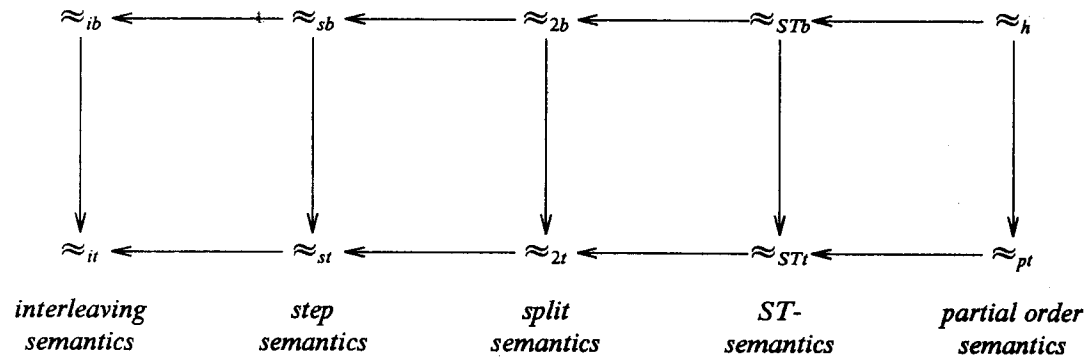


FIGURE 2. *More semantic equivalences*

ST-bisimulation equivalence was introduced by FRITS VAANDRAGER & ME in [8]. In the same paper we observed that for systems without autoconcurrency ST-bisimulation equivalence coincides with split bisimulation equivalence and provided a complete axiomatization on closed ACP-terms for the latter notion. Split bisimulation equivalence was proposed in HENNESSY [10] for a subset of CCS. ACETO & HENNESSY [1] proved that on this subset split bisimulation equivalence is preserved under action refinement. HENNESSY [10] also provided a complete axiomatization for split bisimulation equivalence on this subset. Since - if one forgets about τ-moves - this proof system is sound for ST-bisimulation equivalence, and even for history preserving bisimulation equivalence, it follows that on the domain considered in [10] the three equivalences coincide. In combination with the refinement theorem for history preserving bisimulation equivalence in [6], this yields an alternative proof of Aceto & Hennessy's refinement theorem. Split trace equivalence has been considered in VAANDRAGER [17]. In a joint paper with FRITS VAANDRAGER [9] we will show that on the domain of labelled event structures (prime event structures with binary conflict), or on full CCS, split semantics is not proof against refinement. In fact the equivalences obtained by splitting an event into two parts (its beginning and its end) turned out to be different from the equivalences obtainable by splitting an event into three parts. This was established by means of a rather complicated example (the *owl* example), that also shows that split semantics is strictly less discriminating than ST-semantics. By means of even more complicated examples we established that for each $n \in \mathbb{N}$ split-n semantics is also different from split-n+1 semantics.

The result contributed by the present paper is that ST-bisimulation semantics as well as ST-trace semantics are preserved under action refinement. In [8] it was shown that these semantics do not respect causality. It follows that it is possible to abstract from the causal structure of concurrent systems without assuming action atomicity.

1. Concurrent systems and refinement of actions

In this paper I consider systems that are capable of performing actions from a given set Act of action names. Following [6], as my model for this kind of systems I have chosen labelled event structures here (prime event structures with a binary conflict relation as introduced in Nielsen, Plotkin & Winskel [14]); I could have chosen other models like Petri nets or behaviour structures [16] as well. In this paper I will not distinguish external and internal actions; I do not consider abstraction by hiding of actions.

Definition. A *(labelled) event structure* (over an alphabet Act) is a 4-tuple $\mathcal{E}=(E, <, \sharp, l)$, where
- E is a set of *events*;
- $< \subseteq E \times E$ is a partial order (the *causality relation*) satisfying the principle of *finite causes*:

  $\{e' \in E \mid e' < e\}$ is finite for $e \in E$;
- $\sharp \subseteq E \times E$ is an irreflexive, symmetric relation (the *conflict relation*) satisfying the principle of *conflict heredity*:

  $e_1 \sharp e_2 < e_3 \Rightarrow e_1 \sharp e_3$;
- $l: E \to Act$ is a *labelling function*.

An event structure represents a concurrent system in the following way: action names $a \in Act$ represent actions the system may 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 $e' \sharp e$ means that e' and e cannot happen both in the same run.

One usually writes $e' \leqslant e$ for $e' < e \vee e' = e$, $>$ for $<^{-1}$ and $\geqslant$ for $\leqslant^{-1}$. Causal independence (*concurrency*) of events is expressed by the derived relation $\smile \subseteq E \times E$ defined by: $e' \smile e$ iff $\neg(e' \sharp e \vee e' < e \vee e' > e \vee e' = e)$. By definition $<$, $=$, $>$, $\sharp$ and $\smile$ form a partition of $E \times E$. The concurrency relation $co \subseteq E \times E$, originating from Petri net theory, is defined slightly different from $\smile$: e' co e iff $e' \smile e \vee e' = e$.

The components of an event structure $\mathcal{E}$ will be denoted by respectively $E_{\mathcal{E}}$, $<_{\mathcal{E}}$, $\sharp_{\mathcal{E}}$ and $l_{\mathcal{E}}$. The derived relations will be denoted $\smile_{\mathcal{E}}$, $co_{\mathcal{E}}$, $\leqslant_{\mathcal{E}}$, $>_{\mathcal{E}}$ and $\geqslant_{\mathcal{E}}$.

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

Definition. An *event structure isomorphism* between two event structures $\mathcal{E}, \mathcal{F} \in \mathbb{E}$ is a bijective mapping $f: E_{\mathcal{E}} \to E_{\mathcal{F}}$ such that
- $f(e) <_{\mathcal{F}} f(e') \Leftrightarrow e <_{\mathcal{E}} e'$,
- $f(e) \sharp_{\mathcal{F}} f(e') \Leftrightarrow e \sharp_{\mathcal{E}} e'$ and
- $l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e)$.

$\mathcal{E}$ and $\mathcal{F}$ are *isomorphic* - notation $\mathcal{E} \cong \mathcal{F}$ - if there exists an event structure isomorphism between them. Generally, one does not distinguish isomorphic event structures.

Definition. The *restriction* of an event structure $\mathcal{E}$ to a set $X \subseteq E_{\mathcal{E}}$ of events is the event structure $\mathcal{E} \restriction X = (X, <_{\mathcal{E}} \cap (X \times X), \sharp_{\mathcal{E}} \cap (X \times X), l_{\mathcal{E}} \restriction X)$.

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

In [4] 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. In [6] we considered a more general version of this operation, which I will also use in the present paper: replacing actions by finite, conflict-free, non-empty event structures. Replacing actions by infinite event structures could in general invalidate the principle of finite causes. As explained in [6], 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 [3]. Finally, replacing actions by the empty event structure can drastically change the structure of processes; it can not be explained by a change in the level of abstraction at which processes are regarded. In the concluding section I will discuss possible extensions of my result to these cases.

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

Given an event structure $\mathcal{E}$ and a refinement r, the refined event structure $r(\mathcal{E})$ is constructed as follows. Each event e labelled by a is replaced by a disjoint copy, $r(e)$, 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 $r(e)$, all events which causally followed e will causally follow all the events of $r(e)$, and all events in conflict with e will be in conflict with all the events of $r(e)$.

DEFINITION. A *refinement* $r: Act \to \mathbb{E} - \{0\}$ is a function that takes any action $a \in Act$ into a finite, conflict-free, non-empty event structure $r(a) \in \mathbb{E}$. If $\mathcal{E} \in \mathbb{E}$ and r is a refinement, then $r(\mathcal{E})$ is the event structure defined by:

- $E_{r(\mathcal{E})} = \{(e,e') \mid e \in E_{\mathcal{E}},\ e' \in E_{r(l_{\mathcal{E}}(e))}\}$;
- $(e_1,e_1') <_{r(\mathcal{E})} (e_2,e_2')$ iff $e_1 <_{\mathcal{E}} e_2$ or $(e_1 = e_2 \wedge e_1' <_{r(l_{\mathcal{E}}(e_1))} e_2')$;
- $(e_1,e_1') \#_{r(\mathcal{E})} (e_2,e_2')$ iff $e_1 \#_{\mathcal{E}} e_2$;
- $l_{r(\mathcal{E})}(e,e') = l_{r(l_{\mathcal{E}}(e))}(e')$.

PROPOSITION 1:
i. *If $\mathcal{E} \in \mathbb{E}$ and r is a refinement then $r(\mathcal{E})$ is an event structure indeed.*
ii. *If $\mathcal{E} \in \mathbb{E}$ and r, r' are refinements with $r(a) \cong r'(a)$ for $a \in Act$, then $r(\mathcal{E}) \cong r'(\mathcal{E})$.*
iii. *If $\mathcal{E}, \mathcal{F} \in \mathbb{E}$, r is a refinement and $\mathcal{E} \cong \mathcal{F}$, then $r(\mathcal{E}) \cong r(\mathcal{F})$.*
PROOF: Straightforward. □

This proposition says that refinement is a well-defined operation on event structures, even when isomorphic event structures are identified.

2. THE BEHAVIOUR OF CONCURRENT SYSTEMS I

Let $\mathcal{E}$ be an event structure, modelling the behaviour of a concurrent system P. Classically, a state S of P is given by a set of events from $\mathcal{E}$. Such a set is called a *configuration*. Its elements represent the occurrences of actions that happened before P reached the state S. If two events e' and e cannot happen both in the same run ($e' \#_{\mathcal{E}} e$) then they also cannot occur in the same configuration. So configurations have to be *conflict-free*. Furthermore, if e occurs in a configuration C and e' is a prerequisite for e ($e' <_{\mathcal{E}} e$) then also e' must occur in C. Hence configurations must be *left-closed* with respect to $<_{\mathcal{E}}$. Finally, as is usual, in this paper it is assumed that in a finite period only finitely many actions are performed. Therefore, unlike in many other papers, configurations are required to be finite here.

DEFINITION. A set $X \subseteq E_{\mathcal{E}}$ of events in an event structure $\mathcal{E}$ is *left-closed* in $\mathcal{E}$ if for all $e, e' \in E_{\mathcal{E}}$

$$e' <_{\mathcal{E}} e \in X \Rightarrow e' \in X.$$

X is *conflict-free* in $\mathcal{E}$ if $\mathcal{E} \restriction X$ is conflict-free. A *configuration* of $\mathcal{E}$ is a finite, left-closed, conflict-free subset of $E_{\mathcal{E}}$. Let $\mathcal{C}(\mathcal{E})$ be the set of configurations of $\mathcal{E}$. Write $X \longrightarrow_{\mathcal{E}} X'$ if $X, X' \in \mathcal{C}(\mathcal{E})$ and $X \subseteq X'$.

$X \longrightarrow_{\mathcal{E}} X'$ says that both X and X' represent states of the concurrent system represented by $\mathcal{E}$, and that this system may evolve from the state represented by X to the one represented by X'.

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, one may derive information about the behaviour of $\mathcal{E}$ from the behaviour of $r(\mathcal{E})$.

Let $r(e)$ abbreviate $r(l_{\mathcal{E}}(e))$ and let pr_1 denote projection to the first component.

LEMMA 2: *Let $\mathcal{E}$ be an event structure and r a refinement.*

i. $\tilde{C} \subseteq E_{r(\mathcal{E})}$ *is a configuration of* $r(\mathcal{E})$ *iff*

$\tilde{C} = \{(e, e') \mid e \in C,\ e' \in C_e\}$ *where*

C *is a configuration of* $\mathcal{E}$,

C_e *is a configuration of* $r(e)$ *for* $e \in C$,

$C_e = E_{r(e)}$ *if* e *is not maximal in* C *with respect to* $<_{\mathcal{E}}$.

ii. *If* $\tilde{C} \longrightarrow_{r(\mathcal{E})} \tilde{C}'$ *then* $pr_1(\tilde{C}) \longrightarrow_{\mathcal{E}} pr_1(\tilde{C}')$.

PROOF: See [6]. □

3. EQUIVALENCE NOTIONS FOR CONCURRENT SYSTEMS I

In this section the semantic equivalences of Figure 1 are defined in terms of configurations.
The interleaving equivalences can be defined by means of the *single action transition relations* $\xrightarrow{a}_{\mathcal{E}} \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{E})$ for $a \in Act$ and $\mathcal{E} \in \mathbf{E}$.

DEFINITION. $C \xrightarrow{a}_{\mathcal{E}} C'$ iff $C \longrightarrow_{\mathcal{E}} C'$ and $C' - C = \{e\}$ with $l_{\mathcal{E}}(e) = a$.

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

DEFINITION. A sequence $a_1 \cdots a_n \in Act^*$ is a *(sequential) trace* of an event structure $\mathcal{E}$ if there exist configurations $C_0, \cdots, C_n$ of $\mathcal{E}$ such that $C_0 = \emptyset$ and $C_{i-1} \xrightarrow{a_i}_{\mathcal{E}} C_i$ $(i = 1, \cdots, n)$.
SeqTraces$(\mathcal{E})$ denotes the set of all sequential traces of $\mathcal{E}$.
Two event structures $\mathcal{E}$ and $\mathcal{F}$ are *interleaving trace equivalent* - notation $\mathcal{E} \approx_{it} \mathcal{F}$ - if *SeqTraces*$(\mathcal{E}) =$ *SeqTraces*$(\mathcal{F})$.

DEFINITION. Let $\mathcal{E}, \mathcal{F} \in \mathbf{E}$. A relation $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F})$ is called a *(sequential) bisimulation* between $\mathcal{E}$ and $\mathcal{F}$ if $(\emptyset, \emptyset) \in R$ and whenever $(C, D) \in R$ then for $a \in Act$:

- $C \xrightarrow{a}_{\mathcal{E}} C' \Rightarrow \exists D'$ with $D \xrightarrow{a}_{\mathcal{F}} D'$ and $(C', D') \in R$;
- $D \xrightarrow{a}_{\mathcal{F}} D' \Rightarrow \exists C'$ with $C \xrightarrow{a}_{\mathcal{E}} C'$ and $(C', D') \in R$.

$\mathcal{E}$ and $\mathcal{F}$ are *interleaving bisimulation equivalent* - $\mathcal{E} \approx_{ib} \mathcal{F}$ - if there exists a sequential bisimulation between them.

Step equivalences can be defined by generalizing the single action transition relations $\xrightarrow{a}_{\mathcal{E}} \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{E})$ to *step transition relations* $\xrightarrow{A}_{\mathcal{E}} \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{E})$, where A is a multiset over Act.

DEFINITION. Let $\mathcal{E}$ be an event structure and $A: Act \to \mathbb{N}$ a multiset over Act. For $X \subseteq E_{\mathcal{E}}$ let $l_{\mathcal{E}}(X) \in \mathbb{N}^{Act}$ be the multiset of labels of the events from X, defined by $l_{\mathcal{E}}(X)(a) = |\{e \in X \mid l_{\mathcal{E}}(e) = a\}|$.
Then $C \xrightarrow{A}_{\mathcal{E}} C'$ iff $C \longrightarrow_{\mathcal{E}} C'$ and $C' - C = G \subseteq E_{\mathcal{E}}$ such that $\forall e, e' \in G$: $e \; co_{\mathcal{E}} \; e'$ and $l_{\mathcal{E}}(G) = A$.

Here $C \xrightarrow{A}_{\mathcal{E}} C'$ says that if the system represented by $\mathcal{E}$ is in the state represented by C, then it may concurrently perform the multiset of actions A and reach the state represented by C'. Since A is a multiset rather than a set, actions may occur concurrently with themselves ('autoconcurrency').

DEFINITION. A sequence $A_1 \cdots A_n$ of multisets $A_i \in \mathbb{N}^{Act}$ $(i = 1, ..., n)$ is a *step trace* of an event structure $\mathcal{E}$ if there exist configurations $C_0, \cdots, C_n$ of $\mathcal{E}$ such that $C_0 = \emptyset$ and $C_{i-1} \xrightarrow{A_i}_{\mathcal{E}} C_i$ $(i = 1, ..., n)$.
StepTraces $(\mathcal{E})$ denotes the set of all step traces of $\mathcal{E}$.
Two event structures $\mathcal{E}$ and $\mathcal{F}$ are *step trace equivalent* - $\mathcal{E} \approx_{st} \mathcal{F}$ - if *StepTraces* $(\mathcal{E}) =$ *StepTraces* $(\mathcal{F})$.

DEFINITION. Let $\mathcal{E}, \mathcal{F} \in \mathbf{E}$. A relation $R \subseteq \mathcal{C}(\mathcal{E}) \times \mathcal{C}(\mathcal{F})$ is called a *step bisimulation* between $\mathcal{E}$ and $\mathcal{F}$ if $(\emptyset, \emptyset) \in R$ and whenever $(C, D) \in R$ then for $A \in \mathbb{N}^{Act}$:
- $C \xrightarrow{A}_{\mathcal{E}} C' \;\Rightarrow\; \exists D'$ with $D \xrightarrow{A}_{\mathcal{F}} D'$ and $(C', D') \in R$;
- $D \xrightarrow{A}_{\mathcal{F}} D' \;\Rightarrow\; \exists C'$ with $C \xrightarrow{A}_{\mathcal{E}} C'$ and $(C', D') \in R$.

$\mathcal{E}$ and $\mathcal{F}$ are *step bisimulation equivalent* - $\mathcal{E} \approx_{sb} \mathcal{F}$ - if there exists a step bisimulation between them.

A trace equivalence preserving causal dependencies between action occurrences in runs of processes is the *pomset trace equivalence* as implicitly employed, for instance, in PRATT [15].

DEFINITION. A *partially ordered multiset (pomset)* is an isomorphism class of conflict-free event structures. A pomset u is a *pomset trace* of an event structure $\mathcal{E}$ if u is the isomorphism class of $\mathcal{E} \restriction C$ for some configuration $C \in \mathcal{C}(\mathcal{E})$. *Pomsets* $(\mathcal{E})$ denotes the set of all pomset traces of $\mathcal{E}$.
Two event structures $\mathcal{E}$ and $\mathcal{F}$ are *pomset trace equivalent* - $\mathcal{E} \approx_{pt} \mathcal{F}$ - if *Pomsets* $(\mathcal{E}) =$ *Pomsets* $(\mathcal{F})$.

Sequential traces, step traces as well as pomset traces of an event structure $\mathcal{E}$ represent possible (partial) runs of the system represented by $\mathcal{E}$. A trace of each of these three types specifies a multiset of actions, executed during such a run. However, whereas sequential and step traces in addition only specify a possible order in which these actions may occur (with and without the possibility of simultaneous occurrences), a pomset trace specifies all causal dependencies between the occurrences of these actions, through the partial order inherited from $\mathcal{E}$. From this information all the possible orders in which the actions may occur can be derived.

Like pomset trace equivalence, most of the equivalences that preserve causal dependencies between occurrences of actions are defined by means of partial orders. Therefore, such equivalences are called *partial order equivalences*. It happens that on $\mathbf{E}$ there is only one reasonable trace equivalence based on partial orders - namely $\approx_{pt}$ - and the same can be said about trace equivalences based on steps and on interleaving and about bisimulation equivalences based on steps and on interleaving. However, of late years several bisimulation equivalences based on partial orders have been defined on $\mathbf{E}$:
1986: the *NMS partial ordering equivalence* of DEGANO, DE NICOLA & MONTANARI [5],
1986: the *pomset bisimulation equivalence* or *equipollence* of BOUDOL & CASTELLANI [2],
1987: the *generalized pomset bisimulation equivalence* of VAN GLABBEEK & VAANDRAGER [8] and
1988: the *behaviour structure bisimulation equivalence* of RABINOVICH & TRAKHTENBROT [16].
In my opinion only the last - and finest - one fully captures the interplay of causality and branching and is most worthy of filling up the right upper corner of Figure 1. Originally it was defined on

behaviour structures [16], but in [6] the notion was defined on event structures as well, under the name *history preserving bisimulation equivalence.*

DEFINITION. Let $\mathcal{E},\mathcal{F}\in\mathbf{E}$. 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:

- $f:C\to D$ is an isomorphism between $\mathcal{E}\restriction C$ and $\mathcal{F}\restriction D$;
- $C\xrightarrow{}_{\mathcal{E}}C' \Rightarrow \exists D',f'$ with $D\xrightarrow{}_{\mathcal{F}}D'$, $(C',D',f')\in R$ and $f'\restriction C=f$;
- $D\xrightarrow{}_{\mathcal{F}}D' \Rightarrow \exists C',f'$ with $C\xrightarrow{}_{\mathcal{E}}C'$, $(C',D',f')\in R$ and $f'\restriction C=f$.

$\mathcal{E}$ and $\mathcal{F}$ are *history preserving bisimulation equivalent* - $\mathcal{E}\approx_h\mathcal{F}$ - if there exists a history preserving bisimulation between them.

PROPOSITION 3: *For all equivalences $\approx_1$ and $\approx_2$ defined in this section, the formula*

$$\forall\mathcal{E},\mathcal{F}\in\mathbf{E}:\ \mathcal{E}\approx_1\mathcal{F}\Rightarrow\mathcal{E}\approx_2\mathcal{F}$$

holds iff there is a path $\approx_1\to\cdots\to\approx_2$ in Figure 1.
PROOF: The implications follow directly from the definitions; in order to prove the absence of other implications, it suffices to provide counterexamples against $\approx_{pt}\to\approx_{ib}$, $\approx_{ib}\to\approx_{st}$ and $\approx_{sb}\to\approx_{pt}$.

COUNTEREXAMPLES. In the graphical representations of event structures below, the conventions of [17] are followed: the conflict relation is denoted by means of dotted lines, only immediate conflicts - not the inherited ones - are indicated; the causality relation is represented by arrows, omiting those derivable by transitivity; and 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.

FIGURE 3. *Pomset trace equivalent but not interleaving bisimulation equivalent (standard example)*

The two event structures of Figure 3 are pomset trace equivalent: their pomset traces are $a\to b$, $a\to c$, a and the empty pomset. However, they are not interleaving bisimulation equivalent: both systems represented perform first the action a and then either b or c, but the first system makes the choice between b and c after the execution of a whereas the second one starts with making this choice.

FIGURE 4. *Interleaving bisimulation equivalent but not step trace equivalent (standard example)*

The first system represented in Figure 4 performs two actions a and b concurrently. The second one either performs b after completion of a or vice versa. In interleaving semantics these systems are identified. However, they are not step trace equivalent: only the first system can perform a and b simultaneously.

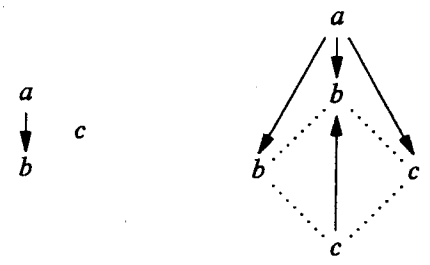


FIGURE 5. *Step bisimulation equivalent but not pomset trace equivalent (new)*

The two systems represented in Figure 5 are step bisimulation equivalent: both systems perform the actions a, b and c exactly once; in both cases a is a prerequisite for b, and c can happen before a, simultaneous with a, between a and b, simultaneous with b, or after b; and in both cases all choices between alternative courses of action are made only when one of the alternatives actually occurs. However, they are not pomset trace equivalent: the pomset resembling the first event structure of Figure 5 is a pomset trace of this first event structure, but not of the second one. □

THEOREM: *Of all equivalences mentioned in this section, only $\approx_{pt}$ and $\approx_h$ are preserved under action refinement.*
PROOF: The two event structures of Figure 5 are step bisimulation equivalent. However, after refining c in $c_1 \rightarrow c_2$ the resulting event structures (below) are not even interleaving trace equivalent.

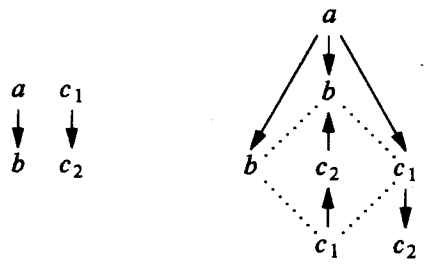


FIGURE 6. *Refined event structures*

Only the first one has a trace $c_1\ a\ b\ c_2$. This shows that no equivalence that is at least as fine as interleaving trace equivalence and at least as coarse as step bisimulation equivalence is preserved under refinement of actions. More counterexamples and the refinement theorems for $\approx_{pt}$ and $\approx_h$ can be found in [6]. □

4. THE BEHAVIOUR OF CONCURRENT SYSTEMS II

A configuration of an event structure $\mathcal{E}$ represents a state S of the system represented by $\mathcal{E}$ by considering two kinds of events with respect to S: those that happened before the system reached this state and those that did not happen (yet). I argue that when events or transitions are considered to have a duration or structure, such configurations do not properly represent all the states of the represented system. Instead I propose to consider a third kind of events with respect to S: those that are currently happening when the system is in state S. This gives rise to the introduction of *ST-configurations* (a name explained in the introduction).

DEFINITION. An *ST-configuration* of $\mathcal{E}$ is a pair (C,P) of subsets of $E_{\mathcal{E}}$, such that $P \subseteq C$, C is finite and conflict-free and

$$e' <_{\mathcal{E}} e \in C \Rightarrow e' \in P.$$

Thus both P and C are configurations and $C-P$ contains only maximal elements in C. An ST-configuration (C,P) represents the state of a concurrent system where C is the set of events whose execution has been started and P (the *past*) is the set of events whose execution has been completed. An ordinary configuration can be regarded as an ST-configuration with $P=C$. Let $\mathbb{S}(\mathcal{E})$ be the set of ST-configurations of $\mathcal{E}$. Write $(C,P) \longrightarrow_{\mathcal{E}} (C',P')$ if $(C,P),(C',P') \in \mathbb{S}(\mathcal{E})$, $C \subseteq C'$ and $P \subseteq P'$.

As in Section 2, 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.

NOTATION. For each pair $(\tilde{C}, \tilde{P}) \in \mathcal{P}(E_{r(\mathcal{E})}) \times \mathcal{P}(E_{r(\mathcal{E})})$ with $\tilde{P} \subseteq \tilde{C}$, there are unique sets $C_e, P_e \subseteq E_{r(e)}$ for every $e \in pr_1(\tilde{C})$ such that $\tilde{C} = \{(e,e') \mid e \in pr_1(\tilde{C}),\ e' \in C_e\}$ and $\tilde{P} = \{(e,e') \mid e \in pr_1(\tilde{C}),\ e' \in P_e\}$. In fact $C_e = \{e' \mid (e,e') \in \tilde{C}\}$ and $P_e = \{e' \mid (e,e') \in \tilde{P}\}$. Now $r^{-1}(\tilde{C}, \tilde{P})$ denotes the unique pair $(C,P) \in \mathcal{P}(E_{\mathcal{E}}) \times \mathcal{P}(E_{\mathcal{E}})$ such that $C = pr_1(\tilde{C})$ and $P = \{e \in C \mid P_e = E_{r(e)}\}$.

LEMMA 4: *Let $\mathcal{E}$ be an event structure and r a refinement.*

i. *$(\tilde{C}, \tilde{P}) \in \mathcal{P}(E_{r(\mathcal{E})}) \times \mathcal{P}(E_{r(\mathcal{E})})$ is an ST-configuration of $r(\mathcal{E})$ iff*

$\tilde{C} = \{(e,e') \mid e \in C,\ e' \in C_e\}$ and $\tilde{P} = \{(e,e') \mid e \in C,\ e' \in P_e\}$ where

(C,P) is an ST-configuration of $\mathcal{E}$,
(C_e,P_e) is an ST-configuration of $r(e)$ for $e \in C$,
$P_e = E_{r(e)}$ iff $e \in P$.

ii. *If $(\tilde{C}, \tilde{P}) \longrightarrow_{r(\mathcal{E})} (\tilde{C}', \tilde{P}')$ then $r^{-1}(\tilde{C}, \tilde{P}) \longrightarrow_{\mathcal{E}} r^{-1}(\tilde{C}', \tilde{P}')$.*

PROOF: i. "$\Rightarrow$". Let $(\tilde{C}, \tilde{P}) \in \mathbb{S}(r(\mathcal{E}))$. First I show that $(C,P) := r^{-1}(\tilde{C}, \tilde{P}) \in \mathbb{S}(\mathcal{E})$.

- $P \subseteq C$ by definition.
- C is finite and conflict-free since $\tilde{C}$ is finite and conflict-free.
- Suppose $d <_{\mathcal{E}} e \in C$. I have to show that $d \in P$.
  Since $e \in C = pr_1(\tilde{C})$ there exists $(e,e') \in \tilde{C}$;
  since $r(d)$ is non-empty there exists $(d,d') \in E_{r(\mathcal{E})}$;
  since $d <_{\mathcal{E}} e$ one has $(d,d') <_{r(\mathcal{E})} (e,e') \in \tilde{C}$;
  and since $(\tilde{C}, \tilde{P})$ is an ST-configuration it follows that $(d,d') \in \tilde{P} \subseteq \tilde{C}$.
  Thus $d \in C$. So it remains to be proven that $P_d = E_{r(d)}$. Obviously $P_d \subseteq E_{r(d)}$.
  Now let $d' \in E_{r(d)}$. Then $(d,d') \in E_{r(\mathcal{E})}$. Exactly as above one obtains $(d,d') \in \tilde{P}$, and hence $d' \in P_d$. Thus $P_d = E_{r(d)}$ and $d \in P$.

Next let $e \in C$. Put $C_e = \{e' \mid (e,e') \in \tilde{C}\}$ and $P_e = \{e' \mid (e,e') \in \tilde{P}\}$. I show that $(C_e,P_e) \in \mathbb{S}(r(e))$.

- $P_e \subseteq C_e$ since $\tilde{P} \subseteq \tilde{C}$.
- C_e is finite since $\tilde{C}$ is finite.
- C_e is conflict-free since $r(e)$ is conflict-free.
- Suppose $e' <_{r(e)} e'' \in C_e$. Then $(e,e') <_{r(\mathcal{E})} (e,e'') \in \tilde{C}$. Hence $(e,e') \in \tilde{P}$ and $e' \in P_e$.

Finally the third requirement is met by construction.

"$\Leftarrow$". Let $(C,P)\in\mathcal{S}(\mathcal{E})$ and $(C_e,P_e)\in\mathcal{S}(r(e))$ for $e\in C$. Suppose $P_e=E_{r(e)} \Leftrightarrow e\in P$ for $e\in C$. Put $\tilde{C}=\{(e,e')\mid e\in C,\ e'\in C_e\}$ and $\tilde{P}=\{(e,e')\mid e\in C,\ e'\in P_e\}$. I show that $(\tilde{C},\tilde{P})\in\mathcal{S}(r(\mathcal{E}))$.

- $\tilde{P}\subseteq\tilde{C}$ since $P_e\subseteq C_e$ for $e\in C$.
- $\tilde{C}$ is finite since C and C_e are finite.
- $\tilde{C}$ is conflict-free since $C=pr_1(\tilde{C})$ is conflict-free.
- Suppose $(d,d')<_{r(\mathcal{E})}(e,e')\in\tilde{C}$. Then $d<_{\mathcal{E}}e$ or $d=e\wedge d'<_{r(e)}e'$.

  If $d<_{\mathcal{E}}e$ then $d\in P$, since $e\in C$ and $(C,P)\in\mathcal{S}(\mathcal{E})$. Thus $d\in C$ and $P_d=E_{r(d)}$. Since $d'\in E_{r(d)}=P_d=\{d'\mid(d,d')\in\tilde{P}\}$ it follows that $(d,d')\in\tilde{P}$.

  If $d=e$ then $d'\in P_e=P_d$, since $d'<_{r(e)}e'\in C_e$ and $(C_e,P_e)\in\mathcal{S}(r(e))$. So also in this case one has $(d,d')\in\tilde{P}$, which had to be proved.

ii. Suppose $(\tilde{C},\tilde{P})\rightarrow_{r(\mathcal{E})}(\tilde{C}',\tilde{P}')$, i.e. $(\tilde{C},\tilde{P}),(\tilde{C}',\tilde{P}')\in\mathcal{S}(r(\mathcal{E}))$, $\tilde{C}\subseteq\tilde{C}'$ and $\tilde{P}\subseteq\tilde{P}'$.

Then $r^{-1}(\tilde{C},\tilde{P})$, $r^{-1}(\tilde{C}',\tilde{P}')\in\mathcal{S}(\mathcal{E})$ (by i.) and $r^{-1}(\tilde{C},\tilde{P})\rightarrow_{\mathcal{E}}r^{-1}(\tilde{C}',\tilde{P}')$ by definition. □

I will end this section with a proposition saying that the ST-configurations of an event structure $\mathcal{E}$ describe the behaviour of the represented concurrent system in the same way as the ordinary configurations of the *split* event structure $split(\mathcal{E})$, obtained from $\mathcal{E}$ by splitting every action a into the sequence of actions a^+ and a^-, representing the beginning and the end of a.

DEFINITION. For Λ a set of labels, let $\mathbf{E}(\Lambda)$ denote the domain of event structures labelled over Λ. So $\mathbf{E}=\mathbf{E}(Act)$.
A Λ-*refinement* $r:Act\rightarrow\mathbf{E}(\Lambda)-\{0\}$ is a function that takes any action $a\in Act$ into a finite, conflict-free, non-empty event structure $r(a)\in\mathbf{E}(\Lambda)$. So a refinement as defined in Section 1 is an Act-refinement. If $\mathcal{E}\in\mathbf{E}$ and r is a Λ-refinement, then $r(\mathcal{E})\in\mathbf{E}(\Lambda)$ is defined exactly as in Section 1.

DEFINITION. Put $Act^{\pm}=\{a^+\mid a\in Act\}\cup\{a^-\mid a\in Act\}$. Let $split:Act\rightarrow\mathbf{E}(Act^{\pm})$ be the $Act^{\pm}$-refinement defined by $E_{split(a)}=\{a^+,a^-\}$, $a^+<_{split(a)}a^-$ and $l_{split(a)}(a^+)=a^+$, $l_{split(a)}(a^-)=a^-$. It induces a function $split:\mathbf{E}(Act)\rightarrow\mathbf{E}(Act^{\pm})$. This function was introduced on Petri nets in [8], and on event structures in [17].

PROPOSITION 4: *For each event structure $\mathcal{E}\in\mathbf{E}$, there exists a bijective mapping $i_{\mathcal{E}}:\mathcal{S}(\mathcal{E})\rightarrow\mathcal{C}(split(\mathcal{E}))$, such that for $S\in\mathcal{S}(\mathcal{E})$:*

$$S\rightarrow_{\mathcal{E}}S' \Leftrightarrow i_{\mathcal{E}}(S)\rightarrow_{split(\mathcal{E})}i_{\mathcal{E}}(S').$$

PROOF: $i_{\mathcal{E}}(C,P)=\{(e,(l_{\mathcal{E}}(e))^+)\mid e\in C\}\cup\{(e,(l_{\mathcal{E}}(e))^-)\mid e\in P\}$. Verification of all requirements is straightforward. □

5. EQUIVALENCE NOTIONS FOR CONCURRENT SYSTEMS II

In this section the remaining equivalences of Figure 2 are defined in terms of ST-configurations.
The most straightforward generalization of interleaving semantics to the setting of ST-configurations yields split semantics. Split equivalences can be defined by generalizing the single action transition relations $\xrightarrow{a}_{\mathcal{E}}\subseteq\mathcal{C}(\mathcal{E})\times\mathcal{C}(\mathcal{E})$ to *split transition relations* $\xrightarrow{a^+}_{\mathcal{E}}$, $\xrightarrow{a^-}_{\mathcal{E}}\subseteq\mathcal{S}(\mathcal{E})\times\mathcal{S}(\mathcal{E})$, for $a\in Act$ and $\mathcal{E}\in\mathbf{E}$.

DEFINITION. $(C,P) \xrightarrow{a^+}_{\mathcal{E}} (C',P')$ iff $(C,P) \rightarrow_{\mathcal{E}} (C',P')$, $P'=P$ and $C'-C=\{e\}$ with $l_{\mathcal{E}}(e)=a$.
$(C,P) \xrightarrow{a^-}_{\mathcal{E}} (C',P')$ iff $(C,P) \rightarrow_{\mathcal{E}} (C',P')$, $C'=C$ and $P'-P=\{e\}$ with $l_{\mathcal{E}}(e)=a$.

Here $(C,P) \xrightarrow{a^+}_{\mathcal{E}} (C',P')$ says that if the system represented by $\mathcal{E}$ is in the state represented by (C,P), then it may start performing an action a and reach the state represented by (C',P').
Furthermore $(C,P) \xrightarrow{a^-}_{\mathcal{E}} (C',P')$ says that if the system is in the state represented by (C,P), then it may end performing an action a and reach the state represented by (C',P').

DEFINITION. A sequence $a_1 \cdots a_n \in (Act^{\pm})^*$ is a *split trace* of an event structure $\mathcal{E}$ if there exist ST-configurations $(C_0,P_0), \cdots, (C_n,P_n)$ of $\mathcal{E}$ such that $(C_0,P_0)=(\emptyset,\emptyset)$ and $(C_{i-1},P_{i-1}) \xrightarrow{a_i}_{\mathcal{E}} (C_i,P_i)$ $(i=1, \cdots, n)$. *SplitTraces*$(\mathcal{E})$ denotes the set of all split traces of $\mathcal{E}$.
Two event structures $\mathcal{E}$ and $\mathcal{F}$ are *split trace equivalent* - $\mathcal{E} \approx_{2t} \mathcal{F}$ - if *SplitTraces*$(\mathcal{E})$ = *SplitTraces*$(\mathcal{F})$.

DEFINITION. Let $\mathcal{E},\mathcal{F} \in \mathbf{E}$. A relation $R \subseteq \mathcal{S}(\mathcal{E}) \times \mathcal{S}(\mathcal{F})$ is called a *split bisimulation* between $\mathcal{E}$ and $\mathcal{F}$ if $((\emptyset,\emptyset),(\emptyset,\emptyset)) \in R$ and whenever $((C,P),(D,Q)) \in R$ then for $a \in Act^{\pm}$:
- $(C,P) \xrightarrow{a}_{\mathcal{E}} (C',P') \Rightarrow \exists D',Q'$ with $(D,Q) \xrightarrow{a}_{\mathcal{F}} (D',Q')$ and $((C',P'),(D',Q')) \in R$;
- $(D,Q) \xrightarrow{a}_{\mathcal{F}} (D',Q') \Rightarrow \exists C',P'$ with $(C,P) \xrightarrow{a}_{\mathcal{E}} (C',P')$ and $((C',P'),(D',Q')) \in R$.

$\mathcal{E}$ and $\mathcal{F}$ are *split bisimulation equivalent* - $\mathcal{E} \approx_{2b} \mathcal{F}$ - if there exists a split bisimulation between them.

Alternatively, split equivalences can be defined as ordinary interleaving equivalences on split event structures, and even as step equivalences on split event structures. The following proposition says that this yields the same trace and bisimulation equivalences as the definitions above.

PROPOSITION 5.1: $\mathcal{E} \approx_{2t} \mathcal{F} \Leftrightarrow split(\mathcal{E}) \approx_{it} split(\mathcal{F}) \Leftrightarrow split(\mathcal{E}) \approx_{st} split(\mathcal{F})$
$\mathcal{E} \approx_{2b} \mathcal{F} \Leftrightarrow split(\mathcal{E}) \approx_{ib} split(\mathcal{F}) \Leftrightarrow split(\mathcal{E}) \approx_{sb} split(\mathcal{F})$.
PROOF: Let $i_{\mathcal{E}}: \mathcal{S}(\mathcal{E}) \rightarrow \mathcal{C}(split(\mathcal{E}))$ be the bijection from the previous proposition, then for $S \in \mathcal{S}(\mathcal{E})$ and $a \in Act^{\pm}$: $S \xrightarrow{a}_{\mathcal{E}} S' \Leftrightarrow i_{\mathcal{E}}(S) \xrightarrow{a}_{split(\mathcal{E})} i_{\mathcal{E}}(S')$. Furthermore, if $C,C' \in \mathcal{C}(split(\mathcal{E}))$ and A is a multiset over $Act^{\pm}$ consisting of the actions $a_1^+, \cdots, a_n^+, b_1^-, \cdots, b_m^-$ then

$$C \xrightarrow{A}_{split(\mathcal{E})} C' \Leftrightarrow C \xrightarrow{a_1^+}_{split(\mathcal{E})} \cdots \xrightarrow{a_n^+}_{split(\mathcal{E})} \xrightarrow{b_1^-}_{split(\mathcal{E})} \cdots \xrightarrow{b_m^-}_{split(\mathcal{E})} C'.$$

From this the proposition follows immediately. □

Split-semantics is just interleaving semantics, but based on interleaving of beginnings and ends of action occurrences, instead of entire action occurrences. However, since different occurrences of the same action can not be distinguished, it is in general not possible to tell when an occurrence of a^+ and an occurrence of a^- originate from to the same occurrence of a. ST-semantics is a refinement of split semantics, where occurrences of a^+ and a^- are explicitly connected if they represent the beginning and end of the same occurrence of a.

DEFINITION. A *pre-interval sequence* is a triple (E,l,σ) with E a set, $l: E \rightarrow Act$ a labelling function and σ a sequence over $E^{\pm} = \{e^+ \mid e \in E\} \cup \{e^- \mid e \in E\}$ whose elements are all different, and which can contain e^- only after e^+ (for $e \in E$). For $l: E \rightarrow Act$ define $l^{\pm}: E^{\pm} \rightarrow Act^{\pm}$ by $l(e^+)=(l(e))^+$ and $l(e^-)=(l(e))^-$. Let (E,l,σ) with $\sigma=\alpha_1 \cdots \alpha_n \in (E^{\pm})^*$ be a pre-interval sequence and let $1 \leqslant i < j \leqslant n$. α_i and α_j are *connected*, notation $\alpha_i \prec \alpha_j$, if $\alpha_i = e^+$ and $\alpha_j = e^-$ for certain $e \in E$. Now two pre-interval sequences $(E,l,\alpha_1 \cdots \alpha_n)$ and $(E',l',\beta_1 \cdots \beta_m)$ are *isomorphic* if $n=m$, $l^{\pm}(\alpha_i)=l'^{\pm}(\beta_i)$ for $1 \leqslant i \leqslant n$, and $\alpha_i \prec \alpha_j \Leftrightarrow \beta_i \prec \beta_j$ for $1 \leqslant i < j \leqslant n$. An *interval sequence* is an isomorphism class of pre-interval sequences.

EXAMPLE: Let $E=\{e_0,e_1,e_2,e_3,e_4\}$, $l(e_0)=l(e_4)=b$ and $l(e_1)=l(e_2)=l(e_3)=a$. Figure 7 shows a pre-interval sequence over E, together with its associated interval sequence. The connectedness relation $\prec$ is represented by arcs.

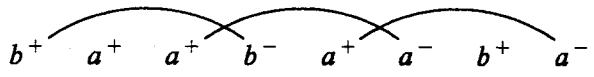


FIGURE 7. *Pre-interval sequence and interval sequence*

DEFINITION. $(C,P) \xrightarrow{e^+}_{\mathfrak{E}} (C',P')$ iff $(C,P) \rightarrow_{\mathfrak{E}} (C',P')$, $P'=P$ and $C'-C=\{e\}$.
$(C,P) \xrightarrow{e^-}_{\mathfrak{E}} (C',P')$ iff $(C,P) \rightarrow_{\mathfrak{E}} (C',P')$, $C'=C$ and $P'-P=\{e\}$.
A structure $(E_{\mathfrak{E}},l_{\mathfrak{E}},\alpha_1 \cdots \alpha_n)$ is a *pre-ST-trace* of an event structure $\mathfrak{E}$ if there exist ST-configurations $(C_0,P_0),\cdots,(C_n,P_n)$ of $\mathfrak{E}$ such that $(C_0,P_0)=(\emptyset,\emptyset)$ and $(C_{i-1},P_{i-1}) \xrightarrow{\alpha_i}_{\mathfrak{E}} (C_i,P_i)$ $(i=1,\cdots,n)$.
An *ST-trace* of $\mathfrak{E}$ is an interval sequence which is the isomorphism class of a pre-ST-trace of $\mathfrak{E}$.
ST-Traces$(\mathfrak{E})$ denotes the set of all ST-traces of $\mathfrak{E}$.
Two event structures $\mathfrak{E}$ and $\mathfrak{F}$ are *ST-trace equivalent* - $\mathfrak{E} \approx_{STt} \mathfrak{F}$ - if *ST-Traces*$(\mathfrak{E})$=*ST-Traces*$(\mathfrak{F})$.

Next I propose another characterization of ST-trace equivalence that will be more convenient later on.

DEFINITION. $\mathfrak{E} \lesssim_{STt} \mathfrak{F}$ iff for every chain of ST-configurations

$$(\emptyset,\emptyset) \rightarrow_{\mathfrak{E}} (C_1,P_1) \rightarrow_{\mathfrak{E}} \cdots \rightarrow_{\mathfrak{E}} (C_n,P_n)$$

in $\mathfrak{E}$ there is a chain

$$(\emptyset,\emptyset) \rightarrow_{\mathfrak{F}} (D_1,Q_1) \rightarrow_{\mathfrak{F}} \cdots \rightarrow_{\mathfrak{F}} (D_n,Q_n)$$

in $\mathfrak{F}$ and a bijection $f:C_n \rightarrow D_n$, satisfying $l_{\mathfrak{F}}(f(e)) = l_{\mathfrak{E}}(e)$, $f(C_i)=D_i$ and $f(P_i)=Q_i$ for $i=1,\cdots,n$.

PROPOSITION 5.2: $\mathfrak{E} \approx_{STt} \mathfrak{F} \Leftrightarrow (\mathfrak{E} \lesssim_{STt} \mathfrak{F} \wedge \mathfrak{F} \lesssim_{STt} \mathfrak{E})$.
PROOF: Write $\mathfrak{E} \lesssim^{*}_{STt} \mathfrak{F}$ iff for every chain

$$(\emptyset,\emptyset) \xrightarrow{\alpha_1}_{\mathfrak{E}} (C_1,P_1) \xrightarrow{\alpha_2}_{\mathfrak{E}} \cdots \xrightarrow{\alpha_n}_{\mathfrak{E}} (C_n,P_n)$$

in $\mathfrak{E}$ (with $\alpha_i \in E_{\mathfrak{E}}^{\pm}$) there is a chain

$$(\emptyset,\emptyset) \xrightarrow{\beta_1}_{\mathfrak{F}} (D_1,Q_1) \xrightarrow{\beta_2}_{\mathfrak{F}} \cdots \xrightarrow{\beta_n}_{\mathfrak{F}} (D_n,Q_n)$$

in $\mathfrak{F}$ and a bijection $f:C_n \rightarrow D_n$, satisfying $l_{\mathfrak{F}}(f(e)) = l_{\mathfrak{E}}(e)$, $f(C_i)=D_i$ and $f(P_i)=Q_i$ for $i=1,\cdots,n$.
Furthermore write $\mathfrak{E} \lesssim^{**}_{STt} \mathfrak{F}$ iff for every chain

$$(\emptyset,\emptyset) \xrightarrow{\alpha_1}_{\mathfrak{E}} (C_1,P_1) \xrightarrow{\alpha_2}_{\mathfrak{E}} \cdots \xrightarrow{\alpha_n}_{\mathfrak{E}} (C_n,P_n)$$

in $\mathfrak{E}$ (with $\alpha_i \in E_{\mathfrak{E}}^{\pm}$) there is a chain

$$(\emptyset,\emptyset) \xrightarrow{\beta_1}_{\mathfrak{F}} (D_1,Q_1) \xrightarrow{\beta_2}_{\mathfrak{F}} \cdots \xrightarrow{\beta_n}_{\mathfrak{F}} (D_n,Q_n)$$

in $\mathfrak{F}$ such that $l_{\mathfrak{E}}^{\pm}(\alpha_i)=l_{\mathfrak{F}}^{\pm}(\beta_i)$ for $1 \leq i \leq n$, and $\alpha_i \prec \alpha_j \Leftrightarrow \beta_i \prec \beta_j$ for $1 \leq i < j \leq n$.
CLAIM 1: $\mathfrak{E} \lesssim_{STt} \mathfrak{F} \Leftrightarrow \mathfrak{E} \lesssim^{*}_{STt} \mathfrak{F}$.
CLAIM 2: $\mathfrak{E} \lesssim^{*}_{STt} \mathfrak{F} \Leftrightarrow \mathfrak{E} \lesssim^{**}_{STt} \mathfrak{F}$.
CLAIM 3: $\mathfrak{E} \lesssim^{**}_{STt} \mathfrak{F} \Leftrightarrow$ *ST-Traces*$(\mathfrak{E}) \subseteq$ *ST-Traces*$(\mathfrak{F})$.

Now the proposition follows by combination of these claims.

Proof of claim 1: "$\Rightarrow$". Suppose $\mathcal{E} \lesssim_{STl} \mathcal{F}$ and $(\varnothing,\varnothing) \xrightarrow{\alpha_1}_{\mathcal{E}} (C_1,P_1) \xrightarrow{\alpha_2}_{\mathcal{E}} \cdots \xrightarrow{\alpha_n}_{\mathcal{E}} (C_n,P_n)$ is a chain in $\mathcal{E}$ with $\alpha_i \in E_{\mathcal{E}}^{\pm}$. Then there must be a chain $(\varnothing,\varnothing) \rightarrow_{\mathcal{F}} (D_1,Q_1) \rightarrow_{\mathcal{F}} \cdots \rightarrow_{\mathcal{F}} (D_n,Q_n)$ in $\mathcal{F}$ and a bijection $f: C_n \rightarrow D_n$, satisfying $l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e)$, $f(C_i)=D_i$ and $f(P_i)=Q_i$ for $i=1,\cdots,n$. Because of this bijection - only considering the 'sizes' of D_i and Q_i - there must be $\beta_i \in E_{\mathcal{F}}^{\pm}$ for $i=1,\cdots,n$ such that $(C_{i-1},P_{i-1}) \xrightarrow{\beta_i}_{\mathcal{E}} (C_i,P_i)$.

"$\Leftarrow$". This follows from the observation that whenever in an event structure $\mathcal{E}$ $(C,P) \rightarrow_{\mathcal{E}} (C',P')$, there exist ST-configurations $(C_0,P_0),\cdots,(C_k,P_k)$ of $\mathcal{E}$ and a sequence $\alpha_1 \cdots \alpha_k \in (E_{\mathcal{E}}^{\pm})^*$ such that $(C_0,P_0)=(C,P)$, $(C_{i-1},P_{i-1}) \xrightarrow{\alpha_i}_{\mathcal{E}} (C_i,P_i)$ $(i=1,\cdots,k)$, and $(C_k,P_k)=(C',P')$.

Proof of claim 2: "$\Rightarrow$". Let $(\varnothing,\varnothing) \xrightarrow{\alpha_1}_{\mathcal{E}} \cdots \xrightarrow{\alpha_n}_{\mathcal{E}} (C_n,P_n)$ and $(\varnothing,\varnothing) \xrightarrow{\beta_1}_{\mathcal{F}} \cdots \xrightarrow{\beta_n}_{\mathcal{F}} (D_n,Q_n)$ be chains of ST-configurations in $\mathcal{E}$ and $\mathcal{F}$ with $\alpha_i \in E_{\mathcal{E}}^{\pm}$ and $\beta_i \in E_{\mathcal{F}}^{\pm}$ for $i=1,\cdots,n$ and let $f: C_n \rightarrow D_n$ be a bijection, satisfying $l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e)$, $f(C_i)=D_i$ and $f(P_i)=Q_i$ for $i=1,\cdots,n$. Since $f(C_i)=D_i$ and $f(P_i)=Q_i$ it follows that $\alpha_i = e^+ \Leftrightarrow \beta_i = f(e)^+$ and $\alpha_i = e^- \Leftrightarrow \beta_i = f(e)^-$ for $i=1,\cdots,n$. Hence $l_{\mathcal{E}}^{\pm}(\alpha_i) = l_{\mathcal{F}}^{\pm}(\beta_i)$ for $1 \leqslant i \leqslant n$, and $\alpha_i \prec \alpha_j \Leftrightarrow \beta_i \prec \beta_j$ for $1 \leqslant i < j \leqslant n$.

"$\Leftarrow$". Let $(\varnothing,\varnothing) \xrightarrow{\alpha_1}_{\mathcal{E}} (C_1,P_1) \xrightarrow{\alpha_2}_{\mathcal{E}} \cdots \xrightarrow{\alpha_n}_{\mathcal{E}} (C_n,P_n)$ and $(\varnothing,\varnothing) \xrightarrow{\beta_1}_{\mathcal{F}} \cdots \xrightarrow{\beta_n}_{\mathcal{F}} (D_n,Q_n)$ be chains of ST-configurations in $\mathcal{E}$ and $\mathcal{F}$ with $\alpha_i \in E_{\mathcal{E}}^{\pm}$ and $\beta_i \in E_{\mathcal{F}}^{\pm}$ for $i=1,\cdots,n$ such that $l_{\mathcal{E}}^{\pm}(\alpha_i) = l_{\mathcal{F}}^{\pm}(\beta_i)$ for $1 \leqslant i \leqslant n$, and $\alpha_i \prec \alpha_j \Leftrightarrow \beta_i \prec \beta_j$ for $1 \leqslant i < j \leqslant n$. Note that $C_i = \{e \in E_{\mathcal{E}} \mid \exists j \leqslant i: \alpha_j = e^+\}$ and $P_i = \{e \in E_{\mathcal{E}} \mid \exists j \leqslant i: \alpha_j = e^-\}$ and similarly for D_i and Q_i. Define $f: C_n \rightarrow D_n$ by $f(e)=d \Leftrightarrow \exists i \leqslant n: (\alpha_i = e^+ \wedge \beta_i = d^+)$. Since $l_{\mathcal{E}}^{\pm}(\alpha_i) = l_{\mathcal{F}}^{\pm}(\beta_i)$ for $1 \leqslant i \leqslant n$, f is well-defined and bijective, and satisfies $l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e)$ and $f(C_i)=D_i$ for $i=1,\cdots,n$. Finally $e \in P_i \Leftrightarrow \exists k < j \leqslant i: (\alpha_k = e^+ \wedge \alpha_j = e^-) \Leftrightarrow \exists k < j \leqslant i: (\beta_k = f(e)^+ \wedge$ (using $\alpha_k \prec \alpha_j \Leftrightarrow \beta_k \prec \beta_j$) $\beta_j = f(e)^-) \Leftrightarrow f(e) \in Q_i$ so $f(P_i)=Q_i$ for $i=1,\cdots,n$.

Finally claim 3 follows directly from the definitions. □

ST-bisimulation equivalence will be defined in the same style as the alternative characterization of ST-trace equivalence. The connection of occurrences of a^+ and a^- that represent the beginning and end of the same occurrence of a is implemented by means of a bijection between related ST-configurations.

DEFINITION. Let $\mathcal{E},\mathcal{F} \in \mathbb{E}$. A relation $R \subseteq \mathcal{S}(\mathcal{E}) \times \mathcal{S}(\mathcal{F}) \times \mathcal{P}(E_{\mathcal{E}} \times E_{\mathcal{F}})$ is called an *ST-bisimulation* between $\mathcal{E}$ and $\mathcal{F}$ if $((\varnothing,\varnothing),(\varnothing,\varnothing),\varnothing) \in R$ and whenever $((C,P),(D,Q),f) \in R$ then:

- $f: C \rightarrow D$ is a bijection, satisfying $l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e)$ and $f(P)=Q$;
- $(C,P) \rightarrow_{\mathcal{E}} (C',P') \Rightarrow \exists D',Q',f'$ with $(D,Q) \rightarrow_{\mathcal{F}} (D',Q')$, $((C',P'),(D',Q'),f') \in R$ and $f' \restriction C = f$;
- $(D,Q) \rightarrow_{\mathcal{F}} (D',Q') \Rightarrow \exists C',P',f'$ with $(C,P) \rightarrow_{\mathcal{E}} (C',P')$, $((C',P'),(D',Q'),f') \in R$ and $f' \restriction C = f$.

$\mathcal{E}$ and $\mathcal{F}$ are *ST-bisimulation equivalent* - $\mathcal{E} \approx_{STb} \mathcal{F}$ - if there exists an ST-bisimulation between them.

Remark that the same equivalence is obtained if in the definition above the general transition relations $\rightarrow$ are replaced by the split transition relations $\xrightarrow{a}$ for $a \in Act^{\pm}$. One direction follows from the requirements for the bijection f; the other one follows as in the proof of Proposition 5.2. Analogously, in [6] it was shown that the definition of history preserving bisimulation equivalence is invariant under replacement of the general transition relations $\rightarrow$ by the single action transition relations $\xrightarrow{a}$ for $a \in Act$. Now it is not difficult to show that if in this version of the definition of ST-bisimulation equivalence the requirement $f(P)=Q$ would be skipped, the resulting equivalence would be split bisimulation equivalence again. This requirement ensures the connection of occurrences of a^+ and a^- originating from the same occurrence of a.

As for split equivalences, the ST-equivalences can be defined alternatively by means of split event structures. First some preliminary definitions.

DEFINITION. For $\mathcal{E}\in\mathbf{E}(Act^{\pm})$, define the connectedness relation $\prec_{\mathcal{E}}\subseteq E_{\mathcal{E}}\times E_{\mathcal{E}}$ by

$$e'\prec_{\mathcal{E}} e \text{ iff } l_{\mathcal{E}}(e)=a^{-} \text{ for certain } a\in Act \text{ and for } d\in E_{\mathcal{E}}\text{: } (d<_{\mathcal{E}} e \Leftrightarrow d\leq_{\mathcal{E}} e').$$

DEFINITION. Write $C \xrightarrow{e}_{\mathcal{E}} C'$ iff $C \rightarrow_{\mathcal{E}} C'$ and $C'-C=\{e\}$. A sequence $\alpha_1\cdots\alpha_n\in E_{\mathcal{E}}^{*}$ is a *pre-trace* of an event structure $\mathcal{E}\in\mathbf{E}(Act^{\pm})$ if there exist configurations $C_0,\cdots,C_n$ of $\mathcal{E}$ such that $C_0=\emptyset$ and $C_{i-1}\xrightarrow{\alpha_i}_{\mathcal{E}} C_i$ $(i=1,\cdots,n)$. Two pre-traces $\alpha_1\cdots\alpha_n$ and $\beta_1\cdots\beta_m$ of $\mathcal{E}$ and $\mathcal{F}$ are $\prec$-*isomorphic* if $n=m$, $l_{\mathcal{E}}(\alpha_i)=l_{\mathcal{F}}(\beta_i)$ for $1\leq i\leq n$, and $\alpha_i\prec_{\mathcal{E}}\alpha_j \Leftrightarrow \beta_i\prec_{\mathcal{F}}\beta_j$ for $1\leq i<j\leq n$. A $\prec$-*trace* of $\mathcal{E}$ is the isomorphism class of a pre-trace of $\mathcal{E}$. $\prec$-*Traces*$(\mathcal{E})$ denotes the set of all $\prec$-traces of $\mathcal{E}$. Two event structures $\mathcal{E}$ and $\mathcal{F}\in\mathbf{E}(Act^{\pm})$ are $\prec$-*trace equivalent* - $\mathcal{E}\approx_{\prec t}\mathcal{F}$ - if $\prec$-*Traces*$(\mathcal{E})=\prec$-*Traces*$(\mathcal{F})$.

DEFINITION. Let $\mathcal{E},\mathcal{F}\in\mathbf{E}(Act^{\pm})$. 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 $\prec$-*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 a bijection, satisfying $l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e)$ and $f(e)\prec_{\mathcal{F}} f(e') \Leftrightarrow e\prec_{\mathcal{E}} e'$;
- $C\rightarrow_{\mathcal{E}} C' \Rightarrow \exists D',f'$ with $D\rightarrow_{\mathcal{F}} D'$, $(C',D',f')\in R$ and $f'\restriction C=f$;
- $D\rightarrow_{\mathcal{F}} D' \Rightarrow \exists C',f'$ with $C\rightarrow_{\mathcal{E}} C'$, $(C',D',f')\in R$ and $f'\restriction C=f$.

$\mathcal{E}$ and $\mathcal{F}$ are $\prec$-*bisimulation equivalent* - $\mathcal{E}\approx_{\prec b}\mathcal{F}$ - if there exists a $\prec$-bisimulation between them.

PROPOSITION 5.3: $\mathcal{E}\approx_{STt}\mathcal{F} \Leftrightarrow split(\mathcal{E})\approx_{\prec t} split(\mathcal{F})$
$\mathcal{E}\approx_{STb}\mathcal{F} \Leftrightarrow split(\mathcal{E})\approx_{\prec b} split(\mathcal{F})$.

PROOF: For $\mathcal{E}\in\mathbf{E}$ define $i:E_{\mathcal{E}}^{\pm}\rightarrow E_{split(\mathcal{E})}$ by $i(e^{+})=(e,(l_{\mathcal{E}}(e))^{+})$ and $i(e^{-})=(e,(l_{\mathcal{E}}(e))^{-})$. Now the bijections $i_{\mathcal{E}}:\mathcal{S}(\mathcal{E})\rightarrow\mathcal{C}(split(\mathcal{E}))$ from Proposition 4 satisfy for $S\in\mathcal{S}(\mathcal{E})$ and $\alpha\in E_{\mathcal{E}}^{\pm}$:

$$S\xrightarrow{\alpha}_{\mathcal{E}} S' \Leftrightarrow i_{\mathcal{E}}(S)\xrightarrow{i(\alpha)}_{split(\mathcal{E})} i_{\mathcal{E}}(S').$$

Hence $\alpha_1\cdots\alpha_n\in(E_{\mathcal{E}}^{\pm})^{*}$ (actually $(E_{\mathcal{E}},l_{\mathcal{E}},\sigma)$ with $\sigma=\alpha_1\cdots\alpha_n$) is a pre-ST-trace of an event structure $\mathcal{E}$ iff $i(\alpha_1)\cdots i(\alpha_n)\in E_{split(\mathcal{E})}^{*}$ is a pre-trace of $split(\mathcal{E})$. Furthermore two pre-ST-traces $\alpha_1\cdots\alpha_n$ and $\beta_1\cdots\beta_m$ of $\mathcal{E}$ are isomorphic iff $i(\alpha_1)\cdots i(\alpha_n)$ and $i(\beta_1)\cdots i(\beta_m)$ are $\prec$-isomorphic. Thus $\prec$-*Traces*$(split(\mathcal{E}))$ is derivable from *ST-Traces*$(\mathcal{E})$ and vice versa. From this the first statement of the proposition follows.

As for the second statement, let $\mathcal{E},\mathcal{F}\in\mathbf{E}$.

$$\mathcal{R}(\mathcal{E},\mathcal{F}) = \{((C,P),(D,Q),f)\in\mathcal{S}(\mathcal{E})\times\mathcal{S}(\mathcal{F})\times\mathcal{P}(E_{\mathcal{E}}\times E_{\mathcal{F}})\mid$$
$$f:C\rightarrow D \text{ is a bijection, satisfying } l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e) \text{ and } f(P)=Q\}.$$

For (S,T,f) and $(S',T',f')\in\mathcal{R}(\mathcal{E},\mathcal{F})$ write $(S,T,f)\rightarrow(S',T',f')$ if $S\rightarrow_{\mathcal{E}} S'$, $T\rightarrow_{\mathcal{F}} T'$, and $f'\restriction C=f$.

$$\mathcal{R}_{split}(\mathcal{E},\mathcal{F}) = \{(C,D,f)\in\mathcal{C}(split(\mathcal{E}))\times\mathcal{C}(split(\mathcal{F}))\times\mathcal{P}(E_{split(\mathcal{E})}\times E_{split(\mathcal{F})})\mid$$
$$f:C\rightarrow D \text{ is a bijection, satisfying } l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e) \text{ and } f(e)\prec_{\mathcal{F}} f(e') \Leftrightarrow e\prec_{\mathcal{E}} e'\}.$$

For (C,D,f) and $(C',D',f')\in\mathcal{R}_{split}(\mathcal{E},\mathcal{F})$ write $(C,D,f)\rightarrow(C',D',f')$ if $C\rightarrow_{split(\mathcal{E})} C'$, $D\rightarrow_{split(\mathcal{F})} D'$, and $f'\restriction C=f$. Define $i:\mathcal{R}(\mathcal{E},\mathcal{F})\rightarrow\mathcal{R}_{split}(\mathcal{E},\mathcal{F})$ by $i(S,T,f)=(i_{\mathcal{E}}(S),i_{\mathcal{F}}(T),i(f))$ where $i_{\mathcal{E}}$ and $i_{\mathcal{F}}$ are the bijections from Proposition 4 and $i(f):i_{\mathcal{E}}(S)\rightarrow i_{\mathcal{F}}(T)$ is defined by $i(f)(e,a^{+})=(f(e),a^{+})$ and $i(f)(e,a^{-})=(f(e),a^{-})$. Now it is not difficult to establish that i is a bijection, satisfying

$$(S,T,f)\rightarrow(S',T',f') \Leftrightarrow i(S,T,f)\rightarrow i(S',T',f').$$

From this it follows that $R\subseteq\mathcal{R}(\mathcal{E},\mathcal{F})$ is an ST-bisimulation between $\mathcal{E}$ and $\mathcal{F}$ iff $i(R)=\{i(S,T,f)\mid(S,T,f)\in R\}\subseteq\mathcal{R}_{split}(\mathcal{E},\mathcal{F})$ is a $\prec$-bisimulation between $split(\mathcal{E})$ and $split(\mathcal{F})$. □

PROPOSITION 5.4: *For all equivalences* $\approx_1$ *and* $\approx_2$ *on* $\mathbf{E}$ *defined so far, the formula*

$$\forall \mathcal{E},\mathcal{F}\in\mathbf{E}\colon \mathcal{E}\approx_1\mathcal{F} \Rightarrow \mathcal{E}\approx_2\mathcal{F}$$

holds iff there is a path $\approx_1 \rightarrow \cdots \rightarrow \approx_2$ *in Figure 2.*

PROOF: In order to prove the announced implications, it suffices to restrict attention to the ones corresponding with an arrow $\approx_1 \rightarrow \approx_2$ in Figure 2. Five of them are dealt with in Proposition 3 already. In order to prove the implications $\approx_{2t} \rightarrow \approx_{st}$ and $\approx_{2b} \rightarrow \approx_{sb}$, consider, for $\mathcal{E}\in\mathbf{E}$, the mapping $j:\mathcal{C}(\mathcal{E})\rightarrow\mathcal{S}(\mathcal{E})$ defined by $j(C)=(C,C)$. Note that j is a well-defined injection with $range(j)=\{(C,P)\in\mathcal{S}(\mathcal{E}) \mid C=P\}$. Now for $C\in\mathcal{C}(\mathcal{E})$, A a multiset over *act*, and $a_1 \cdots a_n \in Act$ an arbitrary enumeration of A, it is easily obtained that

$$\exists C'\colon C \xrightarrow{A}_{\mathcal{E}} C' \wedge j(C')=(S,T) \;\Leftrightarrow\; j(C) \xrightarrow{a_1^+}_{\mathcal{E}} \cdots \xrightarrow{a_n^+}_{\mathcal{E}} \xrightarrow{a_1^-}_{\mathcal{E}} \cdots \xrightarrow{a_n^-}_{\mathcal{E}} (S,T).$$

From this the required implications follow immediately. In order to prove the remaining six implications, first consider the implications between equivalences on $\mathbf{E}(Act^{\pm})$ displayed in Figure 8.

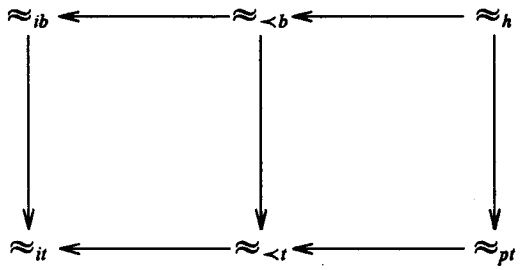


FIGURE 8. *Some semantic equivalences on* $\mathbf{E}(Act^{\pm})$

These implications follow immediately from the definitions. The proofs in [6] that $\approx_{pt}$ and $\approx_h$ are preserved under refinement can be trivially extended to a setting with Λ-refinements for any labelling set Λ. So it follows that

$$\mathcal{E} \approx_{pt} \mathcal{F} \Rightarrow split(\mathcal{E}) \approx_{pt} split(\mathcal{F}) \text{ and } \mathcal{E} \approx_h \mathcal{F} \Rightarrow split(\mathcal{E}) \approx_h split(\mathcal{F}).$$

Now the remaining six implications on $\mathbf{E}(Act)$ follow from Propositions 5.1 and 5.3.

In order to prove the absence of other implications, it suffices to provide counterexamples against $\approx_{pt} \rightarrow \approx_{ib}$, $\approx_{ib} \rightarrow \approx_{st}$, $\approx_{sb} \rightarrow \approx_{2t}$, $\approx_{2b} \rightarrow \approx_{STt}$ and $\approx_{STb} \rightarrow \approx_{pt}$. The first two counterexamples where given already in Section 3. For the third counterexample consider the two event structures of Figure 5. In Section 3 it was established already that they are step bisimulation equivalent. Furthermore they are not split trace equivalent, since $a^+\ c^+\ a^-\ b^+\ c^-\ b^-$ is a split trace of the first one but not of the second one.

FIGURE 9. *ST-bisimulation equivalent but not pomset trace equivalent (A variant of Example 7.1.2.a.ii of [8]).*

The fourth counterexample will be provided in [9]. For the last counterexample consider the two systems represented in Figure 9. Both systems perform the actions a and b exactly once. In the first

system these actions can only be independent, whereas in the second one b can be executed either dependent or independent of a. The difference between the two systems does not occur before (and unless) they reach a state where the execution of a is completed and the execution of b is not yet begun. However, in this state both systems have exactly the same future, consisting of exactly one occurrence of b. Hence they are identified in ST-bisimulation semantics. On the other hand the pomset $a \rightarrow b$ is a pomset trace of the second system, but not of the first. So the two systems are not pomset trace equivalent. This example also shows that ST-semantics does not respect causality. □

6. The refinement theorems

Finally I will prove the announced refinement theorems for ST-semantics. In Van Glabbeek & Vaandrager [9] it will be shown that such a theorem does not hold for split semantics.

Theorem: *Let $\mathcal{E},\mathcal{F} \in \mathbf{E}$ and r be a refinement. Then* $\mathcal{E} \approx_{STb} \mathcal{F} \Rightarrow r(\mathcal{E}) \approx_{STb} r(\mathcal{F})$.

Proof: Let $R \subseteq \mathcal{S}(E_{\mathcal{E}}) \times \mathcal{S}(E_{\mathcal{F}}) \times \mathcal{P}(E_{\mathcal{E}} \times E_{\mathcal{F}})$ be an ST-bisimulation between $\mathcal{E}$ and $\mathcal{F}$. Define the relation $\tilde{R}$ by:

$$\tilde{R} = \{((\tilde{C},\tilde{P}),(\tilde{D},\tilde{Q}),\tilde{f}) \in \mathcal{S}(E_{r(\mathcal{E})}) \times \mathcal{S}(E_{r(\mathcal{F})}) \times \mathcal{P}(E_{r(\mathcal{E})} \times E_{r(\mathcal{F})}) \mid \exists ((C,P),(D,Q),f) \in R$$

$$\text{such that } r^{-1}(\tilde{C},\tilde{P}) = (C,P),\ r^{-1}(\tilde{D},\tilde{Q}) = (D,Q)$$

$$\text{and } \tilde{f} : \tilde{C} \rightarrow \tilde{D} \text{ is a bijection, satisfying } \tilde{f}(e,e') = (f(e),e') \text{ and } \tilde{f}(\tilde{P}) = \tilde{Q}\ \}.$$

I show that $\tilde{R}$ is an ST-bisimulation between $r(\mathcal{E})$ and $r(\mathcal{F})$.

i. $((\varnothing,\varnothing),(\varnothing,\varnothing),\varnothing) \in \tilde{R}$ since $((\varnothing,\varnothing),(\varnothing,\varnothing),\varnothing) \in R$.

ii. Suppose $((\tilde{C},\tilde{P}),(\tilde{D},\tilde{Q}),\tilde{f}) \in \tilde{R}$. Take $((C,P),(D,Q),f) \in R$ such that $r^{-1}(\tilde{C},\tilde{P}) = (C,P)$, $r^{-1}(\tilde{D},\tilde{Q}) = (D,Q)$ and $\tilde{f} : \tilde{C} \rightarrow \tilde{D}$ is a bijection, satisfying $\tilde{f}(e,e') = (f(e),e')$ and $\tilde{f}(\tilde{P}) = \tilde{Q}$. Now three things have to be established:

1. $\tilde{f} : \tilde{C} \rightarrow \tilde{D}$ is a bijection, satisfying $l_{r(\mathcal{F})}(\tilde{f}(e,e')) = l_{r(\mathcal{E})}(e,e')$ and $\tilde{f}(\tilde{P}) = \tilde{Q}$.
2. $(\tilde{C},\tilde{P}) \xrightarrow{}_{r(\mathcal{E})} (\tilde{C}',\tilde{P}') \Rightarrow \exists \tilde{D}',\tilde{Q}',\tilde{f}'$ with $(\tilde{D},\tilde{Q}) \xrightarrow{}_{r(\mathcal{F})} (\tilde{D}',\tilde{Q}')$, $((\tilde{C}',\tilde{P}'),(\tilde{D}',\tilde{Q}'),\tilde{f}') \in \tilde{R}$ and $\tilde{f}' \restriction \tilde{C} = \tilde{f}$.
3. $(\tilde{D},\tilde{Q}) \xrightarrow{}_{r(\mathcal{F})} (\tilde{D}',\tilde{Q}') \Rightarrow \exists \tilde{C}',\tilde{P}',\tilde{f}'$ with $(\tilde{C},\tilde{P}) \xrightarrow{}_{r(\mathcal{E})} (\tilde{C}',\tilde{P}')$, $((\tilde{C}',\tilde{P}'),(\tilde{D}',\tilde{Q}'),\tilde{f}') \in \tilde{R}$ and $\tilde{f}' \restriction \tilde{C} = \tilde{f}$.

ad 1. By construction $\tilde{f} : \tilde{C} \rightarrow \tilde{D}$ is a bijection, satisfying $\tilde{f}(\tilde{P}) = \tilde{Q}$.
Moreover $l_{r(\mathcal{F})}(\tilde{f}(e,e')) = l_{r(\mathcal{F})}(f(e),e') = l_{r(l_{\mathcal{E}}(f(e)))}(e') = l_{r(l_{\mathcal{E}}(e))}(e') = l_{r(\mathcal{E})}(e,e')$.

ad 2. Suppose $(\tilde{C},\tilde{P}) \xrightarrow{}_{r(\mathcal{E})} (\tilde{C}',\tilde{P}')$, i.e. $(\tilde{C}',\tilde{P}') \in \mathcal{S}(r(\mathcal{E}))$, $\tilde{C} \subseteq \tilde{C}'$ and $\tilde{P} \subseteq \tilde{P}'$.
Let $(C',P') = r^{-1}(\tilde{C}',\tilde{P}')$. Using Lemma 4.ii, $(C,P) \xrightarrow{}_{\mathcal{E}} (C',P')$. Since R is an ST-bisimulation, $\exists D',Q',f'$ with $(D,Q) \xrightarrow{}_{\mathcal{F}} (D',Q')$, $((C',P'),(D',Q'),f') \in R$ and $f' \restriction C = f$.
Let $\tilde{D}' = \{(f'(e),e') \mid (e,e') \in \tilde{C}'\}$,
$\tilde{Q}' = \{(f'(e),e') \mid (e,e') \in \tilde{P}'\}$ and
$\tilde{f}' = \{((e,e'),(f'(e),e')) \mid (e,e') \in \tilde{C}'\}$.

For $e \in pr_1(\tilde{C}')$ let $C_e = \{e' \mid (e,e') \in \tilde{C}'\}$ and $P_e = \{e' \mid (e,e') \in \tilde{P}'\}$;
for $d \in pr_1(\tilde{D}')$ let $D_d = \{e' \mid (d,e') \in \tilde{D}'\}$ and $Q_d = \{e' \mid (d,e') \in \tilde{Q}'\}$.
Remark that $Q_{f'(e)} = \{e' \mid (f'(e),e') \in \tilde{Q}'\} = \{e' \mid (e,e') \in \tilde{P}'\} = P_e$ and similarly $D_{f'(e)} = C_e$.
I prove that $(\tilde{D}, \tilde{Q}) \longrightarrow_{r(\mathfrak{F})} (\tilde{D}', \tilde{Q}')$, $((\tilde{C}', \tilde{P}'), (\tilde{D}', \tilde{Q}'), \tilde{f}') \in \tilde{R}$ and $\tilde{f}' \restriction \tilde{C} = \tilde{f}$.

I start with proving that $(\tilde{D}', \tilde{Q}') \in \mathbb{S}(r(\mathfrak{F}))$.

$$pr_1(\tilde{D}') = \{f'(e) \mid e \in pr_1(\tilde{C}')\} = f'(C') = D' \tag{1}$$

so $\tilde{D}' = \{(d,e') \mid d \in D', e' \in D_d\}$ and $\tilde{Q}' = \{(d,e') \mid d \in D', e' \in Q_d\}$.
Using Lemma 4.i, it is then sufficient to show that

(D',Q') is an ST-configuration of $\mathfrak{F}$,
(D_d, Q_d) is an ST-configuration of $r(l_{\mathfrak{F}}(d))$ for $d \in D'$,
$Q_d = E_{r(l_{\mathfrak{F}}(d))}$ iff $d \in Q'$. (2)

The first requirement is already implicit in $(D,Q) \longrightarrow_{\mathfrak{F}} (D',Q')$.
Since $D' = f'(C')$ one may substitute $f'(e)$ for d and $e \in C'$ for $d \in D'$ in the remaining two requirements.
Since $D_{f'(e)} = C_e$, $Q_{f'(e)} = P_e$, $l_{\mathfrak{F}}(f'(e)) = l_{\mathfrak{E}}(e)$ and $Q' = f'(P')$ they reduce to

(C_e, P_e) is an ST-configuration of $r(l_{\mathfrak{E}}(e))$ for $e \in C'$ and
$P_e = E_{r(l_{\mathfrak{E}}(e))}$ iff $e \in P'$.

These follow from Lemma 4.i, using that $(\tilde{C}', \tilde{P}') \in \mathbb{S}(r(\mathfrak{E}))$
and $r^{-1}(\tilde{C}', \tilde{P}') = (C', P')$.

Hence $(\tilde{D}', \tilde{Q}') \in \mathbb{S}(r(\mathfrak{F}))$.
Now (1) and (2) above say that $D' = pr_1(\tilde{D}')$ and $Q' = \{d \in D' \mid Q_d = E_{r(l_{\mathfrak{F}}(d))}\}$.
Hence $r^{-1}(\tilde{D}', \tilde{Q}') = (D', Q')$. It follows that $((\tilde{C}', \tilde{P}'), (\tilde{D}', \tilde{Q}'), \tilde{f}') \in \tilde{R}$.
Finally $\tilde{f}' \restriction \tilde{C} = \tilde{f}$, $\tilde{D} \subseteq \tilde{D}'$ and $\tilde{Q} \subseteq \tilde{Q}'$ by construction, using that $f' \restriction C = f$.
With $(\tilde{D}', \tilde{Q}') \in \mathbb{S}(r(\mathfrak{F}))$, it follows that $(\tilde{D}, \tilde{Q}) \longrightarrow_{r(\mathfrak{F})} (\tilde{D}', \tilde{Q}')$.

ad 3. By symmetry. □

THEOREM: *Let $\mathfrak{E}, \mathfrak{F} \in \mathbb{E}$ and r be a refinement. Then $\mathfrak{E} \approx_{STi} \mathfrak{F} \Rightarrow r(\mathfrak{E}) \approx_{STi} r(\mathfrak{F})$.*

PROOF: It suffices to proof $\mathfrak{E} \lesssim_{STi} \mathfrak{F} \Rightarrow r(\mathfrak{E}) \lesssim_{STi} r(\mathfrak{F})$, so let $\mathfrak{E}, \mathfrak{F} \in \mathbb{E}$ with $\mathfrak{E} \lesssim_{STi} \mathfrak{F}$ and let r be a refinement. Suppose in $r(\mathfrak{E})$ there is a chain of ST-configurations

$$(\varnothing, \varnothing) \longrightarrow_{r(\mathfrak{E})} (\tilde{C}_1, \tilde{P}_1) \longrightarrow_{r(\mathfrak{E})} \cdots \longrightarrow_{r(\mathfrak{E})} (\tilde{C}_n, \tilde{P}_n).$$

By Lemma 4.ii there is a chain of ST-configurations

$$(\varnothing, \varnothing) \longrightarrow_{\mathfrak{E}} (C_1, P_1) \longrightarrow_{\mathfrak{E}} \cdots \longrightarrow_{\mathfrak{E}} (C_n, P_n)$$

in $\mathfrak{E}$ with $(C_i, P_i) = r^{-1}(\tilde{C}_i, \tilde{P}_i)$ for $i = 1, \cdots, n$. Hence there must be a chain

$$(\varnothing, \varnothing) \longrightarrow_{\mathfrak{F}} (D_1, Q_1) \longrightarrow_{\mathfrak{F}} \cdots \longrightarrow_{\mathfrak{F}} (D_n, Q_n)$$

in $\mathfrak{F}$ and a bijection $f : C_n \to D_n$, satisfying $l_{\mathfrak{F}}(f(e)) = l_{\mathfrak{E}}(e)$, $f(C_i) = D_i$ and $f(P_i) = Q_i$ for $i = 1, \cdots, n$.

Let $\tilde{D}_i = \{(f(e),e') \mid (e,e') \in \tilde{C}_i\}$,

$\tilde{Q}_i = \{(f(e),e') \mid (e,e') \in \tilde{P}_i\}$ and

$\tilde{f} = \{((e,e'),(f(e),e')) \mid (e,e') \in \tilde{C}_n\}$.

It remains to be shown that

$$(\varnothing,\varnothing) \rightarrow_{r(\mathfrak{H})} (\tilde{D}_1, \tilde{Q}_1) \rightarrow_{r(\mathfrak{H})} \cdots \rightarrow_{r(\mathfrak{H})} (\tilde{D}_n, \tilde{Q}_n)$$

is a chain of ST-configurations in $r(\mathfrak{H})$ and $\tilde{f}: \tilde{C}_n \rightarrow \tilde{D}_n$ is a bijection satisfying $l_{r(\mathfrak{H})}(\tilde{f}(e,e')) = l_{r(\mathfrak{H})}(e,e')$, $\tilde{f}(\tilde{C}_i) = \tilde{D}_i$ and $\tilde{f}(\tilde{P}_i) = \tilde{Q}_i$ for $i = 1, \cdots, n$. The only nontrivial part of this consist of proving that $(\tilde{D}_i, \tilde{Q}_i) \in \mathfrak{S}(r(\mathfrak{H}))$ for $i = 1, \cdots, n$. This goes exactly as in the previous proof. □

Concluding remarks

In this paper ten semantic equivalences for concurrent systems are defined on a domain of labelled event structures, and their interdependencies are classified as indicated in Figure 2 of the introduction. It has been established - in [4, 6] and [9] respectively - that interleaving, step and split equivalences are strictly based on action atomicity. In particular, the owl example of [9] shows that no equivalence that can be localized between split bisimulation and interleaving trace equivalence is preserved under refinement of actions. On the other hand it has been shown - in [4] and [6] - that the two partial order equivalences of Figure 2 *are* preserved under action refinement and thus need not to be based on action atomicity. Now this paper added that also ST-trace and ST-bisimulation equivalence are preserved under refinement. So the borderline is between split and ST-semantics.

It should be remarked that at all places where split semantics was used before it was studied for a restricted class of concurrent systems (Petri nets without autoconcurrency in [8], a subset of CCS in [1, 10] and deterministic event structures in [17]) on which it coincides with ST-semantics. The examples of [9] suggest that outside such a class, split semantics is not an interesting notion. The reason for mentioning it in this paper is that it seems to be a natural simplification of ST-semantics and in order to indicate that for the purposes of this paper this simplification should not be made.

The refinement operation considered in this paper replaced actions by finite, conflict-free, non-empty event structures. As remarked earlier, a generalization to infinite refinements, leaving all definitions the same, is incompatible with the principle of finite causes: try to refine a in

$$a \longrightarrow b \qquad \text{by} \qquad a_1 \longrightarrow a_2 \longrightarrow a_3 \longrightarrow \cdots .$$

If one would drop this principle, there are (at least) two possibilities of interpreting event structures: events which have an infinite set of causes can happen in a finite time, or they can not. The last interpretation is slightly simpler to grasp, more common, and compatible with the view of this paper, in which the behaviour of concurrent systems - together with all semantic equivalences - is explained in terms of finite configurations (or ST-configurations) only. Using this interpretation any 'generalized' event structure can be transformed in an ordinary event structure satisfying the principle of finite causes, by removing all events that have infinitely many causes. A transformed event structure and its original are equivalent with respect to all equivalences of Figure 2. On the domain of 'generalized' event structures one may drop the restriction that refinements need to be finite, and all theorems and definitions of this paper remain valid. In fact also all proofs remain valid, since (except in the proof of Proposition 1.i) the principle of finite causes is never used. However, it can be argued that infinite refinements change the behaviour of the considered systems in a way that cannot be explained by a change in the level of abstraction at which processes are regarded: consider a system

performing the actions a and b one time each, where the occurrence of b is dependent of the occurrence of a (as depicted above); after replacement of a by an infinite event structure, b cannot happen any more; it occurs in no (finite) configuration. Finally notice that it is also possible to describe this type of refinement on the domain of event structures of Section 1 (that is: satisfying the principle of finite causes), by adding to the definition of refinement that after refinement in the sense of Section 1, events with infinitely many causes should be left out.

A generalization to refinements containing conflicts can be obtained analogously as the above generalization to infinite refinements, but is technically more complicated. On the domain of event structures used in this paper, refinements with conflicts are incompatible with the principle of conflict heredity: try to replace a in

$$a \longrightarrow b \qquad \text{by} \qquad a_1 \cdots\cdots a_2.$$

This problem can be solved by moving to a more general form of event structures where the axiom of conflict heredity is dropped, e.g. flow event structures [3]. This will be done in VAN GLABBEEK & GOLTZ [7], where we define a refinement operator for any function $r: Act \to \mathbf{E} - \{0\}$, thus allowing both infinite refinements and refinements with conflicts. I expect that after this generalization all my theorems remain valid. Each flow event structure is equivalent to an event structure in the sense of Section 1 (with respect to any of the equivalences of Figure 2). Hence an alternative solution consists of appending to the definition of refinement some transformation that turns the refined event structure into an equivalent event structure in the sense of Section 1.

Contrary to the previous generalization, a generalization of the refinement operator to *forgetful refinements*, where replacing actions by the empty event structure is allowed, does not seem very natural. Such refinements can drastically change the behaviour of concurrent systems and can not be explained by a change in the level of abstraction at which these systems are regarded [7]. Moreover, unlike the refinement theorems for partial order semantics [6], the refinement theorem for ST-bisimulation semantics does not hold for forgetful refinements, as is demonstrated by the following counterexample.

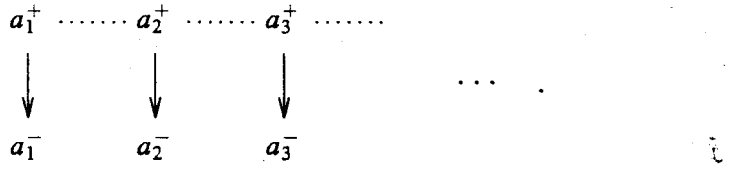
The two event structures above are ST-bisimulation equivalent. However, after replacing a by the empty event structure, the resulting event structures (below) are not ST-bisimulation equivalent.

The refinement theorems for ST-semantics show that in case preservation under refinement is required, it is not necessary to employ partial order semantics. From this the natural question arises if it is necessary to employ at least ST-semantics, i.e. if any equivalence finer then a given interleaving equivalence that is preserved under refinement is also finer then some ST-equivalence. Let $\approx_x$ be an

equivalence on $\mathbb{E}$. Define $\approx_{rx}$ by

$$\mathcal{E} \approx_{rx} \mathcal{F} \text{ iff for all refinements } r:Act \to \mathbb{E}-\{0\} \text{ one has } r(\mathcal{E}) \approx_x r(\mathcal{F}).$$

Then, $\approx_{rx}$ is finer then $\approx_x$ and preserved under refinement. Moreover it is coarser then any other equivalence with these properties. In other words, $\approx_{rx}$ is *fully abstract* with respect to $\approx_x$ and refinement. Of course the definition above is parametrized by the concept of refinement. Let $\approx_{rx}$ be defined under reference to general refinements $r:Act \to \mathbb{E}-\{0\}$, to be elaborated as sketched above; and let $\approx_{r'x}$ be defined under reference to refinements as defined in Section 1. Then I conjecture that $\approx_{STb}$ coincides with $\approx_{rib}$, i.e. ST-bisimulation equivalence is fully abstract with respect to interleaving bisimulation equivalence and action refinement, and also $\approx_{STt}$ coincides with $\approx_{rit}$. To be more precise, let r_c be the refinement that replaces actions $a \in Act$ by



Then I think that $\mathcal{E} \approx_{STb} \mathcal{F} \Leftrightarrow r_c(\mathcal{E}) \approx_{ib} r_c(\mathcal{F})$ and likewise $\mathcal{E} \approx_{STt} \mathcal{F} \Leftrightarrow r_c(\mathcal{E}) \approx_{it} r_c(\mathcal{F})$, from which the conjecture follows. Furthermore, together with Walter Vogler I observed that for finite event structures $\approx_{STb}$ even coincides with $\approx_{r'ib}$. On the other hand $\approx_{r'it}$ is strictly coarser then $\approx_{rit}$, as follows from an example in LARSEN [12], see also [9].

Topics for further research include

- generalizing the refinement operator to infinite refinements and refinements with conflicts, as indicated above (this will be done in [7]),
- defining refinement on other models of concurrency, such as Petri nets, and establishing the correspondence with refinements on event structures (cf. [7]),
- defining 'syntactic refinement' (replacing action symbols by terms in process expressions) on process specification languages, investigating the interaction with communication, proving syntactic refinement theorems and establishing the correspondence with 'semantic refinement', as employed in this paper (cf. [1, 8, 12, 13]),
- proving the full abstraction results conjectured above,
- proving refinement theorems and full abstraction results for the ST-versions of decorated trace semantics - for failure semantics this has been done already in VOGLER [18] in a setting of Petri nets, and for a variant of trace semantics, in the absence of autoconcurrency, modelling a process as a set of semiwords, this has been done in NIELSEN, ENGBERG & LARSEN [13] and LARSEN [12]
- and generalizing the entire theory to a setting with silent actions, or τ-moves.

PRATT [15] and CASTELLANO, DE MICHELIS & POMELLO [4] use the issue of action atomicity as an argument for using partial order semantics instead of interleaving semantics. This paper shows that it is not necessary to employ partial order semantics if one does not want to assume action atomicity; ST-semantics turns out to be sufficient. In VAN GLABBEEK & VAANDRAGER [8] we introduced the (related) criterion of *real-time consistency*. A semantics is real-time consistent if it does not identify systems with a different real-time behaviour. Of course interleaving semantics are not real-time consistent, but again the criterion did not force us to consider partial order semantics: also for this purpose ST-bisimulation semantics turned out to be sufficient. Therefore the question remains whether or not there exists a convincing testing scenario, or some natural operator, that reveals the full distinguishing power of partial order semantics.

REFERENCES

[1] L. ACETO & M. HENNESSY (1989): *Towards action-refinement in process algebras.* In: Proceedings 4th Annual Symposium on Logic in Computer Science (LICS), Asilomar, California, IEEE Computer Society Press, Washington, pp. 138-145.

[2] G. BOUDOL & I. CASTELLANI (1987): *On the semantics of concurrency: partial orders and transition systems.* In: Proceedings TAPSOFT 87, Vol. I (H. Ehrig, R. Kowalski, G. Levi & U. Montenari, eds.), LNCS 249, Springer-Verlag, pp. 123-137.

[3] G. BOUDOL & I. CASTELLANI (1989): *Permutation of transitions: an event structure semantics for CCS and SCCS.* In: Proceedings REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout 1988 (J.W. de Bakker, W.-P. de Roever & G. Rozenberg, eds.), LNCS 354, Springer-Verlag, pp. 411-427.

[4] L. CASTELLANO, G. DE MICHELIS & L. POMELLO (1987): *Concurrency vs Interleaving: an instructive example.* Bulletin of the EATCS 31, pp. 12-15.

[5] P. DEGANO, R. DE NICOLA & U. MONTANARI (1987): *Observational equivalences for concurrency models.* In: Formal Description of Programming Concepts - III, Proceedings of the third IFIP WG 2.2 working conference, Ebberup, Denmark 1986 (M. Wirsing, ed.), Elsevier Science Publishers B.V. (North Holland), pp. 105-129.

[6] R.J. VAN GLABBEEK & U. GOLTZ (1989): *Equivalence notions for concurrent systems and refinement of actions.* Arbeitspapiere der GMD 366, Sankt Augustin, FRG, extended abstract in: Proceedings 14th Symposium on Mathematical Foundations of Computer Science (MFCS), Porąbka-Kozubnik, Poland, August/September 1989 (A. Kreczmar & G. Mirkowska, eds.), LNCS 379, Springer-Verlag, pp. 237-248.

[7] R.J. VAN GLABBEEK & U. GOLTZ (1990): *Refinement of actions in causality based models.* Arbeitspapiere der GMD 428, Sankt Augustin, FRG, to appear in: Proceedings REX Workshop on Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness, Mook 1989 (J.W. de Bakker, W.-P. de Roever & G. Rozenberg, eds.), LNCS, Springer-Verlag, pp. 267-300.

[8] R.J. VAN GLABBEEK & F.W. VAANDRAGER (1987): *Petri net models for algebraic theories of concurrency.* In: Proceedings PARLE conference, Eindhoven, Vol. II (Parallel Languages) (J.W. de Bakker, A.J. Nijman & P.C. Treleaven, eds.), LNCS 259, Springer-Verlag, pp. 224-242.

[9] R.J. VAN GLABBEEK & F.W. VAANDRAGER (1989): *The difference between splitting in n and n + 1,* in preparation.

[10] M. HENNESSY (1988): *Axiomatising finite concurrent processes.* SIAM Journal on Computing 17(5), pp. 997-1017.

[11] L. LAMPORT (1986): *On interprocess communication.* Distributed Computing 1, pp. 77-101.

[12] K.S. LARSEN (1988): *A fully abstract model for a process algebra with refinements.* Master Thesis, Aarhus University, Denmark.

[13] M. NIELSEN, U. ENGBERG & K.S. LARSEN (1989): *Fully abstract models for a process language with refinement.* In: Proceedings REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout 1988 (J.W. de Bakker, W.-P. de Roever & G. Rozenberg, eds.), LNCS 354, Springer-Verlag, pp. 523-548.

[14] M. NIELSEN, G.D. PLOTKIN & G. WINSKEL (1981): *Petri nets, event structures and domains, part I.* Theoretical Computer Science 13(1), pp. 85-108.

[15] V.R. PRATT (1986): *Modelling concurrency with partial orders.* International Journal of Parallel Programming 15(1), pp. 33-71.

[16] A. RABINOVICH & B.A. TRAKHTENBROT (1988): *Behavior Structures and Nets.* Fundamenta Informaticae 11(4), pp. 357-404.

[17] F.W. VAANDRAGER (1988): *Determinism → (event structure isomorphism = step sequence equivalence).* Report CS-R8839, Centrum voor Wiskunde en Informatica, Amsterdam, to appear in: Theoretical Computer Science.

[18] W. VOGLER (1989): *Failures semantics based on interval semiwords is a congruence for refinement.* Bericht TUM-I8905, Institut für Informatik, Technische Universität München, to appear in: Proceedings STACS 90, LNCS, Springer-Verlag.