Theoretical Computer Science 79 (1991) 275-294
North-Holland

Determinism → (event structure isomorphism = step sequence equivalence)

Frits W. Vaandrager*

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

Communicated by M. Nivat
Received May 1989

Abstract

Vaandrager, F.W., Determinism → (event structure isomorphism = step sequence equivalence), Theoretical Computer Science 79 (1991) 275-294.

A concurrent system S is called *deterministic* if for all states s of S we have that whenever S can evolve from state s into states s' and s'' by doing an action a, it must be the case that s' equals s''. It is well known that for deterministic concurrent systems, most of the interleaved equivalences (bisimulation-, failure-, trace-equivalence) coincide. In this paper we prove in the setting of event structures that also most of the non-interleaved equivalences coincide (with each other) on this domain. In the last section of the paper we show that, as a consequence of our result, the causal structure of a deterministic concurrent system can be unravelled by observers who are capable of observing the beginning and termination of events.

1. Introduction

A (discrete) concurrent system generates events as it evolves in time. At any moment a set of events will have occurred and these will be ordered "in time" or by "causal precedence". This order may be partial. When modelling concurrent systems and reasoning about their behaviour, it is often useful to consider different events as occurrences of the same action. This may indicate that certain events are produced by the same physical resource or that they cannot be distinguished by an observer. The relation between events and actions can be expressed by a labelling

* Partial support received from the European Communities under ESPRIT project no. 432, An Integrated Formal Approach to Industrial Software Development (METEOR).

function $l: E \to A$ that relates an action to each event. Different approaches to the modelling of concurrent systems can be classified by looking at the types of labelling functions they allow for. For instance, if one models a concurrent system with an elementary net system [24], then it can never be the case that in some behaviour two events with the same label are concurrent (i.e. not related by the ordering). If we consider the usual semantics for process algebra languages like CCS [17], TCSP [14], ACP [4] and MEIJE [3], then it turns out that these languages are very liberal with respect to labellings of events: there is (almost) no restriction at all. There exists a very rich theory of "comparative concurrency semantics" relating the interleaved semantics for CCS-like languages, i.e. those semantics which do not treat concurrency as a primitive notion. Now a well-known result says that almost all these equivalences (bisimulation equivalence, trace equivalence and everything in between) coincide for deterministic systems (see for instance [9]). A concurrent system S is called *deterministic* if for all states s of S we have that whenever S can evolve from state s into states s' and s'' by doing an action a, it must be the case that s' equals s''.

Recently, many equivalences have been proposed that do consider concurrency as a primitive notion. Besides the event structure equivalence and the step sequence equivalence that will be discussed in this paper, we have for instance occurrence net equivalence [18], NMS equivalence [8], BS bisimulation [27], step failure semantics [26], step bisimulation semantics [19], pomset semantics [22], pomset bisimulation semantics [6], generalised pomset bisimulation and ST-bisimulation [11], split sequence equivalence which we present at the end of this paper, etc.

Now one can ask the obvious question what happens with all these equivalences if we restrict ourselves to the domain of deterministic systems. The main result of this paper is that almost all non-interleaved equivalences coincide (with each other) for deterministic systems. More specifically, we will show that step sequence equivalence and event structure isomorphism agree on this domain. Of the equivalences mentioned above only occurrence net equivalence is not situated in between step sequence equivalence and event structure isomorphism.

1.1. Event structures

A natural domain for modelling concurrency is the class of *event structures*, which were introduced in [18]. By now many different types of event structures have been defined. For an overview we refer to [28]. In our view, an especially important class of event structures is the class of prime event structures. Prime event structures contain no *junk*: every event in the set of events of a prime event structure will occur in at least one behaviour. The event structures used in this paper are labelled prime event structures with binary conflict. Below we give a formal definition of this type of event structure, followed by some explanatory remarks. If one assumes binary conflict, then one can only express that *two* events exclude each other. Thus it is not possible to say that three or more events cannot occur in combination even though each proper subset can. For this one needs more general types of event structures. The assumption of binary conflict is not essential in the proof of the main theorem of this paper. Because most people will be more familiar with event structures with binary conflicts and because the main use we foresee of our theorem lies in the field of CCS-like languages (where conflict is always binary), we decided to present the theorem for the case with binary conflict only, and to leave the generalisation to the case with arbitrary conflict as a (simple) exercise to the reader.

1.2. Arbitrary interleaving versus True concurrency

In the last section of the paper some consequences will be discussed of our result for the issue of arbitrary interleaving versus "True" concurrency. We introduce an operator which splits each event into a beginning and an end and show that the causal structure of a deterministic concurrent system can be unravelled by observers who are capable of observing these beginnings and ends.

1.3. Related work

One can view the main theorem of this paper as a *retrievability* result: given the step sequences of a deterministic event structure, we can retrieve this event structure up to isomorphism. Within the theory of concurrency there are quite a number of other retrievability results. Best and Devillers [5] prove various retrievability results for Petri nets. Kiehn [15] describes how the partial language of a p/t net can be recovered from the set of its step sequences. Shields [25] considers a subclass of deterministic systems (behaviour systems with conservative labelling) which makes it possible to lift concurrency up to a relation on labels, just as in Mazurkiewicz's trace theory [16]. In both cases the partial order structure of a system can be retrieved from firing sequences (or words) and the concurrency relation. In [27], some retrievability results are proved for "behaviour structures".

In this paper we investigate the effect of assuming determinism on the lattice of equivalences in between sequence/trace equivalence and event structure isomorphism. In the course of the discussion we will sketch parts of this lattice: we will define a number of equivalences and establish their mutual relationships. Hence our paper can be viewed as a contribution to the research area of comparative concurrency semantics. Related work on this topic has been reported in [21, 11, 1].

2. Event structures

Definition 2.1. A (*labelled*) *event structure* (*over an alphabet* A) is a 4-tuple $(E, \leqslant, \#, l)$, where
- E is a set of *events*;
- $\leqslant \subseteq E \times E$ is a partial order satisfying the principle of *finite causes*:

$$\{e' \in E \mid e' \leqslant e\} \text{ is finite for all } e \in E;$$

- $\# \subseteq E \times E$ is an irreflexive, symmetric relation (the *conflict relation*) satisfying the principle of *conflict heredity*:

$$e_1 \# e_2 \leqslant e_3 \Rightarrow e_1 \# e_3;$$

- $l: E \to A$ is a *labelling function*.

As usual we write $e' < e$ for $e' \leqslant e \wedge e' \neq e$, $\geqslant$ for $\leqslant^{-1}$, and $>$ for $<^{-1}$. We use $\smile$ to denote the relation $E \times E - (\leqslant \cup \geqslant \cup \#)$. $\smile$ is called the *concurrency relation*. By definition $<, =, >, \#$ and $\smile$ form a partition of $E \times E$.

Remark 2.2. The components of an event structure $\mathbf{E}$ will be denoted, respectively, by $E_{\mathbf{E}}, \leqslant_{\mathbf{E}}, \#_{\mathbf{E}}$ and $l_{\mathbf{E}}$. The derived relations will be denoted $\smile_{\mathbf{E}}, <_{\mathbf{E}}, >_{\mathbf{E}}, \geqslant_{\mathbf{E}}$. For $e \in E_{\mathbf{E}}$, $\mathrm{pre}_{\mathbf{E}}(e)$ denotes the set of events which precede e in the ordering (so $\mathrm{pre}_{\mathbf{E}}(e) = \{e' \in E_{\mathbf{E}} \mid e' \leqslant_{\mathbf{E}} e\}$).

In the graphical representation we either depict the events or their labels, depending on what we want to illustrate. The partial order relation is indicated by arrows. The conflict relation is denoted by means of dotted lines. If we draw no relation between events they are concurrent, unless, by means of the transitive and reflexive closure of the arrows, it can be deduced that they are ordered, or, by means of the principle of conflict heredity, it can be deduced that they are in conflict.

Example 2.3. Let the event structure $\mathbf{E}$ be given by:

$$E_{\mathbf{E}} = \{e_1, e_2, e_3, e_4, e_5\},$$
$$\leqslant_{\mathbf{E}} = \{(e_1, e_2), (e_1, e_3), (e_2, e_3)\} \cup \{(e, e) \mid e \in E_{\mathbf{E}}\},$$
$$\#_{\mathbf{E}} = \{(x, e_4), (e_4, x) \mid x \in \{e_1, e_2, e_3\}\},$$
$$l_{\mathbf{E}}(e_i) = a_i.$$

Graphically we can depict $\mathbf{E}$ as shown in Fig. 1.

2.1. Operational meaning of event structures

The events in an event structure can be anything varying from a clock pulse in a computer, the printing of a file, my act of writing this article, your act of reading it, the next crash of Wall Street, etc.

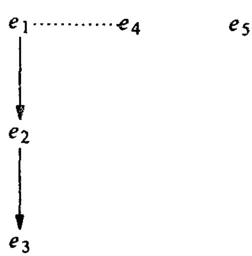

Fig. 1.

The partial order relation expresses that some events are causally related to other events or that for all observers the occurrence of certain events will be seen to precede the occurrence of others. For instance, my act of writing this article will precede your act of reading it. On the other hand, your act of reading this article will probably not be causally related to the next crash of Wall Street. The question what, in general, constitutes a causal link, is a metaphysical one and difficult to answer. However, in a lot of practical situations it is perfectly clear what we mean with causality and reasoning about the behaviour of concurrent systems in terms of causality is useful.

The principle of finite causes says that the systems we consider are discrete and that moreover we do not consider situations like those shown in Figs. 2 and 3. In Fig. 2 it is not clear that any of the e_i can ever happen. In Fig. 2, e_∞ can occur if execution of *all* events $e_1, e_2, \ldots$ finishes after a finite amount of time. Because we do not make any assumptions about the time it takes to perform an event, it is possible that e_1 takes 1 s, e_2 takes 2 s, etc. In that case e_∞ will never take place.

If two events are in conflict, then at most one of them can occur. As a consequence of the principle of conflict heredity we have that when an event occurs, all its "causes" must have occurred before. So if two events e and e' are related in the ordering, say $e < e'$, then occurrence of e is a prerequisite for the occurrence of e'. In general it is not the case that after occurrence of e the occurrence of e' is inevitable. It would be possible to allow event structures where one event has two causes, which are in conflict. Two interpretations of the event structure shown in Fig. 4 are possible: either one can say that e_3 will never occur because it is impossible that all its causes occur (in that case one can just as well leave e_3 out of the event

Fig. 2.

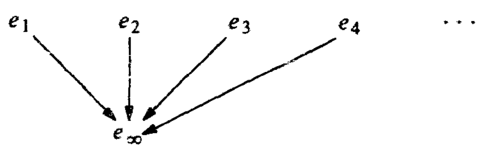

Fig. 3.

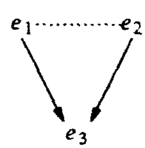

Fig. 4.

structure and adopt the principle of conflict heredity), or one can say that e_3 can occur if a maximal, conflict-free subset of its causes has occurred, so $\{e_1\}$ *or* $\{e_2\}$.

There are no fundamental reasons to adopt the principles of finite causes and conflict heredity. We have included them in our definition of event structures because this makes an elegant formulation of the main result of this paper possible.

The operational intuitions presented in the discussion above, are defined formally below.

Definition 2.4. Let E be an event structure and let X be a subset of $E_{\mathbf{E}}$. We say that X is *left-closed* if

$$e \in X \wedge e' \leqslant_{\mathbf{E}} e \Rightarrow e' \in X.$$

X is *conflict-free* if X does not contain a pair of events which are in conflict, so if $\#_{\mathbf{E}} \cap (X \times X) = \emptyset$. E is *conflict-free* if $\#_{\mathbf{E}} = \emptyset$. A *configuration* of E is a finite, left-closed, conflict-free subset of $E_{\mathbf{E}}$. (Note that Winskel [28] does not require that configurations are finite.) With $\mathscr{C}(\mathbf{E})$ we denote the set of configurations of E.

Example 2.5. Figure 5 depicts all configurations of the event structure of Example 2.3. An arrow is drawn between two configurations if one can be obtained from the other by adding a single event.

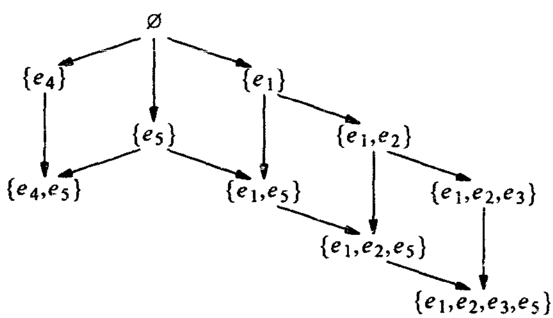

Fig. 5.

Definition 2.6. For any alphabet Σ, we use Σ^* to denote the set of finite sequences over alphabet Σ and Σ^+ to denote the set of finite nonempty sequences over this alphabet. We write λ for the empty sequence and a for the sequence consisting of the single symbol $a \in \Sigma$. By $\sigma * \sigma'$, sometimes abbreviated $\sigma\sigma'$, we denote the concatenation of sequences σ and σ'. On sequences we define a partial ordering $\leqslant$ (the *prefix ordering*) by $\sigma \leqslant \rho$ iff, for some sequence σ', $\sigma\sigma' = \rho$. If $\sigma \leqslant \rho$ we say that σ is a *prefix* of ρ.

Definition 2.7. Let $\mathbf{E}$ be an event structure and let X and Y be configurations of $\mathbf{E}$.

(i) Let $a \in A$. We say that there is an *a-transition* from X to Y, notation $X \rightarrow_{\mathbf{E}}^{a} Y$, if $Y = X \cup \{e\}$ for some event $e \notin X$ with $l_{\mathbf{E}}(e) = a$.

(ii) An action $a \in A$ is *enabled* in X, notation $X \rightarrow_{\mathbf{E}}^{a}$, if $X \rightarrow_{\mathbf{E}}^{a} X'$ for some configuration X'.

(iii) A sequence of actions $\sigma = a_1 * \cdots * a_n \in A^*$ is *enabled* in X, notation $X \rightarrow_{\mathbf{E}}^{\sigma}$, if there exist configurations $X_0, \ldots, X_n$ such that $X = X_0$ and for $1 \leqslant i \leqslant n$: $X_{i-1} \rightarrow_{\mathbf{E}}^{a_i} X_i$. We say that X_n is *obtained from* X *by the occurrence of* σ, notation $X \rightarrow_{\mathbf{E}}^{\sigma} X_n$. We also say that σ is an *(action) sequence* of X.

(iv) A sequence of events $\alpha = e_1 * \cdots * e_n \in E_{\mathbf{E}}^*$ is *enabled* in X, notation $X \rightarrow_{\mathbf{E}}^{\alpha}$, if there exist configurations $X_0, \ldots, X_n$ such that $X = X_0$ and for $1 \leqslant i \leqslant n$: $e_i \notin X_{i-1}$ and $X_i = X_{i-1} \cup \{e_i\}$. We say that α is an *(event) sequence* of X.

(v) With $\mathrm{seq}_{\mathbf{E}}(X)$ we denote the set of action sequences of X, so $\mathrm{seq}_{\mathbf{E}}(X) = \{\sigma \in A^* \mid X \rightarrow_{\mathbf{E}}^{\sigma}\}$.

Proposition 2.8 (no junk). *Let* $\mathbf{E}$ *be an event structure and let* $e \in E_{\mathbf{E}}$. *Then there exists a configuration* X *of* $\mathbf{E}$ *with* $e \in X$.

Proof. Take $X = \mathrm{pre}_{\mathbf{E}}(e)$. Due to the principle of finite causes X is finite. From the fact that $\leqslant_{\mathbf{E}}$ is a partial order it follows that X is left-closed. X is conflict-free due to the principle of conflict heredity. Hence X is a configuration. Clearly $e \in X$. □

3. Three basic equivalences on event structures

We will now define three equivalences on event structures which make increasingly more identifications.

Definition 3.1. An *event structure isomorphism* between two even structures $\mathbf{E}$ and $\mathbf{F}$ is a bijective mapping $f: E_{\mathbf{E}} \rightarrow E_{\mathbf{F}}$ such that:

- $f(e) \leqslant_{\mathbf{F}} f(e') \Leftrightarrow e \leqslant_{\mathbf{E}} e'$,
- $f(e) \#_{\mathbf{F}} f(e') \Leftrightarrow e \#_{\mathbf{E}} e'$, and
- $l_{\mathbf{F}}(f(e)) = l_{\mathbf{E}}(e)$.

$\mathbf{E}$ and $\mathbf{F}$ are *isomorphic*, notation $\mathbf{E} \cong \mathbf{F}$, if there exists an event structure isomorphism between them.

Definition 3.2. Let $\mathbf{E}, \mathbf{F}$ be two event structures. A relation $R \subseteq \mathscr{C}(\mathbf{E}) \times \mathscr{C}(\mathbf{F})$ is a *bisimulation* between $\mathbf{E}$ and $\mathbf{F}$ if:

(1) $\emptyset \, R \, \emptyset$;

(2) If $X \, R \, Y$ and $X \rightarrow_{\mathbf{E}}^{a} X'$ for some $a \in A$, then there exists a $Y' \in \mathscr{C}(\mathbf{F})$ such that $Y \rightarrow_{\mathbf{E}}^{a} Y'$ and $X' \, R \, Y'$;

(3) As (2) but with the roles of X and Y reversed.

$\mathbf{E}$ and $\mathbf{F}$ are *bisimilar*, notation $\mathbf{E} \leftrightarrow \mathbf{F}$, if there exists a bisimulation between them.

Definition 3.3. Two event structures **E** and **F** are *sequence equivalent*, notation $\mathbf{E} \equiv_{\mathrm{seq}} \mathbf{F}$, if:

$$\mathrm{seq}_{\mathbf{E}}(\emptyset) = \mathrm{seq}_{\mathbf{F}}(\emptyset).$$

Remark 3.4. The semantical notion of sequence equivalence, is usually called *trace* equivalence in the settings of process algebra and trace theory as in [23]. However, use of the word trace would be very confusing in a paper on event structures, since event structures are closely related to a completely different type of traces, namely those which are studied in trace theory as in [16]. Therefore we have chosen to use the word "sequence" to denote a finite string of symbols recording the actions in which a process has engaged up to some moment in time.

Proposition 3.5. *$\cong$, $\underline{\leftrightarrow}$ and $\equiv_{\mathrm{seq}}$ are equivalence relations and their relations are $\cong \subset \underline{\leftrightarrow} \subset \equiv_{\mathrm{seq}}$.*

Proof. Standard. $\square$

Example 3.6. The event structures in Fig. 6 show that $\cong$, $\underline{\leftrightarrow}$ and $\equiv_{\mathrm{seq}}$ are really different equivalences. In the graphical representations we have depicted the labels of the events and not the events themselves.

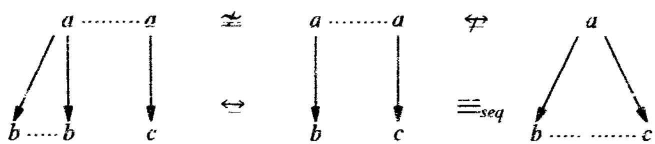

Fig. 6.

The following definition is central to this paper.

Definition 3.7. Let **E** be an event structure. **E** is *deterministic* if for all configurations $X \in \mathscr{C}(\mathbf{E})$ we have that whenever $X \rightarrow^{a}_{\mathbf{E}} Y$ and $X \rightarrow^{a}_{\mathbf{E}} Y'$ for some $a \in A$ and $Y, Y' \in \mathscr{C}(\mathbf{E})$, we have that $Y = Y'$.

So an event structure is deterministic if it does not have a configuration with the property that two different events are enabled which have the same label.

Definition 3.8. Let **E** be an event structure. Two events $e, e' \in E_{\mathbf{E}}$ are in *immediate conflict*, notation $e \#^{1}_{\mathbf{E}} e'$, if they are in conflict and furthermore:

$$e \geqslant_{\mathbf{E}} f \#_{\mathbf{E}} e' \Rightarrow e = f \quad \text{and} \quad e \#_{\mathbf{E}} f \leqslant_{\mathbf{E}} e' \Rightarrow f = e'.$$

Using the notion of immediate conflict we can give a "less operational" characterization of deterministic event structures.

Proposition 3.9. *Let* $\mathbf{E}$ *be an event structure. Then* $\mathbf{E}$ *is deterministic iff*:

$$e \smile_{\mathbf{E}} e' \text{ or } e \#^1_{\mathbf{E}} e' \Rightarrow l_{\mathbf{E}}(e) \neq l_{\mathbf{E}}(e').$$

Proof. Easy. □

It is well known that the linear time-branching time spectrum collapses for deterministic event structures.

Proposition 3.10. *Let* $\mathbf{E}, \mathbf{F}$ *be deterministic event structures. Then* $\mathbf{E} \underline{\leftrightarrow} \mathbf{F} \Leftrightarrow \mathbf{E} =_{\mathrm{seq}} \mathbf{F}$.

Proof. $\Rightarrow$ follows from Proposition 3.5. In order to prove $\Leftarrow$ define a relation $R \subseteq \mathscr{C}(\mathbf{E}) \times \mathscr{C}(\mathbf{F})$ by

$$X \mathrel{R} Y \Leftrightarrow \mathrm{seq}_{\mathbf{E}}(X) = \mathrm{seq}_{\mathbf{F}}(Y).$$

It is easy to show that R gives a bisimulation between $\mathbf{E}$ and $\mathbf{F}$. □

Remark 3.11. In a dictionary [20] we found the following entry for the word "determinism":

(1) a doctrine that all phenomena are determined by preceding occurrences; *esp.* the doctrine that all human acts, choices etc. are causally determined and that free will is illusory;

(2) a belief in predestination.

One may think that the notion of determinism introduced in Definition 3.7 is in conflict with the above description. If one for instance considers the deterministic event structure containing two events labelled a and b which are in conflict, then one may argue that the choice between a and b is not causally determined, that the event structure "has a free will" and "may choose" whether to perform a or b. Therefore one may propose another definition of determinism for event structures which says that an event structure is deterministic iff it is conflict-free. In fact this definition occurs in [1].

We however prefer our own definition because we like to view event structures as "reactive systems". An event structure model of a concurrent system describes how the system reacts to stimuli received from its environment. In the example of the event structure with actions a and b, it is completely determined how a system modelled by this event structure will react to external stimuli: the system has no choice.

Now consider the event structure shown in Fig. 7. This event structure is conflict-free and hence deterministic in the sense of [1]. However, if the environment offers an a, then there is a choice between the "left" a and the "right" a. Depending on how this choice is resolved by the system, it can engage in b or in c afterwards. Hence one can argue that the event structure exhibits nondeterministic behaviour.

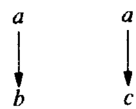

Fig. 7.

4. Noninterleaved equivalences

Many people think that bisimulation equivalence, and consequently also sequence equivalence, make too many identifications on event structures to be of use in general. In bisimulation semantics concurrency is not preserved, i.e. for each event structure we can give a bisimilar event structure with an empty concurrency relation. We elaborate on this below.

Definition 4.1. The *sequentialisation* of an event structure $\mathbf{E}$, notation $\mathcal{S}(\mathbf{E})$, is the event structure $\mathbf{F}$ defined by:

- $E_{\mathbf{F}} = \{\alpha \in (E_{\mathbf{E}})^+ \mid \emptyset \xrightarrow{\alpha}_{\mathbf{E}}\}$;
- $\alpha \leqslant_{\mathbf{F}} \beta$ iff α is a prefix of β;
- $\#_{\mathbf{F}} = (E_{\mathbf{F}} \times E_{\mathbf{F}}) - (\leqslant_{\mathbf{F}} \cup \geqslant_{\mathbf{F}})$;
- $l_{\mathbf{F}}(\alpha * e) = l_{\mathbf{E}}(e)$.

Proposition 4.2. *Let* $\mathbf{E}$ *be an event structure. Then*:
 (i) *the concurrency relation of* $\mathcal{S}(\mathbf{E})$ *is empty*,
 (ii) $\mathbf{E} \underline{\leftrightarrow} \mathcal{S}(\mathbf{E})$,
 (iii) $\mathcal{S}(\mathbf{E}) \cong \mathcal{S}(\mathcal{S}(\mathbf{E}))$.

Proof. Easy. $\square$

4.1. Step semantics

Intuitively, one of the reasons why an event structure is in general different from its sequentialisation is that it sometimes has the possibility of doing a number of events simultaneously in one "step". The notion of a "step" immediately suggests refinements of sequence equivalence and bisimulation equivalence which do not disregard concurrency. These refinements will be called step sequence equivalence and step bisimulation equivalence, respectively. Step sequences have been defined in [10]. Step bisimulations appear in [19]. In [11] they are called "concurrent bisimulations". Below we give the formal definition of step sequence equivalence.

Definition 4.3. Let $\mathbf{E}$ be an event structure and let X and Y be configurations of $\mathbf{E}$.

(i) Let U be a finite subset of $E_{\mathbf{E}}$. We say that Y *U-follows* X, notation $X[U> Y$, if $X \cap U = \emptyset$, the elements of U are pairwise concurrent (so $\forall e, e' \in U$: $e \neq e' \Rightarrow e \smile_{\mathbf{E}} e'$) and $Y = X \cup U$.

(ii) Let $U \subseteq E_{\mathbf{E}}$. We say that U is *enabled* in X (U is a *step* from X), notation $X[U>_{\mathbf{E}}$, if $X[U>_{\mathbf{E}} X'$ for some configuration X' of $\mathbf{E}$.

(iii) A sequence $\alpha = U_1 * \cdots * U_n \in (\mathrm{Pow}(E_{\mathbf{E}}))^*$ is *enabled* in X, notation $X[\alpha>_{\mathbf{E}}$, if there exist configurations $X_0, \ldots, X_n$ such that $X = X_0$ and for $1 \leq i \leq n$: $X_{i-1}[U_i>_{\mathbf{E}} X_i$. We say that X_n *is obtained from X by the occurrence of α*, notation $X[\alpha>_{\mathbf{E}} X_n$. We also say that α is an (*event*) *step sequence of* X.

(iv) Let $\alpha = U_1 * \cdots * U_n \in (\mathrm{Pow}(E_{\mathbf{E}}))^*$ such that $X[\alpha>_{\mathbf{E}} Y$. Let σ be the sequence $l_{\mathbf{E}}(U_1) * \cdots * l_{\mathbf{E}}(U_n)$ where $l_{\mathbf{E}}(U_i)$ denotes the multiset of labels of events in U_i. We say that σ is *enabled* in X, notation $X[\sigma>_{\mathbf{E}}$. We also say that σ is an (*action*) *step sequence* of X, and that Y is *obtained from X by the occurrence of σ*, notation $X[\sigma>_{\mathbf{E}} Y$.

(v) With $\mathrm{step}_{\mathbf{E}}(X)$ we denote the set of action step sequences of X, so $\mathrm{step}_{\mathbf{E}}(X) = \{\sigma \in (\mathrm{Mul}(A))^* \mid X[\sigma>_{\mathbf{E}}\}$.

Definition 4.4. Two event structures $\mathbf{E}$ and $\mathbf{F}$ are *step sequence equivalent*, notation $\mathbf{E} \equiv_{\mathrm{step}} \mathbf{F}$, if:

$$\mathrm{step}_{\mathbf{E}}(\emptyset) = \mathrm{step}_{\mathbf{F}}(\emptyset).$$

Proposition 4.5. *$\equiv_{\mathrm{step}}$ is an equivalence relation. The following relations hold between the equivalences presented thus far:*

$$\begin{array}{ccc} \cong & \subset & \underline{\leftrightarrow} \\ \cap & & \cap \\ \equiv_{\mathrm{step}} & \subset & \equiv_{\mathrm{seq}} \end{array}$$

Proof. Easy. $\square$

Examples 4.6. We give some examples which show that the diagram above gives *all* relations between the equivalences. Our first example (Fig. 8) shows that step

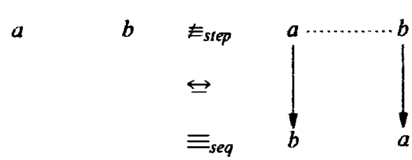

Fig. 8.

semantics (at least sometimes) takes concurrency as a primitive notion. The two leftmost event structures in Fig. 6 are not isomorphic but they are step sequence equivalent. This follows from the observation that on the domain of event structures with empty concurrency relation, step sequence equivalence and sequence equivalence coincide.

The two rightmost event structures in Fig. 6 are not bisimilar, but they are step sequence equivalent.

4.2. Partial order semantics

An *A-labelled partially ordered set* is a triple $(X, \leqslant, l)$ with X a set, $\leqslant$ a partial order on X, and $l: X \to A$ a labelling function. Two such sets $(X_0, \leqslant_0, l_0)$ and $(X_1, \leqslant_1, l_1)$ are *isomorphic* if there exists a bijective mapping $f: X_0 \to X_1$ such that $f(x) \leqslant_1 f(y) \Leftrightarrow x \leqslant_0 y$ and $l_1(f(x)) = l_0(x)$. A *partially ordered multiset* (*pomset*) is an isomorphism class of labelled partially ordered sets. As usual, pomsets can be made setlike by requiring that the events in the partial orders should be chosen from a given set. Below we will view equivalence classes of conflict-free event structures as pomsets.

Definition 4.7. The *restriction* of an event structure $\mathbf{E}$ to a set $X \subseteq E_{\mathbf{E}}$ of events is the event structure $\mathbf{E} \upharpoonright X = (X, \leqslant_{\mathbf{E}} \cap (X \times X), \#_{\mathbf{E}} \cap (X \times X), l_{\mathbf{E}} \upharpoonright X)$.

Definition 4.8. Let $\mathbf{E}$ be an event structure and let X be a configuration of $\mathbf{E}$. The set of *pomsets* of X, notation $\mathrm{pom}_{\mathbf{E}}(X)$, is defined by:

$$\mathrm{pom}_{\mathbf{E}}(X) = \{(\mathbf{E} \upharpoonright (X' - X))/_{\cong} \mid X \subseteq X' \in \mathscr{C}(\mathbf{E})\}.$$

Definition 4.9. Two event structures $\mathbf{E}$ and $\mathbf{F}$ are *pomset equivalent*, notation $\mathbf{E} \equiv_{\mathrm{pom}} \mathbf{F}$, if $\mathrm{pom}_{\mathbf{E}}(\emptyset) = \mathrm{pom}_{\mathbf{F}}(\emptyset)$.

The first systematic study of pomsets is in [12], where they are called *partial words*. Pomset semantics is advocated in [22].

Proposition 4.10. *$\equiv_{\mathrm{pom}}$ is an equivalence relation. It fits in our semantical lattice as follows*:

$$\begin{array}{ccccc} \cong & & \subset & & \underline{\leftrightarrow} \\ \cap & & & & \cap \\ \equiv_{\mathrm{pom}} & \subset & \equiv_{\mathrm{step}} & \subset & \equiv_{\mathrm{seq}} \end{array}$$

Examples 4.11. The two rightmost event structures in Fig. 6 provide an example of two event structures which are identified in pomset semantics, but distinguished in bisimulation semantics. The remaining examples distinguishing pomset equivalence and the other equivalences are displayed in Fig. 9. The example of Fig. 10 is interesting because it only contains conflict-free event structures, and also because it disproves Theorem 3.5 in [1]. Notice that all these examples contain nondeterministic event structures.

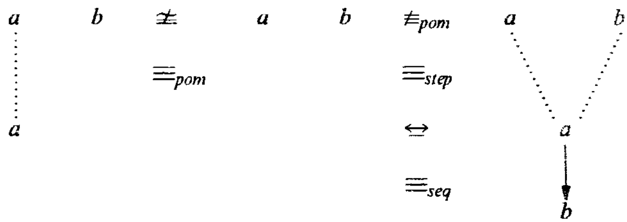

Fig. 9.

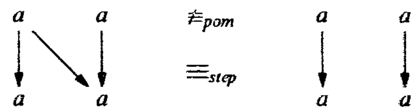

Fig. 10.

5. Determinism → (event structure isomorphism = step sequence equivalence)

Proposition 3.10 stated that bisimulation equivalence and sequence equivalence coincide on the domain of deterministic event structures. Surprisingly, most of the noninterleaved semantics which have been proposed in the literature, also coincide on this domain.

In the introduction of this paper we mentioned a large number of equivalences which are situated in between event structure isomorphism and step sequence equivalence. As a consequence of the following result *all* these equivalences (except for occurrence net equivalence) coincide with event structure isomorphism on the domain of deterministic event structures.

Theorem 5.1. *Let* $\mathbf{E}, \mathbf{F}$ *be deterministic event structures. Then* $\mathbf{E} \cong \mathbf{F} \Leftrightarrow \mathbf{E} \equiv_{\text{step}} \mathbf{F}$.

Lemma 5.2. *Let* $\mathbf{E}$ *be a deterministic event structure and let* X, Y *be configurations of* $\mathbf{E}$ *such that* $\mathbf{E} \restriction X \cong \mathbf{E} \restriction Y$. *Then* $X = Y$.

Proof. Induction on the size of X. If X is the empty set, then Y must be empty too and we are done. Suppose X is nonempty. Let e be a maximal element of X and let $X' = X - \{e\}$. Now we use that there exists an event structure isomorphism f between $\mathbf{E} \restriction X$ and $\mathbf{E} \restriction Y$: we have $\mathbf{E} \restriction X' \cong \mathbf{E} \restriction Y'$ for $Y' = Y - \{f(e)\}$ and furthermore X' and Y' are configurations. Applying the induction hypothesis gives $X' = Y'$. Let $a = l_{\mathbf{E}}(e) = l_{\mathbf{E}}(f(e))$. We have that $X' \rightarrow^{a}_{\mathbf{E}} X$ but also $X' \rightarrow^{a}_{\mathbf{E}} Y$. Now use that $\mathbf{E}$ is deterministic to obtain that $X = Y$. $\square$

Lemma 5.3. *Let* $\mathbf{E}$ *and* $\mathbf{F}$ *be deterministic event structures. Then* $\mathbf{E} \equiv_{\text{pom}} \mathbf{F} \Leftrightarrow \mathbf{E} \cong \mathbf{F}$.

Proof. $\Leftarrow$ is trivial, so the interesting direction is $\Rightarrow$. Define relation $\sim \subseteq E_{\mathbf{E}} \times E_{\mathbf{F}}$ by

$$e_0 \sim e_1 \Leftrightarrow_{\text{def}} \mathbf{E} \upharpoonright \text{pre}_{\mathbf{E}}(e_0) \cong \mathbf{F} \upharpoonright \text{pre}_{\mathbf{F}}(e_1).$$

We claim that $\sim$ gives a bijective mapping between $E_{\mathbf{E}}$ and $E_{\mathbf{F}}$. Because $\mathbf{E} \equiv_{\text{pom}} \mathbf{F}$, it is obvious that $\text{dom}(\sim) = E_{\mathbf{E}}$ and $\text{range}(\sim) = E_{\mathbf{F}}$. Suppose that $e_0 \sim e_1$ and $e_0 \sim e_1'$. We show that $e_1 = e_1'$. By definition we have $\mathbf{E} \upharpoonright \text{pre}_{\mathbf{E}}(e_0) \cong \mathbf{F} \upharpoonright \text{pre}_{\mathbf{F}}(e_1) \cong \mathbf{F} \upharpoonright \text{pre}_{\mathbf{F}}(e_1')$. Application of the previous lemma gives $\text{pre}_{\mathbf{F}}(e_1) = \text{pre}_{\mathbf{F}}(e_1')$. Since both sets have a unique maximal element, these maximal elements must be identical: $e_1 = e_1'$. In the same way we can prove that if $e_0 \sim e_1$ and $e_0' \sim e_1$, this implies $e_0 = e_0'$. Hence $\sim$ gives a bijection between $E_{\mathbf{E}}$ and $E_{\mathbf{F}}$. It is routine to check that this bijection is in fact an event structure isomorphism. □

Proof of Theorem 5.1. From the previous results it follows that in order to prove Theorem 5.1 it is enough to show that for deterministic event structures $\mathbf{E}$, $\mathbf{F}$,

$$\mathbf{E} \equiv_{\text{step}} \mathbf{F} \Rightarrow \mathbf{E} \equiv_{\text{pom}} \mathbf{F}.$$

By definition this is equivalent to

$$\text{step}_{\mathbf{E}}(\emptyset) = \text{step}_{\mathbf{F}}(\emptyset) \Rightarrow \text{pom}_{\mathbf{E}}(\emptyset) = \text{pom}_{\mathbf{F}}(\emptyset).$$

We will prove a slightly stronger statement, namely,

$$\forall X \in \mathscr{C}(\mathbf{E})\ \forall Y \in \mathscr{C}(\mathbf{F}):\quad \text{step}_{\mathbf{E}}(X) = \text{step}_{\mathbf{F}}(Y) \Rightarrow \text{pom}_{\mathbf{E}}(X) = \text{pom}_{\mathbf{F}}(Y).$$

Let $X \in \mathscr{C}(\mathbf{E})$, $Y \in \mathscr{C}(\mathbf{F})$ with $\text{step}_{\mathbf{E}}(X) = \text{step}_{\mathbf{F}}(Y)$. Let X' be a configuration of $\mathbf{E}$ with $X \subseteq X'$. Let $\alpha_0 = \{e_1\}\{e_2\} \ldots \{e_n\}$ be a sequence of singleton steps such that $X[\alpha_0 >_{\mathbf{E}} X'$ and $X' - X = \{e_1, \ldots, e_n\}$. Let $\alpha_1 = \{e_1'\}\{e_2'\} \ldots \{e_n'\}$ be a step sequence such that $Y[\alpha_1 >_{\mathbf{F}}$ and $l_{\mathbf{E}}(e_i) = l_{\mathbf{F}}(e_i')$ for $1 \leq i \leq n$ (due to the fact that X and Y have the same step sequences, such a sequence will always exist). Let $Y' = Y \cup \{e_1', \ldots, e_n'\}$. We claim that the function which maps e_i to e_i' is an event structure isomorphism between $\mathbf{E} \upharpoonright (X' - X)$ and $\mathbf{F} \upharpoonright (Y' - Y)$. For reasons of symmetry we have proved the theorem if we have shown this.

The proof goes by induction to n. The case with $n = 0$ is trivial. Now suppose $n > 0$. Due to the fact that X and Y have the same step sequences and due to the determinism of $\mathbf{E}$ and $\mathbf{F}$, we have

$$\text{step}_{\mathbf{E}}(X \cup \{e_1\}) = \text{step}_{\mathbf{F}}(Y \cup \{e_1'\}).$$

Since

$$X \cup \{e_1\}[\{e_2\} \ldots \{e_n\} >_{\mathbf{E}} X' \quad \text{and} \quad Y \cup \{e_1'\}[\{e_2'\} \ldots \{e_n'\} >_{\mathbf{E}} Y',$$

we can now apply the induction hypothesis which gives

$$\mathbf{E} \upharpoonright (X' - (X \cup \{e_1\})) \cong \mathbf{F} \upharpoonright (Y' - (Y \cup \{e_1'\})).$$

In order to prove the induction step it is enough to show that for $2 \leq i \leq n$, $e_1 <_{\mathbf{E}} e_i \Leftrightarrow e'_1 <_{\mathbf{F}} e'_i$. If $n = 1$ we are done, so assume $n \geq 2$. Let for some i, e_i be minimal in $\{e_2, \ldots, e_n\}$. Then e'_i is minimal in $\{e'_2, \ldots, e'_n\}$. We claim that $e_1 <_{\mathbf{E}} e_i \Leftrightarrow e'_1 <_{\mathbf{F}} e'_i$. Suppose $e_1 <_{\mathbf{E}} e_i$ but not $e'_1 <_{\mathbf{F}} e'_i$. If we show that this leads to a contradiction we have proved the claim because the remaining case is symmetric. If it is not the case that $e'_1 <_{\mathbf{F}} e'_i$ then $e'_1 \smile_{\mathbf{F}} e'_i$. Due to the minimality of e'_i we have that $Y[\{e'_1, e'_i\} >_{\mathbf{F}}$. Now we use that X and Y have the same step sequences and the fact that $\mathbf{E}$ is deterministic. There must be some f such that $X[\{e_1, f\} >_{\mathbf{E}}$ and $l_{\mathbf{E}}(f) = l_{\mathbf{F}}(e'_i) = l_{\mathbf{E}}(e_i)$. Because $e_1 <_{\mathbf{E}} e_i$, $f \neq e_i$. But now there is a contradiction since we can go from configuration $X \cup \{e_1\}$ with an $l_{\mathbf{E}}(f)$-transition to $X \cup \{e_1, f\}$ as well as $X \cup \{e_1, e_i\}$.

Now we have proved that for e_i, which are minimal in $\{e_2, \ldots, e_n\}$, $e_1 <_{\mathbf{E}} e_i \Leftrightarrow e'_1 <_{\mathbf{F}} e'_i$. In order to prove this fact also for e_i which are not minimal, we distinguish between two cases.

(1) For all e_i which are minimal in $\{e_2, \ldots, e_n\}$, we have that $e_1 <_{\mathbf{E}} e_i$. This implies that $e_1 <_{\mathbf{E}} e_l$ for $2 \leq l \leq n$. Further we have that for all e'_i which are minimal in $\{e'_2, \ldots, e'_n\}$, $e'_1 <_{\mathbf{F}} e'_i$. Consequently $e'_1 <_{\mathbf{F}} e'_l$ for $2 \leq l \leq n$, and we are done.

(2) There is an e_i which is minimal in $\{e_2, \ldots, e_n\}$ such that $e_1 \smile_{\mathbf{E}} e_i$. This means that $e'_1 \smile_{\mathbf{F}} e'_i$. We now have the following situation:

$$X \cup \{e_i\}[\{e_1\} \ldots \{e_{i-1}\}\{e_{i+1}\} \ldots \{e_n\} >_{\mathbf{E}} X',$$

$$Y \cup \{e'_i\}[\{e'_1\} \ldots \{e'_{i-1}\}\{e'_{i+1}\} \ldots \{e'_n\} >_{\mathbf{F}} Y'.$$

Of course $X \cup \{e_i\}$ and $Y \cup \{e'_i\}$ have the same step sequences. Application of the induction hypothesis gives

$$\mathbf{E} \upharpoonright \{e_1, e_2, \ldots, e_{i-1}, e_{i+1}, \ldots, e_n\} \cong \mathbf{F} \upharpoonright \{e'_1, e'_2, \ldots, e'_i, e'_{i-1}, \ldots, e'_n\}.$$

Consequently $e_1 <_{\mathbf{E}} e_l \Leftrightarrow e'_1 <_{\mathbf{F}} e'_l$ for $2 \leq l \leq n$. $\square$

Observe that in the proof of Theorem 5.1 we only use that $\mathbf{E}$ and $\mathbf{F}$ have the same sequences of steps containing at most two events.

The diagram below presents the relations between the equivalences presented thus far when restricted to the domain of deterministic event structures.

$$\begin{array}{ccccc} \cong & = & \equiv_{\text{pom}} & = & \equiv_{\text{step}} \\ \cap & & & & \\ \underline{\leftrightarrow} & = & \equiv_{\text{seq}} & & \end{array}$$

The example of Fig. 8 shows that even for deterministic systems there is a difference between arbitrary interleaving and partial order semantics.

6. Arbitrary interleaving versus True concurrency

One can consider event structures up to step sequence equivalence as an interleaving semantics if one is willing to view a multiset of actions as an action again.

In the process algebra languages MEIJE and ACP this idea can be implemented by working for instance with an action structure which is the product of a free commutative monoid and a free commutative group. Under this interpretation one can say that for deterministic systems there is no difference between arbitrary interleaving and True concurrency.

Now one can ask the question to what extent a multiset of more than one action can be considered as something which is observable. In a synchronous system like a systolic architecture there is certainly no problem. After each clock tick, one can just stop the system and examine which "cells" have performed an action. The multiset (or set if the system is deterministic) of actions performed by the separate cells gives the step which is performed by the synchronous system. It is much harder to imagine how a "step" can be observed in an asynchronous system. The only thing I can come up with is that some observer notices the beginning of one action before another action has been finished. In such a situation the observer can conclude that the two actions occur concurrently.

Below, this way of observing concurrent processes is formally implemented by means of an operator *split* on event structures that splits any event e into events e^+ and e^-, which are ordered. One may think of e^+ as the beginning of e and of e^- as the end of e.

Definition 6.1. Let $\mathbf{E}$ be an event structure over some alphabet A. Let $A^+ = \{a^+ \mid a \in A\}$ and $A^- = \{a^- \mid a \in A\}$ be two disjoint copies of A. The event structure $\mathbf{F} = \text{split}(\mathbf{E})$ over alphabet $A^+ \cup A^-$ is given by

$$E_{\mathbf{F}} = \{e^+, e^- \mid e \in E_{\mathbf{E}}\},$$

$$<_{\mathbf{F}} = \{(e^x, f^y) \mid x, y \in \{+, -\} \text{ and } e <_{\mathbf{E}} f\} \cup \{(e^+, e^-) \mid e \in E_{\mathbf{E}}\},$$

$$\#_{\mathbf{F}} = \{(e^x, f^y) \mid x, y \in \{+, -\} \text{ and } e \#_{\mathbf{E}} f\},$$

$$l_{\mathbf{F}}(e^+) = (l_{\mathbf{E}}(e))^+,$$

$$l_{\mathbf{F}}(e^-) = (l_{\mathbf{E}}(e))^-.$$

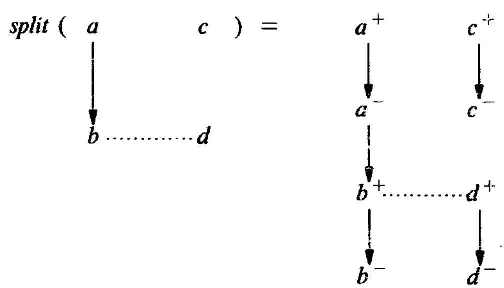

Fig. 11.

Example 6.2. See Fig. 11.

Definition 6.3. Two event structures **E** and **F** are *split sequence equivalent*, notation $\mathbf{E} \equiv_{\text{split}} \mathbf{F}$, if $\text{split}(\mathbf{E}) \equiv_{\text{seq}} \text{split}(\mathbf{F})$.

Split sequence equivalence is closely related to ST-bisimulation semantics as presented in [11] on the domain of Petri nets, but there are some differences. Besides the fact that split sequence equivalence does not respect branching time it is also not real time consistent in the sense of [11]. The idea of splitting actions into a beginning and an end is, on a different and more restricted domain, also described in [13]. Our split-operator can be viewed as a special case of *action refinement* as described in [7, 2].

Lemma 6.4. *Let* **E** *and* **F** *be two event structures. Then*

$$\mathbf{E} \equiv_{\text{pom}} \mathbf{F} \Rightarrow \text{split}(\mathbf{E}) \equiv_{\text{pom}} \text{split}(\mathbf{F}).$$

Proof. The main idea of the proof has already occurred in [7]. Let **E** and **F** be event structures with $\mathbf{E} \equiv_{\text{pom}} \mathbf{F}$. Choose a configuration $X \in \mathscr{C}(\text{split}(\mathbf{E}))$. We must show that there exists a configuration $Y \in \mathscr{C}(\text{split}(\mathbf{F}))$ such that

$$\text{split}(\mathbf{E}) \upharpoonright X \cong \text{split}(\mathbf{F}) \upharpoonright Y.$$

By symmetry it follows that we are ready if we have proved this. Define the sets $X^{\pm}, X^{+} \subseteq E_{\mathbf{E}}$ by

$$X^{\pm} = \{e \in E_{\mathbf{E}} \mid e^{+} \in X \text{ and } e^{-} \in X\},$$

$$X^{+} = \{e \in E_{\mathbf{E}} \mid e^{+} \in X \text{ and } e^{-} \notin X\}.$$

One can easily check that $X^{\pm} \cup X^{+}$ is a configuration of **E**. Since $\mathbf{E} \equiv_{\text{pom}} \mathbf{F}$, there is a configuration $Y \in \mathscr{C}(\mathbf{F})$ and a bijection $f: X^{\pm} \cup X^{+} \to Y$ which gives an event structure isomorphism between $\mathbf{E} \upharpoonright (X^{\pm} \cup X^{+})$ and $\mathbf{F} \upharpoonright Y$. Define $Y^{\text{split}} \subseteq E_{\text{split}(\mathbf{F})}$ by

$$Y^{\text{split}} = \{(f(e))^{+}, (f(e))^{-} \mid e \in X^{\pm}\} \cup \{(f(e))^{+} \mid e \in X^{+}\}.$$

It is not hard to see that Y^{split} is a configuration of split(**F**). Now define a mapping $f^{\text{split}}: X \to Y^{\text{split}}$ by

$$f^{\text{split}}(e^{+}) = (f(e))^{+} \quad \text{for } e^{+} \in X,$$

$$f^{\text{split}}(e^{-}) = (f(e))^{-} \quad \text{for } e^{-} \in X.$$

We claim that f^{split} is an event structure isomorphism between $\text{split}(\mathbf{E}) \upharpoonright X$ and $\text{split}(\mathbf{F}) \upharpoonright Y^{\text{split}}$. A simple argument gives that f^{split} is a bijection. Clearly f^{split} preserves labels. Finally we have that if two events in X are ordered their images under f^{split} are also ordered, and if two events in X are concurrent their images under f^{split} are concurrent too. $\square$

Proposition 6.5. *Let* **E** *and* **F** *be two event structures. Then*

$$\mathbf{E} \equiv_{\text{pom}} \mathbf{F} \Rightarrow \mathbf{E} \equiv_{\text{split}} \mathbf{F}.$$

Proof. $\mathbf{E} \equiv_{\text{pom}} \mathbf{F} \Rightarrow \text{split}(\mathbf{E}) \equiv_{\text{pom}} \text{split}(\mathbf{F}) \Rightarrow \text{split}(\mathbf{E}) \equiv_{\text{seq}} \text{split}(\mathbf{F}) \Rightarrow \mathbf{E} \equiv_{\text{split}} \mathbf{F}$. □

Proposition 6.6. *Let* **E** *and* **F** *be two event structures. Then*

$$\mathbf{E} \equiv_{\text{split}} \mathbf{F} \Rightarrow \mathbf{E} \equiv_{\text{step}} \mathbf{F}.$$

Proof. Let **E** and **F** be two event structures with $\mathbf{E} \equiv_{\text{split}} \mathbf{F}$. Let $\sigma = A_1 \ldots A_m \in (\text{Mul}(A))^*$ with $A_i = \{a_{i1}, \ldots, a_{in_i}\}$ an action step sequence of **E**. We must show that σ is also an action step sequence of **F**. By symmetry we are ready if we have proved this. The following sequence ρ is an action sequence of split(**E**):

$$\rho = a^+_{11} a^+_{12} \ldots a^+_{1n_1} a^-_{11} a^-_{12} \ldots a^-_{1n_1} a^+_{21} \ldots a^+_{m1} \ldots a^+_{mn_m} a^-_{m1} \ldots a^-_{mn_m}.$$

Since $\mathbf{E} \equiv_{\text{split}} \mathbf{F}$, ρ is also an action sequence of split(**F**). Hence split(**F**) has some event sequence α with the property that, if we replace the events in α by their labels, we obtain ρ. Let this α be

$$\alpha = e^+_{11} e^+_{12} \ldots e^+_{1n_1} f^-_{11} f^-_{12} \ldots f^-_{1n_1} e^+_{21} \ldots e^+_{m1} \ldots e^+_{mn_m} f^-_{m1} \ldots f^-_{mn_m}.$$

Note that in general e_{ij} may be different from f_{ij}. However, we do have that $\{e_{i1}, \ldots, e_{in_i}\}$ equals $\{f_{i1}, \ldots, f_{in_i}\}$. From the fact that α is an event sequence of split(**F**) it follows that **F** has the event step sequence

$$\{e_{11}, \ldots, e_{1n_1}\} \ldots \{e_{m1}, \ldots, e_{mn_m}\}.$$

Hence σ is an action step sequence of **F**. □

As a consequence of Propositions 6.5 and 6.6, split sequence equivalence can be located in our semantical lattice as follows:

$$\begin{array}{ccccccc} \cong & & & \subset & & & \underline{\leftrightarrow} \\ \cap & & & & & & \cap \\ \equiv_{\text{pom}} & \subset & \equiv_{\text{split}} & \subset & \equiv_{\text{step}} & \subset & \equiv_{\text{seq}} \end{array} \qquad (*)$$

Examples 6.7. The examples shown in Figs. 12 and 13 show that all equivalences in (*) above are different. Due to Theorem 5.1 and the position of $\equiv_{\text{split}}$ in the semantical lattice we have that for deterministic event structures, split bisimulation equivalence and event structure isomorphism coincide.

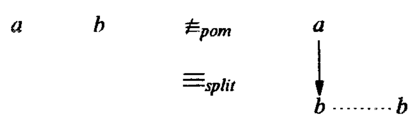

Fig. 12.

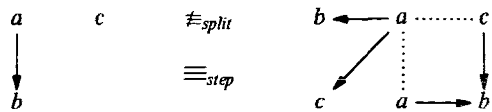

Fig. 13.

Proposition 6.8. *Let* $\mathbf{E}, \mathbf{F}$ *be deterministic event structures. Then* $\mathbf{E} \cong \mathbf{F} \Leftrightarrow \mathbf{E} \equiv_{\text{split}} \mathbf{F}$.

Thus the causal structure of a deterministic concurrent system can be unravelled by observers who are capable of observing the beginning and termination of events.

Acknowledgment

The author would like to thank Rob van Glabbeek for many stimulating discussions and careful proofreading, Henk Goeman for some useful comments on an earlier version, and Alex Rabinovich for pointing out that the assumption of binary conflict is not essential for the results of this paper.

References

[1] L. Aceto, R. De Nicola and A. Fantechi, Testing equivalences for event structures, in: M. Venturini Zilli, ed., *Proc. Advanced School on Mathematical Models for the Semantics of Parallelism*, 1986, Lecture Notes in Computer Science **280** (Springer, Berlin, 1987) 1–20.

[2] L. Aceto and M. Hennessy, Towards action-refinement in process algebras, in: *Proc. 4th Ann. Symp. on Logic in Computer Science (LICS)*, Asilomar, CA (IEEE Computer Society Press, Washington, 1989) 138–145.

[3] D. Austry and G. Boudol, Algèbre de processus et synchronisations, *Theoret. Comput. Sci.* **30** (1984) 91–131.

[4] J.A. Bergstra and J.W. Klop, Algebra of communicating processes with abstraction, *Theoret. Comput. Sci.* 37 (1985) 77–121.

[5] E. Best and R. Devillers, Sequential and concurrent behavior in Petri net theory, *Theoret. Comput. Sci.* **55** (1987) 87–136.

[6] G. Boudol and I. Castellani, Concurrency and atomicity, *Theoret. Comput. Sci.* **59** (1988) 25–84.

[7] L. Castellani, G. De Michelis and L. Pomello, Concurrency vs interleaving: an instructive example, *Bull. EATCS* **31** (1987) 12–15.

[8] P. Degano, R. De Nicola and U. Montanari, Observational equivalences for concurrency models, in: M. Wirsing, ed., *Formal Description of Programming concepts—III, Proc. 3rd IFIP WG 2.2 Working Conf.*, Ebberup 1986 (North-Holland, Amsterdam, 1987) 105–129.

[9] J. Engelfriet, Determinacy → (observation equivalence = trace equivalence), *Theoret. Comput. Sci.* **36** (1985) 21–25.

[10] H.J. Genrich and E. Stankiewicz-Wiechno, A dictionary of some basic notions of Petri nets, in: W. Brauer, ed., *Advanced Course on General Net Theory of Processes and Systems*, Hamburg, 1979, Lecture Notes in Computer Science **84** (Springer, Berlin, 1980).

[11] R.J. van Glabbeek and F.W. Vaandrager, Petri net models for algebraic theories of concurrency, in: J.W. de Bakker, A.J. Nijman and P.C. Treleaven, eds., *Proc. PARLE Conf., Eindhoven, Vol. II (Parallel Languages)* Lecture Notes in Computer Science **259** (Springer, Berlin, 1987) 224–242.

[12] J. Grabowski, On partial languages, *Fund. Inform.* IV(2) (1981) 427–498.

[13] M. Hennessy Axiomatising finite concurrent processes, *SIAM J. Comput.* **17**(5) (1988) 997–1017.
[14] C.A.R. Hoare, *Communicating Sequential Processes* (Prentice Hall, Englewood Cliffs, NJ, 1985).
[15] A. Kiehn, On the interrelation between synchronized and non-synchronized behaviour of Petri nets, *J. Inform. Process. Cybernet. EIK* **24**(1/2) (1988) 3–18.
[16] A. Mazurkiewicz, Trace theory, in: W. Brauer, W. Reisig and G. Rozenberg, eds., *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proc. Advanced Course*, Bad Honnef, September 1986, Lecture Notes in Computer Science **255** (Springer, Berlin, 1987) 279–324.
[17] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science **92** (Springer, Berlin, 1980).
[18] M. Nielsen, G.D. Plotkin and G. Winskel, Petri nets, event structures and domains, part I. *Theoret. Comput. Sci.* **13** (1981) 85–108.
[19] M. Nielsen and P.S. Thiagarajan, Degrees of non-determinism and concurrency: a Petri net view, in: M. Joseph and R. Shyamasundar, eds., *Proc. 5th Conf. on Found. of Softw. Techn. and Theor. Comp. Sci.*, Lecture Notes in Computer Science **181** (Springer, Berlin, 1984) 89–118.
[20] Penguin, *The New Penguin English Dictionary* (Penguin Books, 1986).
[21] L. Pomello, Some equivalence notions for concurrent systems. An overview, in: G. Rozenberg, ed., *Advances in Petri Nets 1985*, Lecture Notes in Computer Science **222** (Springer, Berlin, 1986) 381–400.
[22] V.R. Pratt, Modelling concurrency with partial orders, *Internat. J. Parallel Programming* **15**(1) (1986) 33–71.
[23] M. Rem, Trace theory and systolic computations, in: J.W. de Bakker, A.J. Nijman and P.C. Treleaven, eds., *Proc. PARLE Conf. Eindhoven, Vol. I (Parallel Architectures)* Lecture Notes in Computer Science **258** (Springer, Berlin, 1987) 14–33.
[24] G. Rozenberg and P.S. Thiagarajan, Petri nets: basic notions, structure, behaviour, in: J.W. de Bakker, W.-P. de Roever and G. Rozenberg, eds., *Current Trends in Concurrency*, Lecture Notes in Computer Science **224** (Springer, Berlin, 1986) 585–668.
[25] M.W. Shields, Non sequential behaviour: 1, Internal Report CSR-120-82, Department of Computer Science, University of Edinburgh, 1982.
[26] D.A. Taubner and W. Vogler, The step failure semantics, in: F.J. Brandenburg, G. Vidal-Naquet and M. Wirsing, eds., *Proc. STACS 87*, Lecture Notes in Computer Science **247** (Springer, Berlin, 1987) 348–359.
[27] B.A. Trakhtenbrot, A. Rabinovich and J. Hirshfeld, Discerning causality in the behaviour of automata, Technical Report 104/88, Tel Aviv University, 1988.
[28] G. Winskel, Event structures, in: W. Brauer, W. Reisig and G. Rozenberg, eds., *Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II; Proc. of an Advanced Course*, Bad Honnef, September 1986, Lecture Notes in Computer Science **255** (Springer, Berlin, 1987) 325–392.