Interleaving Semantics and Action Refinement with Atomic Choice

Ingo Czaja
Gesellschaft für Mathematik und Datenverarbeitung
Postfach 1316, D-5205 Sankt Augustin, Germany

Rob J. van Glabbeek
Computer Science Department, Stanford University
Stanford, CA 94305, USA
`rvg@cs.stanford.edu`

Ursula Goltz
Gesellschaft für Mathematik und Datenverarbeitung
Postfach 1316, D-5205 Sankt Augustin, Germany
`goltz@gmdzi.gmd.de`

We investigate how to restrict the concept of *action refinement* such that established interleaving equivalences for concurrent systems are preserved under these restricted refinements. We show that *interleaving bisimulation* is preserved under refinement if we do not allow to refine action occurrences deciding choices and action occurrences involved in autoconcurrency. On the other hand, *interleaving trace equivalence* is still not preserved under these restricted refinements.

Keywords Concurrency, action refinement, atomicity, flow event structures, semantic equivalences, interleaving vs. partial orders, bisimulation.

Notes This paper appeared as Arbeitspapiere der GMD 594, Sankt Augustin 1991. The work presented here has partly been carried out within the Esprit Basic Research Action 3148 (DEMON) and the Sonderforschungsbereich 182 of the University of Erlangen-Nürnberg.

1 Introduction

For the design of concurrent systems, it can be useful to consider a hierarchy of representations, which allows refinement of unstructured entities on a more abstract design level by complex structures on a lower level. One approach which is being investigated is *action refinement* where the entities being refined are the actions which a system may perform. A single action in an abstract view may correspond to a process consisting of many actions in a more concrete representation. Action refinement is being investigated in process algebras [Aceto, AH, NEL] and in semantic models of concurrent systems [GG a/b/c, Vogler a/b, BDKP, Devillers, DD a/b, BGV].

Action refinement can be modelled as an operator, taking a system description P on a given level of abstraction as well as a refinement function ref associating processes with actions, and yielding a system description $ref(P)$ on a lower level, obtained by replacing action occurrences by occurrences of the associated processes. It differs from many other operators that are considered for concurrent systems in that applying it on a representation of a system does not yield a representation of a different system, but a different representation of the same system.

An important question being considered is the following: Which equivalence notions for concurrent systems are *preserved under refinement*? Taking two system representations, P, Q, which are equivalent for some equivalence notion $\sim$, we would expect that $ref(P) \sim ref(Q)$ for any refinement function ref. This expectation indicates a big difference between action refinement and many other approaches for moving to a lower level of abstraction, in which adding more information about the precise implementation of systems makes it more likely that systems turn out to be different that were indistinguishable on the higher level of abstraction. Our expectation is based on the idea that in the original system representations P and Q it is not known how the actions will be implemented. However, it is decided that different occurrences of the same action will be implemented in the same way, so that any difference between P and Q that could arise after refinement is already visible in the abstract representations, namely through the use of different action names for corresponding events.

The preservation question has been addressed in the papers mentioned above. The usual approach to this problem is: Take some well-established equivalence notion. If it is not preserved under refinement, try to find the coarsest equivalence notion contained in it which has this property (see e.g. [Vogler a/b]).

What has not been considered so far is the following question. Given a well-established equivalence notion which is not preserved under refinement, is there a way of restricting either the allowed refinements or the class of system representations under consideration such that preservation of this equivalence in the restricted setting is obtained? This question is addressed in the present paper for two of the most basic equivalences.

We consider system descriptions in which the basic building blocks are the actions which may occur in a system. In our approach an action can be any activity which is considered as a conceptual entity on a chosen level of abstraction. Now we distinguish *atomic* actions that cannot be refined, and compound ones that can. One can think of atomic actions as being instantaneous and of compound actions as durational. Many approaches to concurrency in which actions are not supposed to be refined, in particular the *interleaving* approaches, in which parallelism is equated with arbitrary interleaving of action occurrences, can be thought of as dealing exclusively with the subclass of system representations were all actions in a system are atomic. On the other hand the action refinement approaches cited above take the possibility into account that *any* action can be refined. In this paper we look at systems in which some actions are atomic, and others can be refined. In particular we consider systems in which all action occurrences deciding choices, as well as all actions that can occur concurrently with themselves are atomic. These restrictions turn out to be sufficient to apply action refinement in certain interleaving semantics, while preserving semantic equivalence. Although the distinction between atomic and compound actions may be useful for motivating this research, technically one may just as well drop this distinction and restrict the class of allowed refinement functions to a class of what we call *safe* refinements. Those refinements do not refine action occurrences deciding choices and actions involved in autoconcurrency.

It turns out that the *branching structure* (i.e. the relative order of action occurrences and choices between different courses of action) of concurrent systems plays a rôle in our results. We show that *interleaving bisimulation*, where the branching structure between alternative executions is taken into account, is preserved under safe refinements. On the other hand, we show that *interleaving trace equivalence* (where the branching structure is fully neglected) is still not preserved under these restricted refinements. To investigate safe refinements for equivalence notions between these two extremes in the linear time – branching time spectrum has to be left for future research.

2 Basic Notions

We consider systems which are capable of performing actions from a given set *Act* of action names. By an action we understand here any activity which is considered as an conceptual entity on a chosen level of abstraction. We will not distinguish external and internal actions; we do not consider abstraction by hiding of actions in this paper.

We will sometimes give process algebra terms for examples to make them easier to understand: $+$ will denote choice (as in CCS), $|$ will denote parallel composition (without communication), $a, b, ... \in Act$ denote actions, and $;$ denotes a general sequential composition operator (like the one of ACP). The semicolon will sometimes be omitted and we use the usual precedence rule that $;$ binds stronger than the other operators. However this notation is only used for intuition; formally our results are established for *event structures*.

We will describe a concurrent system by a set of events where each event corresponds to an occurrence of some action. Therefore, events are labelled by action names.

The simplest form of event structures has been introduced in [NPW] and is usually referred to as *prime event structures* with binary conflicts. A labelled prime event structure is a tuple $(E, <, \#, l)$ where E is the set of events, $< \subseteq E \times E$ is an (irreflexive) partial order (the *causality relation*), $\# \subseteq E \times E$ is the so-called *conflict relation*, and $l : E \rightarrow Act$ specifies the labelling of events by action names. As usual, we require $<$ to satisfy the *axiom of finite causes*; that is, any event may have only finitely many predecessors. Furthermore, $\#$ is required to be irreflexive and symmetric and to respect the axiom of conflict heredity: $\forall d, e, f \in E : d \# e \wedge d < f \Longrightarrow e \# f$.

A prime 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, $d < e$ means that d is a prerequisite for e and $d \# e$ means that d and e cannot happen both in the same run.

The behaviour of a prime event structure is described by explaining which subsets of events constitute possible runs of the represented system. These subsets are called *configurations*. They have to be conflict-free and they must be left-closed with respect to $<$ (all prerequisites for any event occurring in the run must also occur).

As shown in [GG a], prime event structures are well suited to define a notion of action refinement, as long as actions are not refined by behaviours containing conflicts.

But as a consequence of the axioms explained above, it is cumbersome to define a more general notion of refinement directly for prime event structures. Whenever a given event occurs, it has to be "enabled" by a unique set of events that is independent of the particular run of the system, namely by all its predecessors according to the causality relation. However, when refining a by $a_1 + a_2$ in $a\,; b$, b should occur alternatively caused by a_1 or by a_2. In prime event structures, this may only be modelled by duplicating the b-event, leading to complicated definitions and proofs. Hence more expressive event structure representations are being used for refinement. *Free event structures* as defined in [DD a] seem to be weakest extension of prime event structures to allow refinements with conflicts in a straightforward way. However, difficulties with the interpretation of the notion of causality in this model lead in [DD b] to a restriction such that prime event structures are no longer a subclass. In [GG a, GG b], *flow event structures* [BC] have been used which are also a convenient model for CCS. This is the model we have chosen here as well, in order to establish our results also for refinements with conflicts.

We will now introduce the main concepts of flow event structures following closely [Boudol].

Definition 2.1

A *(labelled) flow event structure (over an alphabet Act)* is a 4-tuple $\mathcal{E} = (E, \prec, \#, l)$, where

- E is a set of *events*,
- $\prec \subseteq E \times E$ is an irreflexive relation, the *flow relation*,
- $\# \subseteq E \times E$ is a symmetric relation, the *conflict relation*,
- $l : E \longrightarrow Act$ is the *labelling function*.

Let $\mathbb{E}$ denote the domain of flow event structures labelled over Act. The components of $\mathcal{E} \in \mathbb{E}$ will be denoted by $E_{\mathcal{E}}, \prec_{\mathcal{E}}, \#_{\mathcal{E}}$ and $l_{\mathcal{E}}$. The index $\mathcal{E}$ will be omitted if clear from the context.

Two flow event structures $\mathcal{E}$ and $\mathcal{F}$ are *isomorphic* ($\mathcal{E} \cong \mathcal{F}$) iff there exists a bijection between their sets of events preserving $\prec, \#$ and labelling. Often, we will not distinguish isomorphic event structures; the names of events are not important for us.

The interpretation of the conflict and the flow relation is formalised by defining configurations of flow event structures. Configurations must be conflict free; in particular, self-conflicting events will never occur in any configuration. $d \prec e$ will mean that d is a *possible immediate cause* for e. For an event to occur it is necessary that a *complete* non-conflicting set of its causes has occurred. Here a set of causes is complete if for any cause which is not contained there is a conflicting event which is contained. Finally, no cycles with respect to causal dependence may occur.

We will only consider finite configurations here. As usual, we assume that in a finite period of time only finitely many actions may be performed. Now the requirement says that we only consider runs which are executable in a finite period of time. This is no restriction since the infinite configurations which are usually considered are completely determined by the finite ones (see e.g. [Boudol]).

Definition 2.2 Let $\mathcal{E} \in \mathbb{E}$.

(i) $X \subseteq E$ is *left-closed up to conflicts* iff $\forall d, e \in E$: if $e \in X, d \prec e$ and $d \notin X$ then there exists an $f \in X$ with $f \prec e$ and $d \# f$.
$X \subseteq E$ is *conflict-free* iff $\#_{\mathcal{E}} \lceil X = \emptyset$.

(ii) $X \subseteq E$ is a *(finite) configuration* of $\mathcal{E}$ iff X is finite, left-closed up to conflicts and conflict-free and does not contain a causality cycle: $<_X := (\prec \cap (X \times X))^+$ is irreflexive. $Conf(\mathcal{E})$ denotes the set of all (finite) configurations of $\mathcal{E}$.

(iii) A configuration $X \in Conf(\mathcal{E})$ is called *maximal* iff $X \subseteq Y \in Conf(\mathcal{E})$ implies $X = Y$. It is *complete* iff $\forall d \in E : d \notin X \Rightarrow \exists e \in X$ with $d \# e$.

The relation $<_X$ expresses causal dependence in the configuration X. Note that prime event structures are special flow event structures defining $d \prec e$ iff $d < e$. In prime event structures, the requirement of left-closedness up to conflicts for configurations reduces to left-closedness: $\forall d, e \in E$: if $e \in X \wedge d < e$ then $d \in X$; furthermore, in each configuration X, the relation $<_X$ coincides with $<$. However in prime event structures every maximal configuration is complete, whereas in flow event structures this need not be the case (although every complete configuration must be maximal). In [GG c] the difference between complete and maximal configurations is used to model deadlock behaviour: complete configurations indicate successful termination, whereas incomplete maximal configurations model deadlocks.

The following observations may offer some additional motivation for the definition of configurations above.

Definition 2.3 Let $\mathcal{E} \in \mathbb{E}$, $X \in \mathit{Conf}(\mathcal{E})$.

(i) X *enables* $e \in E$ iff $e \notin X$ and $X \cup \{e\} \in \mathit{Conf}(\mathcal{E})$.

(ii) A subset $Y \subseteq X$ is a *prefix* of X iff $\forall d \in X,\ e \in Y:\ d <_X e \Rightarrow d \in Y$ (i.e. Y is left-closed in X w.r.t. $<_X$).

Proposition 2.1 Let $\mathcal{E} \in \mathbb{E}$, $X \in \mathit{Conf}(\mathcal{E})$.

(i) X enables e iff $e \notin X, \neg(\exists d \in X$ with $d \# e)$ and $\forall d \prec e$ we have $d \in X \vee \exists f \in X$ with $f \prec e$ and $d \# f$.

(ii) X can be written as $X = \{e_1, \ldots, e_n\}$ such that for $k < n : \{e_1, \ldots, e_k\}$ enables e_{k+1}.

(iii) For $Y \subseteq X$, $Y \in \mathit{Conf}(\mathcal{E})$ iff Y is a prefix of X.

Proof See [Boudol]. $\square$

We now define transition relations between configurations, describing which event or which action may occur in a configuration and which configuration may then be obtained.

Definition 2.4 Let $\mathcal{E} \in \mathbb{E}$.

$X \xrightarrow{e}_{\mathcal{E}} X'$ iff X enables e and $X' = X \cup \{e\}$.
$X \xrightarrow{a}_{\mathcal{E}} X'$ iff $X \xrightarrow{e}_{\mathcal{E}} X'$ and $l(e) = a \in \mathit{Act}$.

The index $\mathcal{E}$ is omitted if clear from the context.

Next we introduce a relation co_X expressing when two events are *concurrent* (independent) within a configuration X, as well as a global relation *co*, expressing *possible concurrency*, and show that the relations correspond. Furthermore we define when there is a *choice* between two events.

Definition 2.5 Let $\mathcal{E} \in \mathbb{E}$.

(i) For $X \in \mathit{Conf}(\mathcal{E})$, let $co_X \subseteq X \times X$ be defined by
$d \; co_X \; e$ iff $\neg(d <_X e \vee e <_X d)$.

(ii) $co \subseteq E \times E$ is defined by
$d \; co \; e$ iff $\exists X \in \mathit{Conf}(\mathcal{E})$, X enables both d and e and $X \cup \{d, e\} \in \mathit{Conf}(\mathcal{E})$.

(iii) $ch \subseteq E \times E$ is defined by
$d \; ch \; e$ iff $\exists X \in \mathit{Conf}(\mathcal{E})$, X enables both d and e and $X \cup \{d, e\} \notin \mathit{Conf}(\mathcal{E})$.

Proposition 2.2 Let $\mathcal{E} \in \mathbb{E}$, $d, e \in E$.

(i) $d \; co \; e \;\Leftrightarrow\; \exists X \in \mathit{Conf}(\mathcal{E})$ with $d \; co_X \; e$.

(ii) $d \; ch \; e \;\Rightarrow\; d \# e$.

Proof

(i):

"$\Rightarrow$": Let $d \; co \; e$. Then there exists $X \in \mathit{Conf}(\mathcal{E})$ with d, e both enabled by X.
Let $Y := X \cup \{d, e\}$. We show that $d \; co_Y \; e$.
Assume $d <_Y e$. Then $X \cup \{e\}$ would not be a prefix of Y, hence $X \cup \{e\}$ would not be a configuration and e not enabled by X. Thus $\neg(d <_Y e)$ and by symmetry $\neg(e <_Y d)$.

"$\Leftarrow$": Let $X \in \mathit{Conf}(\mathcal{E})$ with $d \; co_X \; e$. Let $X' = X - \{f \in X \mid d <_X f \vee e <_X f\}$.
$d, e \in X'$, since $d, e \in X$ and d and e are not in $<_X$-relation.
Moreover d and e are maximal in X' w.r.t. $<_X$.
Let $X_1 := X' - \{d\}$, $X_2 := X' - \{e\}$, $X'' := X' - \{d, e\}$.
Then $X', X_1, X_2, X'' \in \mathit{Conf}(\mathcal{E})$ with proposition 2.1(iii) and $d \; co \; e$.

(ii):

Suppose $d \; ch \; e$, but $\neg(d \# e)$. We show that for each configuration X enabling d and e, also $X' := X \cup \{d, e\}$ is a configuration.
Let $X \in \mathit{Conf}(\mathcal{E})$, with $X \xrightarrow{d} X_1$ and $X \xrightarrow{e} X_2$. Let $X' := X \cup \{d, e\}$.

X' is finite, since $X \in \mathit{Conf}(\mathcal{E})$.

X' is conflict-free, since $X_1, X_2 \in \mathit{Conf}(\mathcal{E})$ and $\neg(d \# e)$.

X' is left-closed up to conflicts:
Suppose $f \in X'$, $g \in E - X'$ and $g \prec f$. Then $f \in X_i$ for $i = 1$ or $i = 2$ and $g \notin X_i$, so $\exists h \in X_i \subseteq X'$ with $h \prec f$ and $g \# h$, since $X_i \in \mathit{Conf}(\mathcal{E})$.

X' contains no cycles w.r.t. $\prec$:
Suppose X' contains a causality cycle. Since $X_2 \in \mathit{Conf}(\mathcal{E})$ this cycle must contain d. However, we will show that there is no $c \in X'$ with $d \prec c$. Since $X, X_1 \in \mathit{Conf}(\mathcal{E})$, X must be a prefix of X_1 by proposition 2.1(iii), implying that d is maximal in X_1. Thus the only candidate for c is e. So suppose $d \prec e$. Since $d \notin X_2$ and $e \in X_2 \in \mathit{Conf}(\mathcal{E})$ there must be an $f \in X_2 \subseteq X'$ with $f \prec e$ and $d \# f$, contradicting the conflict-freeness of X'.

Thus $X' \in \mathit{Conf}(\mathcal{E})$. $\square$

For prime event structures, the choice relation coincides with the non-inherited conflicts.

Definition 2.6 Let $\mathcal{E} \in \mathbb{E}$ be a prime event structure.

> $\#^1 \subseteq E \times E$ (*immediate conflict*) is defined by
> $d \#^1 e$ iff $d \# e$, $f < d \Rightarrow \neg(f \# e)$ and $f < e \Rightarrow \neg(f \# d)$.

Proposition 2.3 Let $\mathcal{E} \in \mathbb{E}$ be a prime event structure, $d, e \in E$.

> Then $d \; ch \; e \;\Leftrightarrow\; d \#^1 e$.

Proof
"$\Rightarrow$": Let $d \; ch \; e$. Then there exists $X \in \mathit{Conf}(\mathcal{E})$ enabling both d and e.
$d \# e$ by proposition 2.2.
Assume there is an f with $f < d$. Then $f \in X$, since $X \cup \{d\}$ is left-closed. Since $f \in X \cup \{e\} \in \mathit{Conf}(\mathcal{E})$, we have $\neg(f \# e)$.
Similarly $f < e \Rightarrow \neg(f \# d)$.

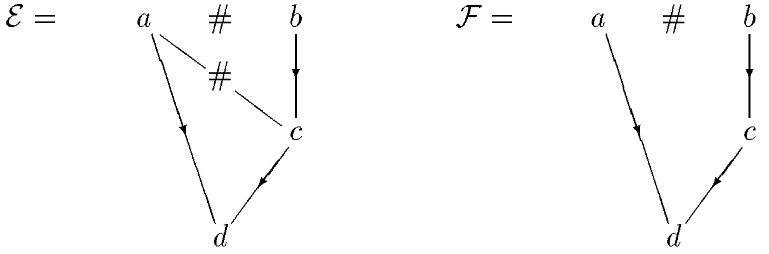
"$\Leftarrow$": Let $d \#^1 e$. Let $X := \{f \in E \mid f < d \vee f < e\}$.
Since $d \# e$ it cannot be the case that $d < e$ or $e < d$ by the axiom of conflict heredity and the irreflexivity of $\#$. Thus $d, e \notin X$.
Let $X_1 = X \cup \{d\}$ and $X_2 := X \cup \{e\}$.
Since $d \# e$ we have $X \cup \{d, e\} \notin \mathit{Conf}(\mathcal{E})$, so it suffices to prove that X, X_1 and X_2 are configurations.

X, X_1 and X_2 are finite (axiom of finite causes).
X, X_1 and X_2 are left-closed (transitivity of $<$).
X, X_1 and X_2 are conflict-free:
It suffices to prove that X_1 is conflict-free, since this implies that X is conflict-free, and the conflict-freeness of X_2 follows by symmetry.
Suppose $d', e' \in X_1$ and $d' \# e'$. We may assume $d' \leq d$ and $e' < e$, since the cases that d' and e' are both predecessors of d or of e are excluded by the axiom of conflict heredity and the irreflexivity of $\#$. But then $d \# e'$ by conflict heredity, violating the definition of $\#^1$. $\square$

Note that for prime event structures the conflict relation is fully determined by the immediate conflicts, and hence by the choice relation. On the other hand, in flow event structures the choice relation does not determine the conflict relation. Moreover, the part of the conflict relation that is not determined by the choice relation influences the set of configurations (as well as the deadlock behaviour [GG c]).

Example 2.1

Consider the following two flow event structures, where the labelling is the identity. Here and later we represent the flow relation by arrows of the form $\longrightarrow$.



In both event structures the events a and c are in *semantic conflict*, in the sense that they cannot occur in the same configuration. However, in $\mathcal{F}$ they are not in *syntactic conflict*: $\neg(a \#_{\mathcal{F}} c)$.
We have $E_{\mathcal{E}} = E_{\mathcal{F}}$, $\prec_{\mathcal{E}} = \prec_{\mathcal{F}}$ and $l_{\mathcal{E}} = l_{\mathcal{F}}$. Moreover the choice relations of $\mathcal{E}$ and $\mathcal{F}$ agree: only $a\ ch\ b$. Nevertheless $\mathcal{E}$ has a configuration $\{a, d\}$ which $\mathcal{F}$ has not.

Refinement of actions in flow event structures may now be defined as follows. We assume a refinement function $ref : Act \longrightarrow \mathbb{E} - \{O\}$ (where O denotes the empty flow event structure) and replace each event labelled by a by a disjoint copy of $ref(a)$. The conflict and causality structure will be inherited. Refinements in which some actions are replaced by the empty process 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 [GG a]. Therefore they are not considered here.

Definition 2.7

(i) A function $ref : Act \longrightarrow \mathbb{E} - \{O\}$ is called a *refinement function* .

(ii) Let $\mathcal{E} \in \mathbb{E}$ and let ref be a refinement function.
Then the *refinement of $\mathcal{E}$ by ref*, $ref(\mathcal{E})$, is the flow event structure defined by

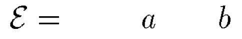
- $E_{ref(\mathcal{E})} = \{(e, e') \mid e \in E_{\mathcal{E}}, e' \in E_{ref(l_{\mathcal{E}}(e))}\}$,
- $(d, d') \prec_{ref(\mathcal{E})} (e, e')$ iff $d \prec_{\mathcal{E}} e$ or $(d = e \wedge d' \prec_{ref(l_{\mathcal{E}}(d))} e')$,
- $(d, d') \#_{ref(\mathcal{E})} (e, e')$ iff $d \#_{\mathcal{E}} e$ or $(d = e \wedge d' \#_{ref(l_{\mathcal{E}}(d))} e')$,
- $l_{ref(\mathcal{E})}(e, e') = l_{ref(l_{\mathcal{E}}(e))}(e')$.

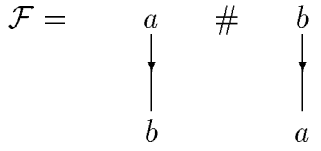
The following examples illustrate that the usual interleaving equivalences are not preserved under this refinement operation.

The event structures we consider in examples from now on will always be prime; hence we will only draw immediate conflicts and omit arcs for the elements of the flow relation obtainable by transitivity.

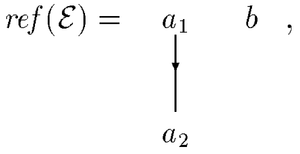
Example 2.2

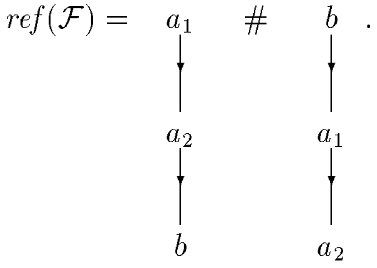
Consider the event structures $\mathcal{E}$ and $\mathcal{F}$ below, corresponding to $a|b$ and $ab + ba$.





$\mathcal{E}$ and $\mathcal{F}$ are indistinguishable in any kind of interleaving semantics. We refine action a by a_1a_2, i.e. $\mathit{ref}(a) = a_1 \longrightarrow a_2$, and then have

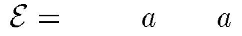


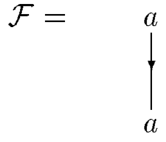


$\mathit{ref}(\mathcal{E})$ and $\mathit{ref}(\mathcal{F})$ are not interleaving equivalent, since $a_1b\,a_2$ is a possible execution of $\mathit{ref}(\mathcal{E})$ but not of $\mathit{ref}(\mathcal{F})$.

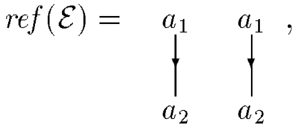
Example 2.3

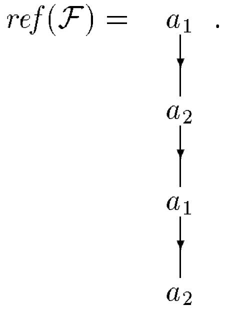
Consider the event structures $\mathcal{E}$ and $\mathcal{F}$ below, corresponding to $a|a$ and aa.





$\mathcal{E}$ and $\mathcal{F}$ are indistinguishable in common interleaving semantics. We refine a by a_1a_2 and then have





Now $\mathit{ref}(\mathcal{E})$ and $\mathit{ref}(\mathcal{F})$ are not interleaving equivalent, since only in $\mathit{ref}(\mathcal{E})$ it is possible to start with two occurrences of a_1.

We observe that in example 2.2 an event of $\mathcal{F}$ is refined which decides a choice. In example 2.3, an action is refined which may occur concurrently with itself. Hence we have identified two classes of events which will definitely cause problems for interleaving semantics if they are refined. We will now call actions labelling such events *critical* (with respect to a particular event structure $\mathcal{E}$) and consider the subclass of flow event structures where all critical actions are atomic, in the sense that they cannot really be refined. We could restrict the allowed refinement functions such that atomic actions may only be 'refined' by themselves. However, our refinement theorem will hold even in a slightly more general setting. First of all we may allow *renaming* of atomic actions, obtained by refining them by conflict-free event structures containing exactly one event, as well as *restriction* of certain (atomic) actions, obtained by refining them by event structures consisting of one self-conflicting event. In this way the relabelling and restriction operators of CCS can still be understood as instances of the refinement operator. Moreover we allow *generalised renamings*, replacing (atomic) actions by (possibly infinite) choices $a_1 + a_2 + \ldots$. The crucial restriction is that the execution of any event of a refined atomic action a leads to its immediate termination, thereby preserving atomicity for these actions. Formally this is ensured if and only if every non-empty configuration of $\mathit{ref}(a)$ is complete. This can be obtained by requiring that all events of $\mathit{ref}(a)$ are mutually in conflict.

Definition 2.8 Let $\mathcal{E} \in \mathbb{E}$.

(i) An action $a \in \mathit{Act}$ is *critical* in $\mathcal{E}$ iff
$\exists d, e \in E$ with $d \; \mathit{ch} \; e$ and $l(d) = a$ or
$\exists d, e \in E$, $d \neq e$ with $d \; \mathit{co} \; e$ and $l(d) = l(e) = a$.
$\mathit{Crit}_{\mathcal{E}} := \{a \in \mathit{Act} \mid a \text{ is critical in } \mathcal{E}\}$ is the set of all critical actions in $\mathcal{E}$.

(ii) A refinement function ref is called *safe* w.r.t. $\mathcal{E}$ iff
$\forall a \in \mathit{Crit}_{\mathcal{E}} \; \forall d, e \in E_{\mathit{ref}(a)}, \; d \neq e : \; d \#_{\mathit{ref}(a)} e$.

Proposition 2.4 Let $\mathcal{E} \in \mathbb{E}$, $a \in \mathit{Crit}_{\mathcal{E}}$, and ref a safe refinement w.r.t. $\mathcal{E}$.

Then every non-empty configuration of $\mathit{ref}(a)$ is complete.

Proof Trivial. □

In flow event structures it may be difficult to find out which actions are critical and should therefore be atomic. A safe strategy is to avoid autoconcurrency altogether and make every event which is in conflict with another event atomic. The result of this strategy depends on the particular flow event structure representation of a system by the choice of a syntactic conflict relation $\#$ (recall that semantic and syntactic conflict do not coincide for flow event structures). The most liberal possibilities for refinement are obtained by choosing a minimal conflict relation to model the intended behaviour.

3 Interleaving trace semantics

We now investigate whether interleaving equivalences are preserved by safe refinements. We start by considering the simplest notion, the usual *interleaving trace equivalence* where the possible sequences of actions are compared.

Definition 3.1

$w = a_1 \dots a_n \in Act^*$ is a *(sequential) trace* of $\mathcal{E} \in \mathbb{E}$ iff
$\exists X_0, \dots, X_n \in Conf(\mathcal{E}) : X_0 = \emptyset$ and $X_{i-1} \xrightarrow{a_i} X_i$, $i = 1, \dots, n$.

$SeqTraces(\mathcal{E})$ denotes the set of all sequential traces of $\mathcal{E}$.

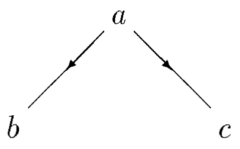
$\mathcal{E}, \mathcal{F} \in \mathbb{E}$ are called *interleaving trace equivalent* ($\mathcal{E} \approx_{it} \mathcal{F}$) iff
$SeqTraces(\mathcal{E}) = SeqTraces(\mathcal{F})$.

The following example shows that this equivalence is not preserved by safe refinements.

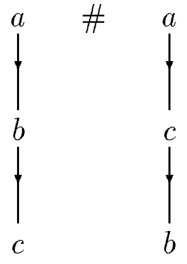
Example 3.1

Consider the event structures $\mathcal{E}$ and $\mathcal{F}$ below, corresponding to $a(b|c)$ and $abc + acb$.

$\mathcal{E} =$

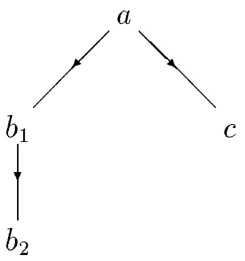


$\mathcal{F} =$

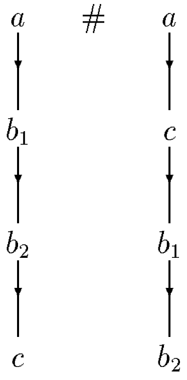


$\mathcal{E} \approx_{it} \mathcal{F}$, since the only traces for both event structures are abc and acb (and the prefixes). Action b is not critical. We refine b by $b_1 b_2$.

$ref(\mathcal{E}) =$



$ref(\mathcal{F}) =$



We have $ref(\mathcal{E}) \not\approx_{it} ref(\mathcal{F})$, since $ab_1c \in SeqTraces(ref(\mathcal{E}))$,
but $ab_1c \notin SeqTraces(ref(\mathcal{F}))$.

One can now ask if it is possible to strengthen the requirement of safe refinements, say into *trace-safe* refinements, by requiring more actions to be atomic, in such a way that interleaving trace equivalence is preserved under trace-safe refinements. Requiring all actions that can occur concurrently with another action to be atomic would work, but this constraint is very restrictive. It seems not easy to find a less restrictive one.

4 Interleaving bisimulation semantics

In this section we will show that interleaving bisimulation equivalence [Park, Milner] is preserved under safe refinements.

Definition 4.1 Let $\mathcal{E}, \mathcal{F} \in \mathbb{E}$.

A relation $R \subseteq \mathit{Conf}(\mathcal{E}) \times \mathit{Conf}(\mathcal{F})$ is called an *interleaving bisimulation between* $\mathcal{E}$ *and* $\mathcal{F}$ iff $(\emptyset, \emptyset) \in R$, and if $(X, Y) \in R$ then

- $X \xrightarrow{a} X' \Rightarrow \exists Y'$ with $Y \xrightarrow{a} Y'$ and $(X', Y') \in R$,
- $Y \xrightarrow{a} Y' \Rightarrow \exists X'$ with $X \xrightarrow{a} X'$ and $(X', Y') \in R$.

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

In [GG a] it has been shown that the configurations of a refined event structure may be deduced compositionally from the configurations of the original event structure and those of the refinements of actions. We will use this result and the following lemmas in the proof of the refinement theorem.

Definition 4.2 Let $\mathcal{E} \in \mathbb{E}$, let *ref* be a refinement function.

(i) We call $\tilde{X}$ a *refinement of a configuration* $X \in \mathit{Conf}(\mathcal{E})$ *by ref* iff

- $\tilde{X} = \bigcup_{e \in X} (\{e\} \times X_e)$ where $\forall e \in X : X_e \in \mathit{Conf}(\mathit{ref}(l_{\mathcal{E}}(e))) - \{\emptyset\}$,
- $\mathit{busy}(\tilde{X}) \subseteq \mathit{max}(X)$ where
    - · $\mathit{max}(X)$ denotes the set of all maximal events in X w.r.t. $<_X$,
    - · $\mathit{busy}(\tilde{X}) := X - \mathit{compl}(\tilde{X})$ and
    - · $\mathit{compl}(\tilde{X}) := \{e \in X \mid X_e \text{ is a complete configuration}\}$.

(ii) $\mathit{ref}(X)$ denotes the set of refinements of $X \in \mathit{Conf}(\mathcal{E})$ by *ref* and for $S \subseteq \mathit{Conf}(\mathcal{E})$, $\mathit{ref}(S) = \bigcup_{X \in S} \mathit{ref}(X)$.

Clearly, if $\tilde{X} \in \mathit{ref}(X)$ then $X = \mathit{pr}(\tilde{X}) := \{e \in E_{\mathcal{E}} \mid \exists e' \in E_{\mathit{ref}(l(e))} : (e, e') \in \tilde{X}\}$ and we have $\mathit{compl}(\tilde{X}) \subseteq \mathit{pr}(\tilde{X})$, $\mathit{pr}(\tilde{X}) - \mathit{max}(\mathit{pr}(\tilde{X})) \subseteq \mathit{compl}(\tilde{X})$.

Proposition 4.1 [GG a] Let $\mathcal{E} \in \mathbb{E}$, let *ref* be a refinement function.

Then $\mathit{Conf}(\mathit{ref}(\mathcal{E})) = \mathit{ref}(\mathit{Conf}(\mathcal{E}))$.

Lemma 4.1 Let $\mathcal{E} \in \mathbb{E}$, *ref* a refinement function and $\tilde{X} \in \mathit{Conf}(\mathit{ref}(\mathcal{E}))$.

If $\mathit{compl}(\tilde{X}) \subseteq Y \subseteq \mathit{pr}(\tilde{X})$ then $Y \in \mathit{Conf}(\mathcal{E})$.

Proof Immediate from proposition 2.1(iii), since $\mathit{pr}(\tilde{X}) - \mathit{max}(\mathit{pr}(\tilde{X})) \subseteq Y \subseteq \mathit{pr}(\tilde{X})$. □

With lemma 4.1 we have in particular $\mathit{compl}(\tilde{X}) \in \mathit{Conf}(\mathcal{E})$.

Lemma 4.2 Let $\mathcal{E} \in \mathbb{E}$, $X \in \mathit{Conf}(\mathcal{E})$ and $a \notin \mathit{Crit}_{\mathcal{E}}$.

$X \xrightarrow{a} X_1,\ X \xrightarrow{a} X_2 \Rightarrow X_1 = X_2$.

Proof Immediate from the definitions. □

Lemma 4.3 Let $\mathcal{E} \in \mathbb{E}$, *ref* a safe refinement function w.r.t. $\mathcal{E}$ and $\tilde{X} \in \mathit{Conf}(\mathit{ref}(\mathcal{E}))$.

If $e \in \mathit{pr}(\tilde{X}) - \mathit{compl}(\tilde{X})$ then $l_{\mathcal{E}}(e) \notin \mathit{Crit}_{\mathcal{E}}$.

Proof With proposition 4.1 we have $X_e \in \mathit{Conf}(\mathit{ref}(l_{\mathcal{E}}(e))) - \{\emptyset\}$.
If $l_{\mathcal{E}}(e) \in \mathit{Crit}_{\mathcal{E}}$ than X_e is complete by proposition 2.4.
However, since $e \in \mathit{pr}(\tilde{X}) - \mathit{compl}(\tilde{X})$, X_e is not complete. □

Lemma 4.4 Let $\mathcal{E} \in \mathbb{E}$, *ref* a safe refinement function w.r.t. $\mathcal{E}$ and $\tilde{X} \in \mathit{Conf}(\mathit{ref}(\mathcal{E}))$.

If $\mathit{compl}(\tilde{X}) \cup \{e\} \in \mathit{Conf}(\mathcal{E})$ then $\mathit{pr}(\tilde{X}) \cup \{e\} \in \mathit{Conf}(\mathcal{E})$.

Proof W.l.o.g. we assume that $e \notin \mathit{pr}(\tilde{X})$.
With proposition 2.1(i) we prove that $\mathit{pr}(\tilde{X})$ enables e.

Suppose $\exists d \in \mathit{pr}(\tilde{X})$ with $d \# e$. Since $\mathit{compl}(\tilde{X}) \cup \{e\}$ is conflict-free, it follows that $d \in \mathit{pr}(\tilde{X}) - \mathit{compl}(\tilde{X})$. Thus $\mathit{compl}(\tilde{X})$ enables d (with lemma 4.1) as well as e, so $d\ \mathit{ch}\ e$. However, with lemma 4.3 we have $l(d) \notin \mathit{Crit}_{\mathcal{E}}$, contradicting $d\ \mathit{ch}\ e$.

Now suppose $d \prec e$ and $d \notin \mathit{pr}(\tilde{X}) \supseteq \mathit{compl}(\tilde{X})$. Since $\mathit{compl}(\tilde{X}) \cup \{e\} \in \mathit{Conf}(\mathcal{E})$ there is an $f \in \mathit{compl}(\tilde{X}) \subseteq \mathit{pr}(\tilde{X})$ with $f \prec e$ and $d \# f$. □

With these lemmas we can prove the theorem.

Theorem 4.1 Let $\mathcal{E}, \mathcal{F} \in \mathbb{E}$, let *ref* be a safe refinement function w.r.t. $\mathcal{E}$ and $\mathcal{F}$.

Then $\quad \mathcal{E} \approx_{ib} \mathcal{F} \quad \Rightarrow \quad \mathit{ref}(\mathcal{E}) \approx_{ib} \mathit{ref}(\mathcal{F})$.

Proof Let R be a bisimulation between $\mathcal{E}$ and $\mathcal{F}$.
Let $\tilde{R} = \{(\tilde{X}, \tilde{Y}) \in \mathit{Conf}(\mathit{ref}(\mathcal{E})) \times \mathit{Conf}(\mathit{ref}(\mathcal{F})) \mid$
$(\mathit{compl}(\tilde{X}), \mathit{compl}(\tilde{Y})) \in R$,
there exist bijections $f : \mathit{pr}(\tilde{X}) \longrightarrow \mathit{pr}(\tilde{Y})$ with $l_{\mathcal{F}}(f(e)) = l_{\mathcal{E}}(e)$
and $\tilde{f} : \tilde{X} \longrightarrow \tilde{Y}$ with $\tilde{f}(e, e') = (f(e), e')\}$

We show that $\tilde{R}$ is a bisimulation between $\mathit{ref}(\mathcal{E})$ and $\mathit{ref}(\mathcal{F})$.

i. Obviously $(\emptyset, \emptyset) \in \tilde{R}$.

ii. Let $(\tilde{X}, \tilde{Y}) \in \tilde{R}$. Then $(\mathit{compl}(\tilde{X}), \mathit{compl}(\tilde{Y})) \in R$ and there are bijections f and $\tilde{f}$ as required in the definition of $\tilde{R}$. We have to show the bisimulation properties. It suffices to check one of them, the other follows by symmetry.

Let $\tilde{X} \xrightarrow{a} \tilde{X}'$. Let $\tilde{X}' - \tilde{X} = \{(e^\circ, e^*)\}$, $l_{\mathcal{E}}(e^\circ) = b$. Then $l_{\mathit{ref}(\mathcal{E})}((e^\circ, e^*)) = l_{\mathit{ref}(b)}(e^*) = a$.
Now we have to show : $\exists \tilde{Y}' \in \mathit{Conf}(\mathit{ref}(\mathcal{F}))$ with $\tilde{Y} \xrightarrow{a} \tilde{Y}'$ and $(\tilde{X}', \tilde{Y}') \in \tilde{R}$.
For the construction of $\tilde{Y}'$, we distinguish two cases.

1. $\mathit{pr}(\tilde{X}') = \mathit{pr}(\tilde{X})$.

In this case we can use the bijection f to define $\tilde{Y}'$, since its domain is not affected by adding the new event.

Let $\tilde{Y}' := \{(f(e), e') \mid (e, e') \in \tilde{X}'\}$.
With $\tilde{Y} = \{(f(e), e') \mid (e, e') \in \tilde{X}\}$ we have $\tilde{Y}' = \tilde{Y} \cup \{(f(e^\circ), e^*)\}$.

First we show that $\tilde{Y}'$ is a configuration with $\tilde{Y} \xrightarrow{a} \tilde{Y}'$.
Then we define the new functions f' and $\tilde{f}'$ and check their properties.
Finally we prove $(\mathit{compl}(\tilde{X}'), \mathit{compl}(\tilde{Y}')) \in R$.

With proposition 4.1 we show $\tilde{Y}' \in \mathit{Conf}(\mathit{ref}(\mathcal{F}))$:

- $\mathit{pr}(\tilde{Y}') = f(\mathit{pr}(\tilde{X}')) = f(\mathit{pr}(\tilde{X})) = \mathit{pr}(\tilde{Y}) \in \mathit{Conf}(\mathcal{F})$, since $\tilde{Y} \in \mathit{Conf}(\mathit{ref}(\mathcal{F}))$;
- $\forall d \in \mathit{pr}(\tilde{Y}')$: $Y'_d = X'_{f^{-1}(d)} \in \mathit{Conf}(\mathit{ref}(l_{\mathcal{E}}(f^{-1}(d)))) - \{\emptyset\} = \mathit{Conf}(\mathit{ref}(l_{\mathcal{F}}(d))) - \{\emptyset\}$, since $\tilde{X}' \in \mathit{Conf}(\mathit{ref}(\mathcal{E}))$;
- $\mathit{busy}(\tilde{Y}') = \mathit{pr}(\tilde{Y}') - \mathit{compl}(\tilde{Y}') \subseteq \mathit{pr}(\tilde{Y}) - \mathit{compl}(\tilde{Y}) \subseteq \mathit{max}(\mathit{pr}(\tilde{Y})) = \mathit{max}(\mathit{pr}(\tilde{Y}'))$, since $\mathit{pr}(\tilde{Y}) = \mathit{pr}(\tilde{Y}')$ and $\mathit{compl}(\tilde{Y}) \subseteq \mathit{compl}(\tilde{Y}')$.

Thus $\tilde{Y}'$ is a configuration of $\mathit{ref}(\mathcal{F})$.
Since $l_{\mathit{ref}(\mathcal{F})}((f(e^\circ), e^*)) = l_{\mathit{ref}(b)}(e^*) = a$ we have $\tilde{Y} \xrightarrow{a} \tilde{Y}'$.

Let $f' := f$ and $\tilde{f}' := \tilde{f} \cup \{((e^\circ, e^*), (f(e^\circ), e^*))\}$. The new function $\tilde{f}'$ is bijective, since the events (e°, e^*) and $(f(e^\circ), e^*)$ do not occur in the domain and the range of $\tilde{f}$, respectively. By construction f' and $\tilde{f}'$ have the required properties.

In order to establish $(\tilde{X}', \tilde{Y}') \in \tilde{R}$ we still have to show $(\mathit{compl}(\tilde{X}'), \mathit{compl}(\tilde{Y}')) \in R$.
Again we have two cases:

1.1. $\mathit{compl}(\tilde{X}') = \mathit{compl}(\tilde{X})$.

Using the properties of $\tilde{f}'$, $\mathit{compl}(\tilde{Y}') = f'(\mathit{compl}(\tilde{X}')) = f(\mathit{compl}(\tilde{X})) = \mathit{compl}(\tilde{Y})$, hence $(\mathit{compl}(\tilde{X}'), \mathit{compl}(\tilde{Y}')) \in R$.

1.2. $\mathit{compl}(\tilde{X}') \neq \mathit{compl}(\tilde{X})$.

Then $\mathit{compl}(\tilde{X}') = \mathit{compl}(\tilde{X}) \cup \{e^\circ\}$ and $\mathit{compl}(\tilde{X}) \xrightarrow{b} \mathit{compl}(\tilde{X}')$.

Since $(\mathit{compl}(\tilde{X}), \mathit{compl}(\tilde{Y})) \in R$ there is a $Y' \in \mathit{Conf}(\mathcal{F})$ with $\mathit{compl}(\tilde{Y}) \xrightarrow{b} Y'$ and $(\mathit{compl}(\tilde{X}'), Y') \in R$.

Using the properties of $\tilde{f}'$, $\mathit{compl}(\tilde{Y}') = f'(\mathit{compl}(\tilde{X}')) = f(\mathit{compl}(\tilde{X}) \cup \{e^\circ\}) = \mathit{compl}(\tilde{Y}) \cup \{f(e^\circ)\}$, hence $\mathit{compl}(\tilde{Y}) \xrightarrow{b} \mathit{compl}(\tilde{Y}')$.

Since $e^\circ \in \mathit{compl}(\tilde{X}') \subseteq \mathit{pr}(\tilde{X}') = \mathit{pr}(\tilde{X})$, it follows that $e^\circ \in \mathit{pr}(\tilde{X}) - \mathit{compl}(\tilde{X})$ and with lemma 4.3 $b = l_{\mathcal{E}}(e^\circ) \notin \mathit{Crit}_{\mathcal{E}}$. Hence $Y' = \mathit{compl}(\tilde{Y}')$ with lemma 4.2.

Thus $(\mathit{compl}(\tilde{X}'), \mathit{compl}(\tilde{Y}')) \in R$, which had to be proved.

2. $\mathit{pr}(\tilde{X}') \neq \mathit{pr}(\tilde{X})$.

The proof of the second case is similar to that of the first case. However the construction of $\tilde{Y}'$ is more complicated since we cannot just use the function f. We need to describe the events by which f is extended.

$\mathit{pr}(\tilde{X}') = \mathit{pr}(\tilde{X}) \cup \{e^\circ\}$ and $l_{\mathcal{E}}(e^\circ) = b$.

$$\mathit{compl}(\tilde{X}') = \begin{cases} \mathit{compl}(\tilde{X}) \cup \{e^\circ\} & \text{if } \{e^*\} \text{ is a complete configuration of } \mathit{ref}(b) \\ \mathit{compl}(\tilde{X}) & \text{otherwise} \end{cases}$$

Thus $\mathit{compl}(\tilde{X}') \subseteq \mathit{compl}(\tilde{X}) \cup \{e^\circ\} \subseteq \mathit{pr}(\tilde{X}) \cup \{e^\circ\} = \mathit{pr}(\tilde{X}')$.
With lemma 4.1 it follows that $\mathit{compl}(\tilde{X}) \cup \{e^\circ\} \in \mathit{Conf}(\mathcal{E})$.
Hence $\mathit{compl}(\tilde{X}) \xrightarrow{b} \mathit{compl}(\tilde{X}) \cup \{e^\circ\}$.
Since $(\mathit{compl}(\tilde{X}), \mathit{compl}(\tilde{Y})) \in R$, there is a $d^\circ \in E_{\mathcal{F}}$ with $\mathit{compl}(\tilde{Y}) \xrightarrow{b} \mathit{compl}(\tilde{Y}) \cup \{d^\circ\}$, $l_{\mathcal{F}}(d^\circ) = b$ and $(\mathit{compl}(\tilde{X}) \cup \{e^\circ\}, \mathit{compl}(\tilde{Y}) \cup \{d^\circ\}) \in R$.

We now extend the bijection f by (e°, d°).
For this we verify that d° is not in the range of f:
Suppose $d^\circ \in \mathit{pr}(\tilde{Y})$. Since $d^\circ \notin \mathit{compl}(\tilde{Y})$, $f^{-1}(d^\circ) \in \mathit{pr}(\tilde{X}) - \mathit{compl}(\tilde{X})$. Thus $\mathit{compl}(\tilde{X}) \cup \{f^{-1}(d^\circ)\} \in \mathit{Conf}(\mathcal{E})$ with lemma 4.1 and $\mathit{compl}(\tilde{X}) \xrightarrow{b} \mathit{compl}(\tilde{X}) \cup \{f^{-1}(d^\circ)\}$.
Hence $b \notin \mathit{Crit}_{\mathcal{E}}$ with lemma 4.3 and then $e^\circ = f^{-1}(d^\circ)$ with lemma 4.2.
Thus $e^\circ \in \mathit{pr}(\tilde{X})$ which contradicts $\mathit{pr}(\tilde{X}') \neq \mathit{pr}(\tilde{X})$. So $d^\circ \notin \mathit{pr}(\tilde{Y})$.

Now we can define $\tilde{Y}', f'$ and $\tilde{f}'$.
Let $f' := f \cup (e^\circ, d^\circ)$ and $\tilde{Y}' := \{(f'(e), e') \mid (e, e') \in \tilde{X}'\} = \tilde{Y} \cup \{(d^\circ, e^*)\}$.

Using proposition 4.1 we proof $\tilde{Y}' \in \mathit{Conf}(\mathit{ref}(\mathcal{F}))$:

- $\mathit{pr}(\tilde{Y}') = \mathit{pr}(\tilde{Y}) \cup \{d^\circ\} \in \mathit{Conf}(\mathcal{F})$ with lemma 4.4;
- $\forall d \in \mathit{pr}(\tilde{Y}')$: $Y'_d = X'_{f'^{-1}(d)} \in \mathit{Conf}(\mathit{ref}(l_{\mathcal{E}}(f'^{-1}(d)))) - \{\emptyset\} = \mathit{Conf}(\mathit{ref}(l_{\mathcal{F}}(d))) - \{\emptyset\}$;

- $busy(\tilde{Y}') = pr(\tilde{Y}') - compl(\tilde{Y}') = pr(\tilde{Y}) \cup \{d^\circ\} - compl(\tilde{Y}') \subseteq$
  $pr(\tilde{Y}) \cup \{d^\circ\} - compl(\tilde{Y})$, since $compl(\tilde{Y}) \subseteq compl(\tilde{Y}')$.
  It remains to show that $pr(\tilde{Y}) \cup \{d^\circ\} - compl(\tilde{Y}) \subseteq max(pr(\tilde{Y}) \cup \{d^\circ\})$.
  By proposition 2.1(iii) $pr(\tilde{Y})$ is a prefix of $pr(\tilde{Y}) \cup \{d^\circ\}$, so $d^\circ \in max(pr(\tilde{Y}) \cup \{d^\circ\})$.
  Now suppose $d \in pr(\tilde{Y}) - compl(\tilde{Y})$, but $d \notin max(pr(\tilde{Y}) \cup \{d^\circ\})$.
  Then $d \in max(pr(\tilde{Y}))$ by proposition 4.1, so $d <_{pr(\tilde{Y}) \cup \{d^\circ\}} d^\circ$. Since $compl(\tilde{Y}) \cup \{d^\circ\}$ is a prefix of $pr(\tilde{Y}) \cup \{d^\circ\}$ by proposition 2.1(iii), we have $d \in compl(\tilde{Y})$, which yields a contradiction.

Thus $\tilde{Y}'$ is a configuration of $ref(\mathcal{F})$ and with $l_{ref(\mathcal{F})}(d^\circ, e^*) = a$ we have $\tilde{Y} \xrightarrow{a} \tilde{Y}'$.

The new function f' is bijective, as shown above. The same for $\tilde{f}' := \tilde{f} \cup \{((e^\circ, e^*), (d^\circ, d^*))\}$. By construction f' and $\tilde{f}'$ have the required properties.

Now we show $(\tilde{X}', \tilde{Y}') \in \tilde{R}$ by proving $(compl(\tilde{X}'), compl(\tilde{Y}')) \in R$.
There are again two cases:

2.1. $compl(\tilde{X}') = compl(\tilde{X})$.
By construction of $\tilde{Y}'$ (as in part 1.1 of the proof) $compl(\tilde{Y}') = compl(\tilde{Y})$ and hence $(compl(\tilde{X}'), compl(\tilde{Y}')) \in R$.

2.2. $compl(\tilde{X}') \neq compl(\tilde{X})$.
Then $compl(\tilde{X}') = compl(\tilde{X}) \cup \{e^\circ\}$ and by construction of $\tilde{Y}'$, $compl(\tilde{Y}') = compl(\tilde{Y}) \cup \{d^\circ\}$. We had already that $(compl(\tilde{X}) \cup \{e^\circ\}, compl(\tilde{Y})) \cup \{d^\circ\}) \in R$, hence $(compl(\tilde{X}'), compl(\tilde{Y}')) \in R$.

Thus $\tilde{R}$ is a bisimulation between $ref(\mathcal{E})$ and $ref(\mathcal{F})$. □

Conclusion

We have investigated here how two well-established interleaving equivalences behave with respect to a restricted notion of action refinement. We have considered a class of refinements where events deciding choices are considered atomic. It turned out that interleaving trace semantics — neglecting the branching structure — is then still not preserved under refinements whereas interleaving bisimulation yields the desired preservation result.

It remains to be investigated what happens for other interleaving equivalences in the linear time – branching time spectrum, e.g. for failure semantics. Another interesting question is what happens for step semantics (where several actions happening simultaneously are considered). The example $a(bc|d)$ versus $ab(c|d) + a(b|d)c$ with $ref(d) = d_1 d_2$ may be used to show that *step trace equivalence* is not preserved under safe refinements as considered here. For *step bisimulation equivalence* we conjecture that we do have preservation under safe refinements.

Acknowledgements
The idea to investigate refinements with atomic choice was suggested by Manfred Broy. Three anonymous referees helped with their comments to improve the paper.

References

[**Aceto**] L. Aceto: *Action Refinement in Process Algebras*, PhD-Thesis, University of Sussex, 1990, Report No. 3/91, University of Sussex, Computer Science, February 1991

[**AH**] L. Aceto, M. Hennessy: *Adding Action Refinement to a Finite Process Algebra*, Report No. 6/90, University of Sussex, Computer Science, November 1990, extended abstract in: Proc. ICALP 91, LNCS 510, Springer-Verlag, pp 506–519, 1991

[**BDKP**] E. Best, R. Devillers, A. Kiehn, L. Pomello: *Concurrent Bisimulation in Petri Nets*, Acta Informatica, Vol. 28, pp 231–264, 1991

[**Boudol**] G. Boudol: *Flow Event Structures and Flow Nets*, in I. Guessarian (ed.): Semantics of Systems of Concurrent Processes, LNCS 469, Springer-Verlag, pp 62–95, 1990

[**BC**] G. Boudol, I. Castellani: *Permutation of Transitions: An Event Structure Semantics for CCS and SCCS*, in J.W. de Bakker, W.-P. de Roever & G. Rozenberg (eds.): Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354, Springer-Verlag, pp 411–427, 1989

[**BGV**] W. Brauer, R. Gold, W. Vogler: *A Survey of Behaviour and Equivalence Preserving Refinements of Petri Nets*, in: Advances in Petri Nets 1990, LNCS 483, Springer-Verlag, pp 1–46, 1990

[**DD a**] Ph. Darondeau, P. Degano: *Event structures, Causal trees, and Refinements*, in: Proc. MFCS 90, LNCS 452, Springer-Verlag, pp 239–245, 1990

[**DD b**] Ph. Darondeau, P. Degano: *Refinement of Actions in Event Structures and Causal Trees*, manuscript, 1991

[**Devillers**] R. Devillers: *Maximality Preserving Bisimulation*, Technical Report LIT-1214, Université Libre de Bruxelles, March 1990, to appear in TCS

[**GG a**] R.J. van Glabbeek, U. Goltz: *Refinement of Actions in Causality Based Models*, in J.W. de Bakker, W.-P. de Roever & G. Rozenberg (eds.): Stepwise Refinement of Distributed Systems: Models, Formalism, Correctness, LNCS 430, Springer-Verlag, pp 267–300, 1990

[**GG b**] R.J. van Glabbeek, U. Goltz: *Equivalences and Refinement*, in: I. Guessarian (ed.): Semantics of Systems of Concurrent Processes, LNCS 469, Springer-Verlag, pp 309–333, 1990

[GG c] R.J. van Glabbeek, U. Goltz: *A Deadlock-sensitive Congruence for Action Refinement*, SFB-Bericht Nr. 342/23/90 A, TUM-19044, Technische Universität München, November 1990

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

[NEL] M. Nielsen, U. Engberg, K. S. Larsen: *Fully Abstract Models for a Process Language with Refinement*, in J.W. de Bakker, W.-P. de Roever & G. Rozenberg (eds.): Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, LNCS 354, Springer-Verlag, pp 523–548, 1989

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

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

[Vogler a] W. Vogler: *Failures Semantics Based on Interval Semiwords is a Congruence for Refinement*, Distributed Computing, Vol. 4, pp 139–162, 1991

[Vogler b] W. Vogler: *Bisimulation and Action Refinement*, in: Proc. STACS 91, LNCS 480, Springer-Verlag, pp 309–321, 1991