REIHE INFORMATIK
14/96

Some Properties of Refinement

Friederike Benjes
Universität Mannheim
Seminargebäude A5
D-68131 Mannheim

Some Properties of Refinement

Friederike Benjes

August 26, 1996

1 Introduction

In concurrent reactive systems refinement has two aspects: there is the notion of a horizontal refinement and of a vertical refinement. A typical instance of horizontal refinement is the substitution of a sequential process S by a system of parallel processes that exhibits the same communication behaviour as S.

In contrast, vertical refinement or action refinement deals with the substitution of a communication by a process.

We are here dealing with action refinement. Action refinement has been considered as a syntactic operation ([AH91], [AH93], [NEL89]) as well as a semantic operation.

We consider here two problems concerning refinement.
The first deals with (semantic) refinement in flow event structures and configuration structures. It is well-known ([CZ89], [BC91]) that there may exist configurations in the product $\mathcal{E}_1 \parallel_A \mathcal{E}_2$ of event structures that do not map to configurations of the individual event structures under projection $(\exists \mathcal{E}_1, \mathcal{E}_2 : \exists X \in \mathit{Conf}(\mathcal{E}_1 \parallel_A \mathcal{E}_2) : \Pi_1(X) \notin \mathit{Conf}(\mathcal{E}_1))$.
[Sch91] and [CZ89] showed that languages using only the operators $+, ;, \parallel_A$ but no refinement operator do not create event structures of that type. [Co95] attempted to show the same for languages with a refinement operator – however their proof contained a mistake. In the first part we will give a corrected proof.

The second problem deals with the connection of syntactic and semantic refinement.
In [GGR92] a notion of syntactic refinement was defined and compared with semantic refinement on flow event structures. It was shown that in the case of refining synchronizing actions syntactic and semantic refinement coincide only under fairly restrictive conditions.

We present a different notion of syntactic refinement, that can be seen as motivated by [DG95] who proposed a "parallel call of procedure" for refining synchronizing actions. The basic idea is to consider the refinement of a synchronizing action as the call of a procedure to which both partners have to "inscribe". This view still leaves the question open of who takes control for this procedure. We investigate a variant where one of the synchronizing processes takes control. We show that this notion of refinement is much closer to the semantic notion of refinement by presenting a criterion that guarantees that the two kinds of refinement coincide and that is satisfied in most cases.

In section 2 we give some elementary definitions and propositions. In section 3 refinement is be defined on prime and flow event structures and on configuration structures. Chapter 4 deals with the consistency of a flow event structure semantics and a configuration structure semantics. In section 5 the new definition of syntactic refinement is introduced and compared to semantic refinement. And finally the appendix contains the main proof for section 5.

2 Foundations

2.1 The Language L

Definition 2.1.1: Let *Act* be a countable set of actions, $a \in Act$. The following grammar defines the terms of the language **L**:

$P ::= a \mid P; P \mid P + P \mid P \parallel_A P \mid P[a \rightsquigarrow P]$

We use a family of synchronization operators $\{\parallel_A\}_{A \subseteq Act}$ corresponding to the TCSP approach. The refinement operator $[a \rightsquigarrow P]$ acts on single actions $a \in Act$ at a time.

$\mathbf{L^o} \subseteq \mathbf{L}$ denotes the set of terms not containing refinement operators.

Definition 2.1.2: Let $P \in \mathbf{L}$. Then $L(P)$ is the set of actions occuring in P. $S(P)$ is the set of synchronizing actions of the term P. (see [GGR92]):

$$\begin{array}{lcl}
L(a) & := & \{a\} \\
L(P_1 + P_2) & := & L(P_1) \cup L(P_2) \\
L(P_1; P_2) & := & L(P_1) \cup L(P_2) \\
L(P_1 \parallel_A P_2) & := & L(P_1) \cup L(P_2) \cup A \\
L(P_1[a \rightsquigarrow P_2]) & := & \begin{cases} (L(P_1) \setminus \{a\}) \cup L(P_2) & \text{if } a \in L(P_1) \\ L(P_1) & \text{otherwise} \end{cases}
\end{array}$$

$$\begin{array}{lcl}
S(a) & := & \emptyset \\
S(P_1 + P_2) & := & S(P_1) \cup S(P_2) \\
S(P_1; P_2) & := & S(P_1) \cup S(P_2) \\
S(P_1 \parallel_A P_2) & := & S(P_1) \cup S(P_2) \cup ((L(P_1) \cup L(P_2)) \cap A) \\
S(P_1[a \rightsquigarrow P_2]) & := & \begin{cases} (S(P_1) \setminus \{a\}) \cup L(P_2) & \text{if } a \in S(P_1) \\ S(P_1) \cup S(P_2) & \text{if } a \in L(P_1) \setminus S(P_1) \\ S(P_1) & \text{otherwise} \end{cases}
\end{array}$$

Terms are called well-formed, if all the actions introduced by a refinement operator applied to P, i.e. $P[a \rightsquigarrow Q]$ are different (more concrete) from the actions in P. We define well-formedness by induction on the syntactical structure of terms:

Definition 2.1.3: (well-formedness)
All actions $a \in Act$ are well-formed.
If P and Q are well-formed, so are $P; Q$, $P + Q$, $P \parallel_A Q$.
If P and Q are well-formed and $L(P) \cap L(Q) = \emptyset$ then $P[a \rightsquigarrow Q]$ is well-formed.
(see [GGR92])

Syntactic substitution almost corresponds to textual replacement. It is defined by induction on the syntactical structure of a term – but it is only defined on terms not containing refinement operators.

Definition 2.1.4: (syntactic substitution $\{\frac{Q}{a}\}$)
Let P, Q, P_1, P_2 be in $\mathbf{L^o}$, $a, b \in Act$, $A \subseteq Act$.

$$b\{\tfrac{Q}{a}\} := \begin{cases} Q & \text{if } b = a \\ b & \text{otherwise} \end{cases}$$

$$(P_1; P_2)\{\tfrac{Q}{a}\} := P_1\{\tfrac{Q}{a}\}; P_2\{\tfrac{Q}{a}\}$$

$$(P_1 + P_2)\{\tfrac{Q}{a}\} := P_1\{\tfrac{Q}{a}\} + P_2\{\tfrac{Q}{a}\}$$

$$(P_1 \parallel_A P_2)\{\tfrac{Q}{a}\} := \begin{cases} P_1\{\tfrac{Q}{a}\} \parallel_A P_2\{\tfrac{Q}{a}\} & \text{if } a \notin A \\ P_1\{\tfrac{Q}{a}\} \parallel_{A \setminus \{a\} \cup L(Q)} P_2\{\tfrac{Q}{a}\} & \text{if } a \in A \end{cases}$$

(see [GGR92])

2.2 Prime Event Structures

Definition 2.2.1: (prime event structure)
$\mathcal{E} = (E, \leq, \#, l)$ is a *prime event structure* labelled over Act iff

- E is a countable set of events
- $\leq \subseteq E \times E$ is a partial order (causal relation)
- $\# \subseteq E \times E$ is an irreflexive and symmetric relation (conflict relation)
- $l : E \rightarrow Act$ is a labelling function

and

- $\forall e \in E : \downarrow e = \{e' \in E \mid e' \leq e\}$ is finite (principle of finite causes)
- $\forall e, e', e'' \in E : e \# e' \leq e'' \Rightarrow e \# e''$ (principle of conflict heredity)

The class of all prime event structures is denoted by $\mathbf{P}$. The empty prime event structure is denoted by $\emptyset$.

Definition 2.2.2: (configuration of a prime event structure)
Let $\mathcal{E} = (E, \leq, \#, l)$ be a labelled prime event structure. A subset $X \subseteq E$ is called *configuration* of $\mathcal{E}$ iff

- X is conflict-free, i.e. $\forall d, e \in X : \neg(d \# e)$ and
- X is left-closed, i.e. $\forall e \in X : \downarrow e \subseteq X$.

A configuration X of $\mathcal{E}$ is called *complete* iff $\forall e \in E \setminus X : \exists e' \in X : e' \# e$.
A configuration X of $\mathcal{E}$ is called *maximal* iff $\forall X' \in Conf(\mathcal{E}') : X \not\subset X'$.

Remark 2.2.3 Configurations of prime event structures are maximal iff they are complete.

Definition 2.2.4: Let $\mathcal{E} = (E, \leq, \#, l)$ be a labelled prime event structure.

a) *Conf*$(\mathcal{E})$ is the set of all configurations of $\mathcal{E}$ (the finite ones and the infinite ones).
Conf$_f(\mathcal{E})$ is the set of all finite configurations of $\mathcal{E}$.

b) Let $e \in E$. $\downarrow e$ denotes the set of all events that have to occur before e:
$\downarrow e := \{e' \in E \mid e' \leq e\}$.

2.3 Flow Event Structures

Definition 2.3.1: (flow event structure)
$\mathcal{E} = (E, \prec, \#, l)$ is a *flow event structure* labelled over *Act* iff

- E is a countable set of events
- $\prec \subseteq E \times E$ is irreflexive (flow relation)
- $\# \subseteq E \times E$ symmetric (conflict relation)
- $l : E \to Act$ labelling function

The class of all flow event structures is denoted by $\mathbf{F}$. $\emptyset$ denotes the empty flow event structure.

Definition 2.3.2: (configuration of a flow event structure)
Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure. A subset $X \subseteq E$ is called *configuration* of $\mathcal{E}$ iff

(i) X conflict-free, i. e. $\forall d, e \in X : \neg(d\#e)$,

(ii) $\leq_X := (\prec \cap (X \times X))^*$ (the reflexive and transitive closure of $\prec$ in X) is a partial order, i.e. $\prec$ is cycle-free on X,

(iii) $\forall e \in X$: $\{e' \in X \mid e' \leq_X e\}$ is finite (principle of finite causes) and

(iv) $\forall e \in X \; \forall e' \in E \setminus X : e' \prec e \Rightarrow \exists e'' \in X : \; e'\#e'' \prec e$ (X left-closed up to conflicts).

Like configurations in prime event structures a configuration is called *complete* iff
$\forall e \in E \setminus X : \exists e' \in X : e'\#e$.
A configuration is *maximal* iff it is maximal with respect to inclusion.

Remark 2.3.3 Configurations of flow event structures can be maximal without being complete (see [GG90]).

Definition 2.3.4: Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure.

a) *Conf*$(\mathcal{E})$ is the set of all configurations of $\mathcal{E}$ (the finite ones and the infinite ones).
Conf$_f(\mathcal{E})$ is the set of all finite configurations of $\mathcal{E}$.

b) Let e be in E, $X \in$ *Conf*$(\mathcal{E})$. Then $\downarrow_X e := \{e' \in X \mid e' \leq_X e\}$.

Lemma 2.3.5 Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure, $X, Y \in Conf(\mathcal{E})$ and $e \in X, e \in Y$. If $\downarrow_X(e) \neq \downarrow_Y(e)$ then exists $x \in X$ & $y \in Y : x\#y$.

Proof:
W.l.o.g. let $b \in \downarrow_X(e)$ and $b \notin \downarrow_Y(e)$. Then exist
$x_1, ..., x_{n+1} \in \downarrow_X(e) : b = x_1 \prec ... \prec x_n \prec x_{n+1} = e$. Then exists $j \leq n : x_j \notin \downarrow_Y(e)$ and $x_{j+1} \in \downarrow_Y(e)$. Since $x_j \prec x_{j+1}$ and $\downarrow_Y(e)$ is a configuration exists $y \in \downarrow_Y(e) : x_j\#y \prec x_{j+1}$.
Thus $x_j \in \downarrow_X(e), y \in \downarrow_Y(e)$ and $x_j\#y$.

■

Remark 2.3.6 Each prime event structure $\mathcal{E} = (E, \leq, \#, l)$ can also be seen as a flow event structure $\mathcal{E}' = (E, <, \#, l)$. On prime event structures the definition for configurations of prime event structures coincides with the one for flow event structures.

2.4 Domains

Definition 2.4.1: (domains)
Let $(D, \sqsubseteq), (D', \sqsubseteq')$ be partial orders.

$(D, \sqsubseteq)$ is *isomorphic* to $(D', \sqsubseteq')$ ($(D, \sqsubseteq) \cong (D', \sqsubseteq')$) if exists $f : D \to D'$, f bijective and
$\forall d_1, d_2 \in D : d_1 \sqsubseteq d_2 \Leftrightarrow f(d_1) \sqsubseteq' f(d_2)$.

An element $d \in D$ is called *least upper bound* of $X \subseteq D$ ($d = \bigsqcup X$) iff
$(\forall x \in X : x \sqsubseteq d)$ & $(\forall d' \in D : (\forall x \in X : x \sqsubseteq d') \Rightarrow d \sqsubseteq d')$.

An element $p \in D$ is called a *complete prime* iff for any $X \subseteq D$ with $\bigsqcup X \in D$:
$p \sqsubseteq \bigsqcup X \Rightarrow \exists x \in X : p \sqsubseteq x$.
$P(D) := \{p \in D \mid p \text{ is a complete prime }\}$.

Two elements $x, y \in D$ are called *consistent* ($x \uparrow y$) iff $\exists z \in D : x \sqsubseteq z$ & $y \sqsubseteq z$.

$X \subseteq D$ is called *pairwise consistent* iff $\forall x, y \in X : x \uparrow y$.

$(D, \sqsubseteq)$ is called *coherent* iff every pairwise consistent subset $X \subseteq D$ has a least upper bound $\bigsqcup X$ in D.

$(D, \sqsubseteq)$ is called *finitary* iff $\forall p \in P(D) : \downarrow p := \{d \in D \mid d \sqsubseteq p\}$ is finite.

$(D, \sqsubseteq)$ is called *(ω)-prime algebraic* iff $P(D)$ is countable and
$\forall d \in D : d = \bigsqcup\{p \in P(D) \mid p \sqsubseteq d\}$.

We call any finitary coherent (ω)-prime algebraic domain a *domain*.

(see [Bo90])

Lemma 2.4.2: For any unlabelled prime event structure $\mathcal{E}$ the poset $(Conf(\mathcal{E}), \subseteq)$ is a domain, and any domain $(D, \sqsubseteq)$ is isomorphic to the poset of configurations of a prime event structure. More specifically we have $(D, \sqsubseteq) \cong (Conf(\mathcal{K}(D, \sqsubseteq)), \subseteq)$ with
$\mathcal{K}(D, \sqsubseteq) := (P(D), \#, \leq)$, $p_1\#p_2 \Leftrightarrow \nexists d \in D : p_1, p_2 \sqsubseteq d$ and $p_1 \leq p_2 \Leftrightarrow p_1 \sqsubseteq p_2$.
(First Representation Theorem in [Bo90])

Lemma 2.4.3: For any unlabelled flow event structure $\mathcal{E}$ the poset $(\mathit{Conf}(\mathcal{E}), \subseteq)$ is a domain. Its complete primes are the configurations $\downarrow_X (e)$ for $X \in \mathit{Conf}(\mathcal{E})$. Conversely if $(D, \sqsubseteq)$ is a domain then $(D, \sqsubseteq)$ is isomorphic to the poset $(\mathit{Conf}(\mathcal{E}), \subseteq)$ of a flow event structure $\mathcal{E}$.
(Second Representation Theorem in [Bo90])

Definition 2.4.4: (labelled domain)
We call $(D, \sqsubseteq, l)$ a *labelled domain* (labelled over *Act*) iff $(D, \sqsubseteq)$ is a domain and $l : P(D) \rightarrow \mathit{Act}$.
Two labelled domains $(D, \sqsubseteq, l), (D', \sqsubseteq', l')$ are called *isomorphic* if exists $f : D \rightarrow D'$, f bijective, $\forall d_1, d_2 \in D : d_1 \sqsubseteq d_2 \Leftrightarrow f(d_1) \sqsubseteq f(d_2)$, $\forall d \in P(D) : l(d) = l'(f(d))$.

We can transfer lemma 2.4.2 and 2.4.3 to the labelled case:

Lemma 2.4.5: Let $\mathcal{E}$ be a prime event structure labelled over *Act*. Then $(\mathit{Conf}(\mathcal{E}), \subseteq, l')$ with $l'(X) = \begin{cases} l(e) & \text{if } X = \downarrow e \\ \text{undef.} & \text{otherwise} \end{cases}$ is a labelled domain.
If $(D, \sqsubseteq, l)$ is a labelled domain, then $\mathcal{K}(D, \sqsubseteq, l) = (P(D), \#, \leq, l)$ is a labelled prime event structure with $(\mathit{Conf}(\mathcal{K}(D, \sqsubseteq, l), \subseteq, l')$ is isomorphic to $(D, \sqsubseteq, l)$.

Lemma 2.4.6: Let $\mathcal{E}$ be a labelled flow event structure. Then $(\mathit{Conf}(\mathcal{E}), \subseteq, l')$ with $l'(X) = \begin{cases} l(e) & \text{if } X = \downarrow_X e \\ \uparrow & \text{otherwise} \end{cases}$ is a labelled domain. For any labelled domain $(D, \sqsubseteq, l)$ there exists a labelled flow event structure whose set of configurations is isomorphic to $(D, \sqsubseteq, l)$.

2.5 Configuration Structures

Definition 2.5.1: (configuration structure)
Let E be a set of events, $\mathcal{C} \subseteq \{X \subseteq E \mid X \text{ finite}\}$, $\surd \subseteq \mathcal{C}$. $(\mathcal{C}, \surd)$ is called a *(stable) configuration structure* iff

(i) $\emptyset \in \mathcal{C}$

(ii) $\forall X, Y, Z \in \mathcal{C} : X \cup Y \subseteq Z \Rightarrow X \cup Y \in \mathcal{C}$

(iii) $\forall X \in \mathcal{C}\ \forall x \neq x' \in X : \exists Y \subseteq X : (x \in Y \Leftrightarrow x' \notin Y)$

(iv) $\forall X, Y \in \mathcal{C} : X \cup Y \in \mathcal{C} \Rightarrow X \cap Y \in \mathcal{C}$ (Stability)

(v) $\forall X \in \surd : \forall Y \in \mathcal{C} : X \not\subset Y$

Let $E_{\mathcal{C}} := \bigcup_{X \in \mathcal{C}}$, $l : E_{\mathcal{C}} \rightarrow \mathit{Act}$. Then $(\mathcal{C}, \surd, l)$ is called a *(labelled) configuration structure.* (see [Co95])

The class of all configuration structures is denoted by $\mathbf{K}$.

Definition 2.5.2 Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure. Let $E' := \{e \in E \mid \exists X \in \mathit{Conf}(\mathcal{E}) : e \in X\}$. Define $\mathcal{C}(\mathcal{E}) := (\mathit{Conf}_f(\mathcal{E}), \surd, l')$ with $\surd := \{X \in \mathit{Conf}_f(\mathcal{E}) \mid X \text{ complete}\}$ and $l' := l\lceil E'$.

Lemma 2.5.3: Let $\mathcal{E}$ be a flow event structure. Then $\mathcal{C}(\mathcal{E})$ is a stable configuration structure.

Proof: proposition 2.25 in [Co95]

Remark 2.5.4: There exist stable configuration structures that cannot be created by flow event structures, but we will not consider such structures.

In the remainder of this paper we will implicitly assume all configuration structures to be stable, except if stated otherwise.

2.6 Equivalence relations

Let $\mathcal{E}_1 = (E_1, \prec_1, \#_1, l_1), \mathcal{E}_2 = (E_2, \prec_2, \#_2, l_2)$ be two labelled flow event structures and let $E_{C_i} = \{e \in E_i \mid \exists X \in Conf(\mathcal{E}_i) : e \in X\}$. E_{C_i} contains only the events that occur in some configuration. For example self-conflicting events of E_i are not contained in E_{C_i}.

Definition 2.6.1: (event structure isomorphism $\cong_e$)
$\mathcal{E}_1 \cong_e \mathcal{E}_2 :\Leftrightarrow \exists f : E_1 \to E_2$, f bijective with $\forall e, e' \in E_1$:
$e \prec_1 e' \Leftrightarrow f(e) \prec_2 f(e')$
$e \#_1 e' \Leftrightarrow f(e) \#_2 f(e')$
$l_1(e) = l_2(f(e))$

Definition 2.6.2: (domain isomorphism $\cong_d$)
$\mathcal{E}_1 \cong_d \mathcal{E}_2 :\Leftrightarrow \exists h : Conf(\mathcal{E}_1) \to Conf(\mathcal{E}_2)$ with
h bijective
$\forall X, X' \in Conf(\mathcal{E}_1) : X \subseteq X' \Leftrightarrow h(X) \subseteq h(X')$
$\forall X \in Conf(\mathcal{E}_1) : l_1(X) = l_2(h(X))$.

Definition 2.6.3: (configuration structure isomorphism $\cong_c$)
$\mathcal{E}_1 \cong_c \mathcal{E}_2 :\Leftrightarrow \exists f : E_{C_1} \to E_{C_2}$, f bijective and
$\forall X \subseteq E_1 : X \in Conf(\mathcal{E}_1) \Leftrightarrow f(X) \in Conf(\mathcal{E}_2)$.
$\forall X \in Conf(\mathcal{E}_1) : X$ complete $\Leftrightarrow f(X)$ complete
$\forall E \in E_{C_1} : l_1(e) = l_2(f(e))$.

Lemma 2.6.4: For all flow event structures $\mathcal{E}_1, \mathcal{E}_2$ we have:
$\mathcal{E}_1 \cong_e \mathcal{E}_2 \Rightarrow \mathcal{E}_1 \cong_c \mathcal{E}_2 \Rightarrow \mathcal{E}_1 \cong_d \mathcal{E}_2$.

Proof:
$\mathcal{E}_1 \cong_e \mathcal{E}_2 \Rightarrow \mathcal{E}_1 \cong_c \mathcal{E}_2$ is obvious. Let now $\mathcal{E}_1 \cong_c \mathcal{E}_2$, i.e. $\exists f : E_{C_1} \to E_{C_2}$ such that f is bijective and $\forall X \subseteq E_1 : X \in Conf(\mathcal{E}_1) \Leftrightarrow f(X) \in Conf(\mathcal{E}_2)$. Obviously f extended to sets: $f : Conf(\mathcal{E}_1) \to Conf(\mathcal{E}_2)$ is also bijective and $\forall X, X' \in Conf(\mathcal{E}_1) : X \subseteq X' \Leftrightarrow f(X) \subseteq f(X')$.
■

Definition 2.6.5: (interleaving trace equivalence $\approx_{it}$)
Let $\mathcal{E}$ be a flow event structure and $X, Y \in Conf(\mathcal{E})$. We define $X \to^a Y$ iff $\exists e \in E \setminus X$ such that $l(e) = a$ and $Y = X \cup \{e\}$.

A sequence of actions $t = \langle a_1, a_2 ... \rangle$ in *Act* is called *trace* of $\mathcal{E}_i$, if there exist configurations $X_0, ..., X_n \in Conf(\mathcal{E})$ with $X_0 = \emptyset$ and $\forall 0 \le i \le n \Leftrightarrow 1$: $X_i \rightarrow^{a_i} X_{i+1}$.
$Traces(\mathcal{E})$ denotes the set of all traces that can be constructed from $Conf(\mathcal{E})$.
$Traces(\mathcal{E}) := \{t \mid t \text{ trace of } \mathcal{E}\}$.

Two flow event structures are called *interleaving trace equivalent* iff their sets of traces coincide: $\mathcal{E}_1 \approx_{it} \mathcal{E}_2$ iff $Traces(\mathcal{E}_1) = Traces(\mathcal{E}_2)$.
(see [GG90])

Remark 2.6.6: If $\mathcal{E}_1$ and $\mathcal{E}_2$ are prime event structures then $\mathcal{E}_1 \cong_e \mathcal{E}_2 \Leftrightarrow \mathcal{E}_1 \cong_d \mathcal{E}_2$ (see lemma 2.4.2 and 2.4.5).

Lemma 2.6.7: For each flow event structure $\mathcal{E} = (E, \prec, \#, l)$ there exists a domain equivalent prime event structure $\mathcal{P}(\mathcal{E}) := (E', \le', \#', l')$ with
$E' := \{\downarrow_X (e) \mid e \in X \in Conf(\mathcal{E})\}$,
$X \le' X' :\Leftrightarrow X \subseteq X'$,
$X \#' X' :\Leftrightarrow X \cup X' \notin Conf(\mathcal{E})$,
$l'(\downarrow_X (e)) := l(e)$.
I.e. for all flow event structures $\mathcal{E}$: $\mathcal{P}(\mathcal{E}) \cong_d \mathcal{E}$.
(see lemma 2.4.5 and 2.4.6)

2.7 Other definitions

Definition 2.7.1: Let E_1, E_2 be sets of events such that $* \notin E_1 \cup E_2$. Then
$E_1 \times^* E_2 := \{(e_1, e_2) \mid (e_1 \in E_1 \;\&\; e_2 \in E_2) \vee (e_1 \in E_1 \;\&\; e_2 = *) \vee (e_1 = * \;\&\; e_2 \in E_2)\}$.

Let $X \subseteq E_1 \times^* E_2$. Then $\Pi_1(X) := \{e_1 \in E_1 \mid \exists e_2 \in E_2 \cup \{*\} : (e_1, e_2) \in X\}$ and
$\Pi_2(X) := \{e_2 \in E_2 \mid \exists e_1 \in E_1 \cup \{*\} : (e_1, e_2) \in X\}$.

Definition 2.7.2 Let E_1, E_2 be sets of events with $* \notin E_1 \cup E_2$, $l_1 : E_1 \rightarrow Act$, $l_2 : E_2 \rightarrow Act$ and $A \subseteq Act$.

Then $E_1 \times_A^* E_2 := \{(e_1, *) \mid e_1 \in E_1\} \cup \{(*, e_2) \mid e_2 \in E_2\} \cup \{(e_1, e_2) \mid e_1 \in E_1, e_2 \in E_2, l_1(e_1) = l_2(e_2) \in A\}$.

and $E_1 \times_A E_2 := \{(e_1, *) \mid e_1 \in E_1, l_1(e_1) \notin A\} \cup \{(*, e_2) \mid e_2 \in E_2, l_2(e_2) \notin A\}$
$\cup \{(e_1, e_2) \mid e_1 \in E_1, e_2 \in E_2, l_1(e_1) = l_2(e_2) \in A\}$.

3 Semantic Refinement

In this section two denotational semantics for the language **L** of definition 2.1.1 are given – one for flow event structures (in the first subsection) and one for configuration structures (in the last subsection). In the second subsection a refinement-operator on prime event structures is defined.

3.1 Semantic of L on Flow Event Structures

We define a denotational semantics for the language **L** like in [GGR92]:

First of all the operators on flow event structures are defined:

Definition 3.1.1: Let $\mathcal{E}_1 = (E_1, \prec_1, \#_1, l_1)$, $\mathcal{E}_2 = (E_2, \prec_2, \#_2, l_2)$ be flow event structures, $A \subseteq Act$, $a \in Act$, and $E_1 \cap E_2 = \emptyset$, $* \notin E_1 \cup E_2$. Define

- $\mathcal{E}_1 + \mathcal{E}_2 := (E_1 \cup E_2, \prec_1 \cup \prec_2, \#_1 \cup \#_2 \cup (E_1 \times E_2) \cup (E_2 \times E_1), l_1 \cup l_2)$.
- $\mathcal{E}_1; \mathcal{E}_2 := (E_1 \cup E_2, \prec_1 \cup \prec_2 \cup (E_1 \times E_2), \#_1 \cup \#_2, l_1 \cup l_2)$.
- $\mathcal{E}_1 \parallel_A \mathcal{E}_2 := (E, \prec, \#, l)$

$$\begin{array}{rl} E := & E_1 \times_A^* E_2 \\ \prec := & \{((e_1,e_2),(e_1',e_2')) \mid e_1 \prec_1 e_1' \vee e_2 \prec_2 e_2'\} \\ \# := & \{((e_1,e_2),(e_1',e_2')) \mid e_1 \#_1 e_1' \vee e_2 \#_2 e_2' \\ & \vee (e_1 = e_1' \neq * \;\&\; e_2 \neq e_2') \\ & \vee (e_2 = e_2' \neq * \;\&\; e_1 \neq e_1') \\ & \vee (e_1 = e_1' = * \;\&\; e_2 = e_2' \& l_2(e_2) \in A) \\ & \vee (e_2 = e_2' = * \;\&\; e_1 = e_1' \& l_1(e_1) \in A) \end{array}$$

$$l(e_1,e_2) := \begin{cases} l_1(e_1) & \text{if } e_2 = * \\ l_2(e_2) & \text{otherwise} \end{cases}$$

- $E_2 \neq \emptyset$. $\mathcal{E}_1[a \rightsquigarrow \mathcal{E}_2] := (E, \prec, \#, l)$

$$\begin{array}{rl} E = & \{(e_1,e_2) \in E_1 \times E_2 \mid l_1(e_1) = a\} \cup \{(e_1,*) \mid e_1 \in E_1, l_1(e_1) \neq a\} \\ \prec = & \{((e_1,e_2),(e_1',e_2')) \mid e_1 \prec_1 e_1' \vee (e_1 = e_1' \;\&\; e_2 \prec_2 e_2')\} \\ \# = & \{((e_1,e_2),(e_1',e_2')) \mid e_1 \#_1 e_1' \vee (e_1 = e_1' \;\&\; e_2 \#_2 e_2')\} \end{array}$$

$$l(e_1,e_2) = \begin{cases} l_1(e_1) & \text{if } e_2 = * \\ l_2(e_2) & \text{otherwise} \end{cases}$$

The denotational semantics $[\![.]\!]_F$ for the language **L** will now be defined inductively:

Definition 3.1.2: $[\![.]\!]_F : \mathbf{L} \to \mathbf{F}$:

$[\![a]\!]_F := (\{e\}, \emptyset, \emptyset, (e,a))$
$[\![P + Q]\!]_F := [\![P]\!]_F + [\![Q]\!]_F$
$[\![P; Q]\!]_F := [\![P]\!]_F ; [\![Q]\!]_F$
$[\![P \parallel_A Q]\!]_F := [\![P]\!]_F \parallel_A [\![Q]\!]_F$
$[\![P[a \rightsquigarrow Q]\,]\!]_F := [\![P]\!]_F \; [a \rightsquigarrow \; [\![Q]\!]_F]$

Remark 3.1.3 This semantics is almost the same as the one in [Co95] – only the definition of the product differs slightly from the one we use: [Co95] uses: $\mathcal{E}_1 \parallel'_A \mathcal{E}_2 := (E, \prec, \#, l)$ with $E, \#, l$ as before and $(e_1, e_2) \prec (e'_1, e'_2) :\Leftrightarrow (e_1 \prec_1 e'_1 \vee e_2 \prec_2 e'_2)\ \&\ (\neg(e_1, e_2)\#(e'_1, e'_2))$.
But [Co95] showed that this semantics (which we will denote with $[\![.]\!]_{F'}$) yields the same consistency results as $[\![.]\!]_F$ – see also lemma 4.3.1.

Like on flow event structures we now define equivalence relations on terms:

Definition 3.1.4: Let P_1, P_2 be terms of $\mathbf{L}$, then

$P_1 \cong_e P_2 :\Leftrightarrow [\![P_1]\!]_F \cong_e [\![P_2]\!]_F$.

$P_1 \cong_d P_2 :\Leftrightarrow [\![P_1]\!]_F \cong_d [\![P_2]\!]_F$.

$P_1 \cong_c P_2 :\Leftrightarrow [\![P_1]\!]_F \cong_c [\![P_2]\!]_F$.

$P_1 \approx_{it} P_2 :\Leftrightarrow \mathit{Traces}([\![P_1]\!]_F) = \mathit{Traces}([\![P_2]\!]_F)$.

(we also write $\mathit{Traces}(P)$ for $\mathit{Traces}([\![P]\!]_F)$.

Lemma 3.1.5: $\cong_e$ and $\cong_c$ are congruences on $\mathbf{L}$.

Proof: see [GGR92]

3.2 Refinement on Prime Event Structures

Problems with Refinement on Prime Event Structures

We saw that it is rather easy to define refinement on flow event structures: each event to be refined will be replaced by a flow event structure and each event that has been replaced for e inherits the relations to the environment from e.

This kind of refinement is not appropriate for prime event structures because a refined event structure might no longer be a prime event structure.

Example:

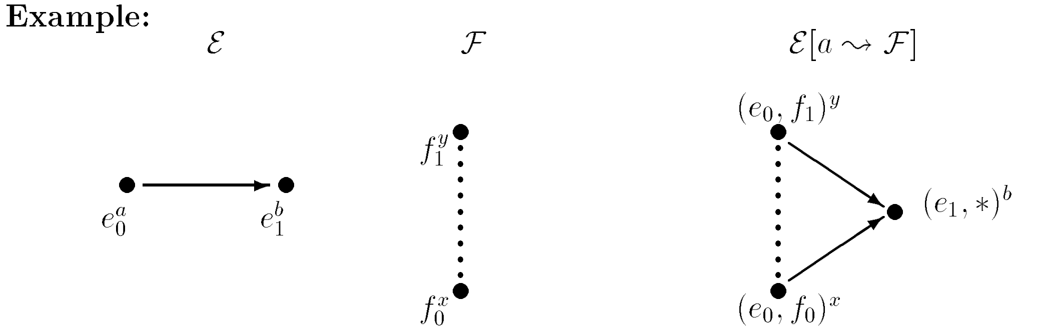

Figure 1: The refinement of a prime event structure does not yield a prime event structure (e^a denotes the event e labelled with a)

But with help of lemma 2.6.7 we know that each flow event structure can be turned into an equivalent prime event structure: Let $\mathcal{E}$ be an arbitrary flow event structure, then $\mathcal{E} \cong_d \mathcal{P}(\mathcal{E})$ which is a prime event structure.

Now if we want to refine a prime event structure we take the flow event structure result and turn it into a prime event structure. This gives rise to the following definition:

Definition 3.2.1 (refinement of prime event structures)
Let $\mathcal{E}_1, \mathcal{E}_2$ be two prime event structures and $a \in Act$. Define $\mathcal{E}_1[a \rightsquigarrow \mathcal{E}_2]_P := \mathcal{P}(\mathcal{E}_1[a \rightsquigarrow \mathcal{E}_2])$ (with $\mathcal{P}$ from lemma 2.6.7).
Thus $\mathcal{E}_1[a \rightsquigarrow \mathcal{E}_2]_P := (E, \leq, \#, l)$ with
$E = \{\downarrow_X (e) \mid e \in X \in Conf(\mathcal{E}_1[a \rightsquigarrow \mathcal{E}_2])\}$,
$\downarrow_X (e) \leq \downarrow_Y (f) :\Leftrightarrow \downarrow_X (e) \subseteq \downarrow_Y (f)$,
$\downarrow_X (e) \# \downarrow_Y (f) :\Leftrightarrow \downarrow_X (e) \cup \downarrow_Y (f) \notin Conf(\mathcal{E}_1[a \rightsquigarrow \mathcal{E}_2])$,
$l(\downarrow_X (e)) = l'(e)$,
with $\mathcal{E}' = (E', \leq', \#', l') = \mathcal{E}_1[a \rightsquigarrow \mathcal{E}_2]$ being the flow event structure constructed by refinement.

3.3 Semantics of L on Configuration Structures

We define a denotational semantics for the language **L** like in [Co95] (with a slightly modified notation).

Definition 3.3.1: Let $(\mathcal{C}_1, \surd_1, l_1)$ and $(\mathcal{C}_2, \surd_2, l_2)$ with $E_1 = \bigcup_{X \in \mathcal{C}_1} X$ and $E_2 = \bigcup_{X \in \mathcal{C}_2} X$ be configuration structures, $A \subseteq Act$, $a \in Act$, $E_1 \cap E_2 = \emptyset, * \notin E_1 \cup E_2$. Define:

- $\mathcal{C}_1 + \mathcal{C}_2 := (C_1 \cup C_2, \surd_1 \cup \surd_2, l_1 \cup l_2)$
- $\mathcal{C}_1 ; \mathcal{C}_2 := (\mathcal{C}, \surd, l)$ such that
  $\mathcal{C} = \mathcal{C}_1 \cup \{X_1 \cup X_2 \mid X_1 \in \mathcal{C}_1, X_2 \in \mathcal{C}_2, X_2 \neq \emptyset \Rightarrow X_1 \in \surd_1\}$,
  $\surd = \{X_1 \cup X_2 \mid X_1 \in \surd_1 \;\&\; X_2 \in \surd_2\}$,
  $l = l_1 \cup l_2$.
- $\mathcal{C}_1 \parallel_A \mathcal{C}_2 := (\mathcal{C}, \surd, l)$ with
  $\mathcal{C}$ is the smallest set with:
  - (i) $\emptyset \in \mathcal{C}$
  - (ii) $\forall (e_1, e_2) \in E_1 \times_A E_2$ and $\forall X \in \mathcal{C}$: $\Pi_i(X \cup \{(e_1, e_2)\}) \in \mathcal{C}_i$ and Π_i injective on $X \cup \{(e_1, e_2)\}$ implies $X \cup \{(e_1, e_2)\} \in \mathcal{C}$.
    Recall that $\Pi_i(e_1, e_2) = \begin{cases} e_i & \text{if } e_i \neq * \\ \text{undefined} & \text{otherwise} \end{cases}$

  $\surd = \{X \in \mathcal{C} \mid \Pi_1(X) \in \surd_1 \;\&\; \Pi_2(X) \in \surd_2\}$

  $l(e_1, e_2) = \begin{cases} l_1(e_1) & \text{if } e_2 = * \\ l_2(e_2) & \text{otherwise} \end{cases}$
- Let $\bigcup_{X \in \mathcal{C}_2} X \neq \emptyset$. $\mathcal{C}_1[a \rightsquigarrow \mathcal{C}_2] := (\mathcal{C}, \surd, l)$ with
  $E := \{(e_1, e_2) \in E_1 \times E_2 \mid l_1(e_1) = a\} \cup \{(e_1, *) \mid e_1 \in E_1, l_1(e_1) \neq a\}$
  For $e_1 \in E_1$ and $X \subseteq E$ define $X(e_1) := \{e_2 \in E_2 \mid (e_1, e_2) \in X\}$.
  $\mathcal{C} = \{X \subseteq E \mid X$ satisfies (i),(ii),(iii)$\}$:
  (i) $\Pi_1(X) \in \mathcal{C}_1$

(ii) $\forall e_1 \in \Pi_1(X) : l_1(e_1) = a \Rightarrow X(e_1) \in \mathcal{C}_2$
(iii) $\forall Y \subseteq \Pi_1(X)$ with $\{e_1 \in \Pi_1(X) \mid X(e_1) \in \surd_2 \vee l_1(e_1) \neq a\} \subseteq Y$: $Y \in \mathcal{C}_1$.

$$\surd = \{X \in \mathcal{C} \mid \Pi_1(X) \in \surd_1 \ \& \ \forall e_1 \in \Pi_1(X) : (l_1(e_1) \neq a \vee X(e_1) \in \surd_2)\}$$

$$l(e_1, e_2) = \begin{cases} l_1(e_1) & \text{if } e_2 = * \\ l_2(e_2) & \text{otherwise} \end{cases}$$

We now inductively define the denotational semantics $[\![.]\!]_K$ for the language $\mathbf{L}$:

Definition 3.3.2: $[\![.]\!]_K : \mathbf{L} \rightarrow \mathbf{K}$:

$[\![a]\!]_K := (\{\emptyset, \{e\}\}, \{\{e\}\}, \{(e, a)\})$
$[\![P + Q]\!]_K := [\![P]\!]_K + [\![Q]\!]_K$
$[\![P; Q]\!]_K := [\![P]\!]_K ; [\![Q]\!]_K$
$[\![P \parallel_A Q]\!]_K := [\![P]\!]_K \parallel_A [\![Q]\!]_K$
$[\![P[a \rightsquigarrow Q]\ ]\!]_K := [\![P]\!]_K\ [a \rightsquigarrow\ [\![Q]\!]_K]$

4 Consistency of flow event structure semantics and configuration structure semantics for L

4.1 Introduction

Two denotational semantics have been defined for the language $\mathbf{L}$ – based on flow event structures respectively on configuration structures. We want to know whether these semantics are consistent, i.e. if $\mathcal{C}([\![P]\!]_F) = [\![P]\!]_K$ holds for an arbitrary term $P \in \mathbf{L}$? (Only finite configurations have to be considered because for all terms $P \in \mathbf{L}$ the equation $Conf([\![P]\!]_F) = Conf_f([\![P]\!]_F)$ holds.)

The consistency can easily be shown for terms that do not contain the parallel operator $\parallel_A$.

In order to show for arbitrary terms P_1, P_2 that $\mathcal{C}([\![P_1 \parallel_A P_2]\!]_F) = [\![P_1 \parallel_A P_2]\!]_K$ one has to show that the set of configurations $Conf([\![P_1 \parallel_A P_2]\!]_F)$ is the same as the one constructed with $[\![.]\!]_K$. With definition 3.3.1 we see that one necessary condition for this is that the projections of configurations of the process $[\![P_1 \parallel_A P_2]\!]_F = [\![P_1]\!]_F \parallel_A [\![P_2]\!]_F$ are configurations of the components $[\![P_1]\!]_F$ and $[\![P_2]\!]_F$. This is a quite natural demand because the possible executions (i.e. the configurations) of a process should not be enlarged by putting another process in parallel.

We thus have to show for arbitrary terms P_1 and P_2 that for all configurations $X \in Conf([\![P_1]\!]_F \parallel_A [\![P_2]\!]_F)$: $\Pi_1(X) \in Conf([\![P_1]\!]_F)$ and $\Pi_2(X) \in Conf([\![P_2]\!]_F)$.
As we see in the following example this condition does not hold for arbitrary flow event structures: We can find flow event structures $\mathcal{E}_1$ and $\mathcal{E}_2$ such that $\Pi_1(Conf(\mathcal{E}_1 \parallel_A \mathcal{E}_2)) \not\subseteq Conf(\mathcal{E}_1)$, i.e. $\exists X \in Conf(\mathcal{E}_1 \parallel_A \mathcal{E}_2)$ such that $\Pi_1(X) \notin Conf(\mathcal{E}_1)$.

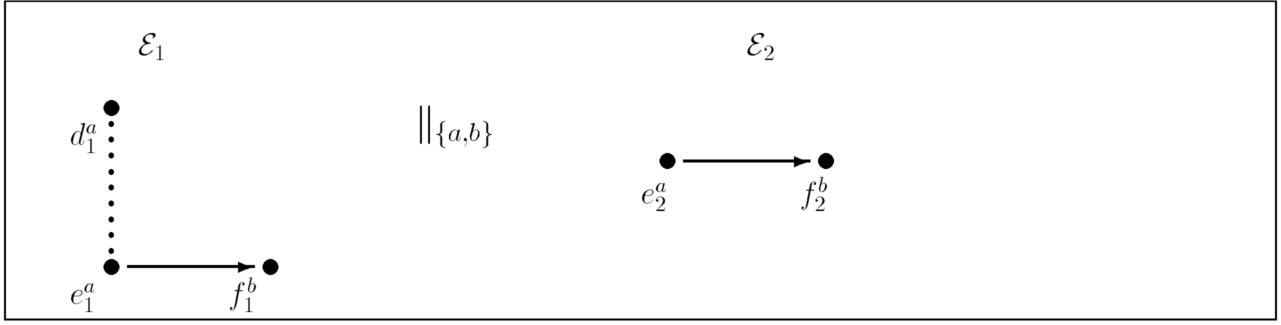

Figure 2: Problem $\mathcal{E}_1$

In our example $X = \{(d_1, e_2), (f_1, f_2)\}$ is a configuration of $\mathcal{E}_1 \|_{\{a,b\}} \mathcal{E}_2$, but $\Pi_1(X) = \{d_1, f_1\} \notin \mathit{Conf}(\mathcal{E}_1)$, because $\Pi_1(X)$ is not left-closed up to conflicts. X is left-closed up to conflicts, because the events $(e_1, *)$ and (e_1, e_2) are both conflicting (d_1, e_2) and (d_1, e_2) is predecessor of (f_1, f_2).

In our example $\mathcal{E}_1$ is the "critical" event structure because for $\mathcal{E}_2$ no event structure $\mathcal{E}_3$ exists with $\Pi_1 \mathit{Conf}(\mathcal{E}_2 \|_A \mathcal{E}_3) \not\subseteq \mathit{Conf}(\mathcal{E}_2)$.

We want to know if critical event structures like $\mathcal{E}_1$ can be created by the language **L**.

Recall that $\mathbf{L^o}$ denotes the set of those terms of our language **L** that do not contain a refinement operator. [Co95] showed that for all terms $P \in \mathbf{L^o}$: $\mathcal{C}([\![P]\!]_F) = [\![P]\!]_K$ and in particular $\forall P_1, P_2 \in \mathbf{L^o}$: $\Pi_i(\mathit{Conf}([\![P_1]\!]_F \|_A [\![P_2]\!]_F)) \subseteq \mathit{Conf}([\![P_i]\!]_F)$ (see lemma 4.3.1).

It was claimed in [Co95] that the consistency result is also valid for all terms of the language **L**. But this proof seems to contain a mistake. We will give a correct proof here.

4.1.1 The Problem of the Proof in [Co95]

Definition 4.1.1: (delta axiom)
Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure. $\mathcal{E}$ *satisfies the delta axiom* iff $\forall d, e, f \in E$: $d \# e \prec f \;\&\; d \not\prec f \Rightarrow \exists g \in E : (e \# g \prec f) \;\&\; (\forall e' \in E \setminus \{e\} : (g \# e') \Rightarrow (e \# e' \;\&\; e' \sim f))$.
Here $e \sim e' :\Leftrightarrow e \# e' \vee e \prec e' \vee e' \prec e$.

In [Co95] the following facts are mentioned:

- All flow event structures $[\![P]\!]_F$ created by terms $P \in \mathbf{L^o}$ satisfy the delta axiom.
- If the flow event structures $\mathcal{E}_1$, $\mathcal{E}_2$ satisfy the delta axiom then $\forall A \subseteq \mathit{Act}$: $\forall i \in \{1, 2\} : \Pi_i(\mathit{Conf}(\mathcal{E}_1 \|_A \mathcal{E}_2)) \subseteq \mathit{Conf}(\mathcal{E}_i)$.

We want to show that the same holds for all terms of the language **L**.

[Co95] claims that the delta axiom is preserved by refinement. We can disprove this however by a counter example:

Let P_1, P_2 be in $\mathbf{L^o}$ with $P_1 = ((a \|_\emptyset b); c) \|_{\{a,b\}} (a + b)$ and $P_2 = u; v$.
Then $[\![P_1[b \rightsquigarrow P_2]\,]\!]_F$ does **not** satisfy the delta axiom.

Consider Figure 3: (self-conflicting events are encircled). If one chooses $d := (e_0, *), e := (e_3, f_0), f := (e_6, *)$, one cannot find g with the property $(e \# g \prec f) \;\&\; (\forall e' \in E \setminus \{e\} : (g \# e') \Rightarrow (e \# e' \;\&\; e' \sim f))$. The only possible candidates for g are (e_5, f_0) and (e_5, f_1), but for these one has $e' := (e_3, f_1) \# g$, $e' \neq e$ but $\neg(e' \# e)$. $(e_2, *)$ is no candidate for g, because $d \# (e_2, *)$ and $d \not\prec f$.

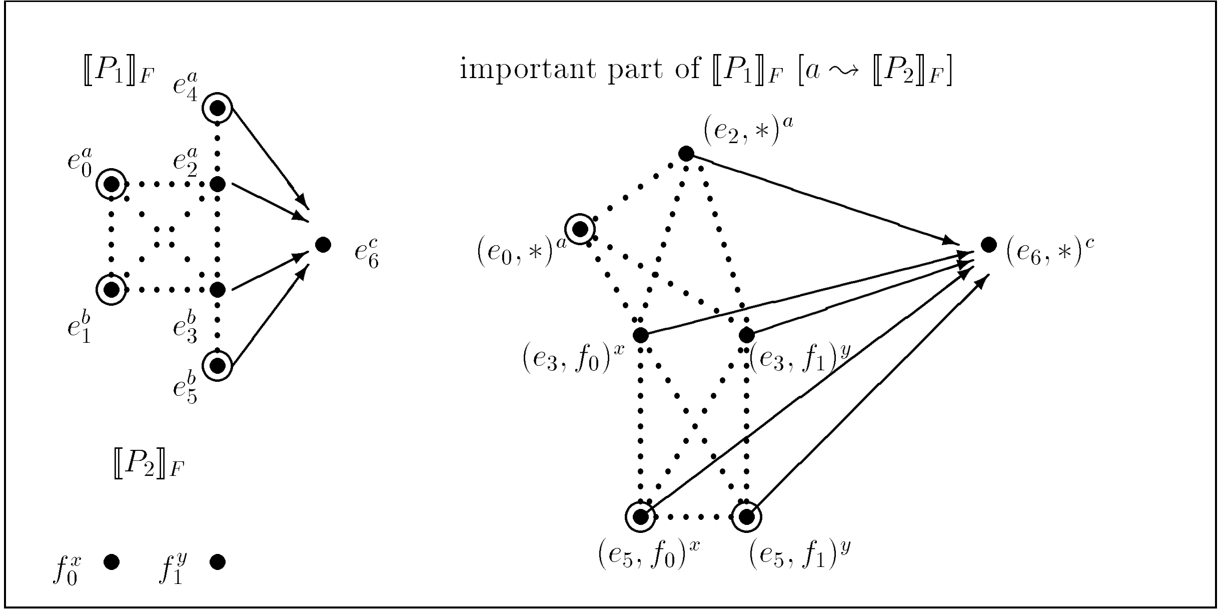

Figure 3: delta axiom is not satisfied for $d = (e_0,*), e = (e_3,f_0), f = (e_6,*)$

4.2 Characterization of "critical" event structures

Definition 4.2.1 A flow event structure $\mathcal{E}_1 = (E_1, \prec_1, \#_1, l_1)$ is called *crictical* if there exists a flow event structure $\mathcal{E}_2 = (E_2, \prec_2, \#_2, l_2)$ and an action set $A \subseteq Act$ such that there exists a configuration $X \in Conf(\mathcal{E}_1 \parallel_A \mathcal{E}_2)$ with $\Pi_1(X) \notin Conf(\mathcal{E}_1)$.

We want to know if such critical flow event structures can be created by the language **L**. Since the definition of critical uses a quantification over all event structures it is not easy to use and we look for a simpler characterization. So how can the critical flow event structures be characterized?

In the example the difficulty arose because a new causal relationship was created between d_1 and f_1 by the events d_2 and f_2. This leads to the following idea:

$\mathcal{E} = (E, \prec, \#, l)$ can only be critical if it is possible to extend the flow relation in such a way that a new configuration arises, i.e. there exists $\prec' \supset \prec$ with $\exists X \in Conf(E, \prec', \#, l)$ but $X \notin Conf(E, \prec, \#, l)$.

We do not consider all possible extensions of $\prec$ but only those ones that add a finite number of predecessors to a single event:

Definition 4.2.2 Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure, $f \in E, D \subseteq E, D$ finite. Define an event structure with an extended flow relation:
$H(\mathcal{E}, D, f) := (E, \prec \cup (D \times f), \#, l)$.

Thus the only difference between $\mathcal{E}$ and $H(\mathcal{E}, D, f)$ is that in $H(\mathcal{E}, D, f)$ all events of D are predecessors of f.

We now want to characterize critical event structures by introducing the notion of "problematic" event structures:

Definition 4.2.3 Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure. $\mathcal{E}$ is called *problematic* iff exists $D \subseteq E$, D finite and $\forall d \in D : d \not\prec f$: such that exists $X \in$*Conf*$(H(\mathcal{E}, D, f)) \setminus$*Conf*$(\mathcal{E})$.

Thus an event structure $\mathcal{E} = (E, \prec, \#, l)$ is problematic if adding a finite number of predecessors to an event f (that all were in no relation to f before) leads to a new configuration.

Lemma 4.2.4 Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure, D a finite subset of E such that $\forall d \in D : d \not\prec f$. Let $Y \in$*Conf*$(H(\mathcal{E}, D, f)) \setminus$*Conf*$(\mathcal{E})$, i.e. $\mathcal{E}$ is problematic. Let $X := \{e \in Y \mid e \leq'_Y f\}$ with $\leq'_Y := (\prec' \cap (Y \times Y))^*)$. Then $X \in$*Conf*$(H(\mathcal{E}, D, f)) \setminus$*Conf*$(\mathcal{E})$ and $X \setminus \{f\} \in$*Conf*$(\mathcal{E})$.

Proof: evident.

Proposition 4.2.5: For all flow event structures $\mathcal{E}$ the following implication holds: If $\mathcal{E}$ is critical then $\mathcal{E}$ is problematic.

Proof: Let A be a subset of *Act*, $X \in$*Conf*$(\mathcal{E}_1 \parallel_A \mathcal{E}_2)$ and $\Pi_1(X) \notin$*Conf*$(\mathcal{E}_1)$. Then there exists $e_1 \notin \Pi_1(X), (f_1, f_2) \in (X), e_1 \prec_1 f_1$ and $\forall (d_1, d_2) \in X : \neg(d_1 \#_1 e_1) \vee d_1 \not\prec_1 f_1$.

Let $z \in Z \subseteq E_1$. Define $\mathcal{M}(Z, z)$ iff $\exists z' \notin Z, z' \prec_1 z$ & $\forall z'' \in Z : \neg(z'' \# z') \vee \neg(z'' \prec_1 z)$. For example $\mathcal{M}(\Pi_1(X), f_1)$ holds.

Consider $\downarrow_X (f_1, f_2) = \{(d_1, d_2) \in X \mid (d_1, d_2) \leq_X (f_1, f_2)\}$. Since X is a configuration, it follows that (f_1, f_2) only has a finite number of predecessors in X, so $\downarrow_X (f_1, f_2)$ is finite. Because of this there exist minimal elements in the set $\{(y_1, y_2) \in \downarrow_X (f_1, f_2) \mid \mathcal{M}(\Pi_1(\downarrow_X (f_1, f_2)), y_1)\}$. Let $(f'_1, f'_2) \in \downarrow_X (f_1, f_2)$ be such a minimal element, then $\mathcal{M}(\Pi_1(\downarrow_X (f_1, f_2)), f'_1)$ and $\forall (x_1, x_2) \leq_X (f'_1, f'_2)$ & $(x_1, x_2) \neq (f'_1, f'_2) : \neg \mathcal{M}(\Pi_1(\downarrow_X (f_1, f_2)), x_1)$.

Since $\mathcal{M}(\Pi_1(\downarrow_X (f_1, f_2)), f'_1)$ holds we can conclude that $D := \{d'_1 \in \Pi_1(\downarrow_X (f'_1, f'_2)) \mid d'_1 \not\prec f'_1$ & $\exists x_1 \notin \Pi_1(X), x_1 \prec_1 f'_1, d'_1 \# x_1\} \neq \emptyset$. $D \subseteq \downarrow_X (f'_1, f'_2)$ shows that D is finite.

Now let us look at $H(\mathcal{E}_1, D, f'_1)$ with $\prec'_1 := \prec_1 \cup D \times \{f'_1\}$ and $Y := \downarrow_{\Pi_1(X)} (f'_1) \cup \bigcup_{d'_1 \in D} \downarrow_{\Pi_1(X)} (d'_1) = \{x_1 \in \Pi_1(X) \mid x_1 \leq_{\Pi_1(X)} f'_1 \vee \exists d'_1 \in D : x_1 \leq_{\Pi_1(X)} d'_1\}$. Then $Y \subseteq \Pi_1(\downarrow_X (f'_1, f'_2))$ holds.

We will show, that $Y \notin$*Conf*$(\mathcal{E}_1)$ but $Y \in$*Conf*$(H(\mathcal{E}_1, D, f'_1))$ and therefore conclude that $\mathcal{E}_1$ is problematic.

- $Y \notin$*Conf*$(\mathcal{E}_1)$ because it is not left-closed up to conflicts: $e'_1 \prec_1 f'_1 \in Y$ & $\forall h'_1 \in Y : \neg(h'_1 \# e'_1) \vee \neg(h'_1 \prec_1 f'_1)$.
- $Y \in$*Conf*$(H(\mathcal{E}_1, D, f'_1))$:

  (i) Y is conflict-free since $Y \subset \Pi_1(X)$

  (ii) $\leq_Y$ is a partial order with respect to $\prec_1$ and with respect to $\prec'_1$ because $Y \subset \Pi_1(X)$ and $\forall (d'_1, f'_1) \in \prec'_1 \setminus \prec_1$: $\forall d'_2 : (d'_1, d'_2) \in X : (d'_1, d'_2) \leq_X (f'_1, f'_2)$

  (iii) evident

  (iv) Y is left-closed up to conflicts:

Let $z_1 \in Y, x_1 \notin Y, x_1 \prec_1 z_1$.
Case 1: $z_1 = f_1', x_1 \notin \Pi_1(X)$. According to the definition of D there exists $d_1' \in Y$ with $x_1 \#_1 d_1' \prec_1' f_1'$.
Case 2: $z_1 = f_1', x_1 \in \Pi_1(X) \Rightarrow x_1 \in Y$, since $x_1 \prec_1 f_1'$.
Case 3: $z_1 \neq f_1'$. $\forall z_1 \in Y : z_1 \in \Pi_1(\downarrow_X (f_1', f_2'))$, and therefore if $z_1 \neq f_1'$ then $\neg\mathcal{M}(\Pi_1(\downarrow_X (f_1, f_2)), z_1)$, so that $\forall x_1 \notin \Pi_1(\downarrow_X (f_1, f_2)), x_1 \prec_1 z_1 : \exists y_1 \in\downarrow_X (f_1, f_2)$ with $x_1 \#_1 y_1 \prec_1 z_1$ and therefore $y_1 \leq_Y f_1'$, and $y_1 \in Y$. If $x_1 \prec_1 z_1$ and $x_1 \in \Pi_1(\downarrow_X (f_1, f_2))$, then $x_1 \in Y$.
This shows that Y is left-closed up to conflicts with respect to $\prec_1'$.

We conclude that $Y \in Conf(H(\mathcal{E}_1, D, f_1'))$.

■

Corollary 4.2.6: Let $\mathcal{E}_1$, $\mathcal{E}_2$ be flow event structures with $\mathcal{E}_1$ and $\mathcal{E}_2$ not being problematic. Then $\forall i \in \{1,2\} : \Pi_i(Conf(\mathcal{E}_1 \parallel_A \mathcal{E}_2)) \subseteq Conf(\mathcal{E}_i)$.

Remark 4.2.7: The following (simpler) definition of problematic is not sufficient to ensure proposition 4.2.5: If we called $\mathcal{E}$ problematic iff $\exists d, f \in E, d \not\prec f$, so that $\exists X \in Conf(\mathcal{E}')\setminus Conf(\mathcal{E})$ with $\mathcal{E}' = (E, \prec \cup\{(d,f)\}, \#, l)$ the implication "$\mathcal{E}$ is critical $\Rightarrow$ $\mathcal{E}$ is problematic" would not hold, as we see in the following example:

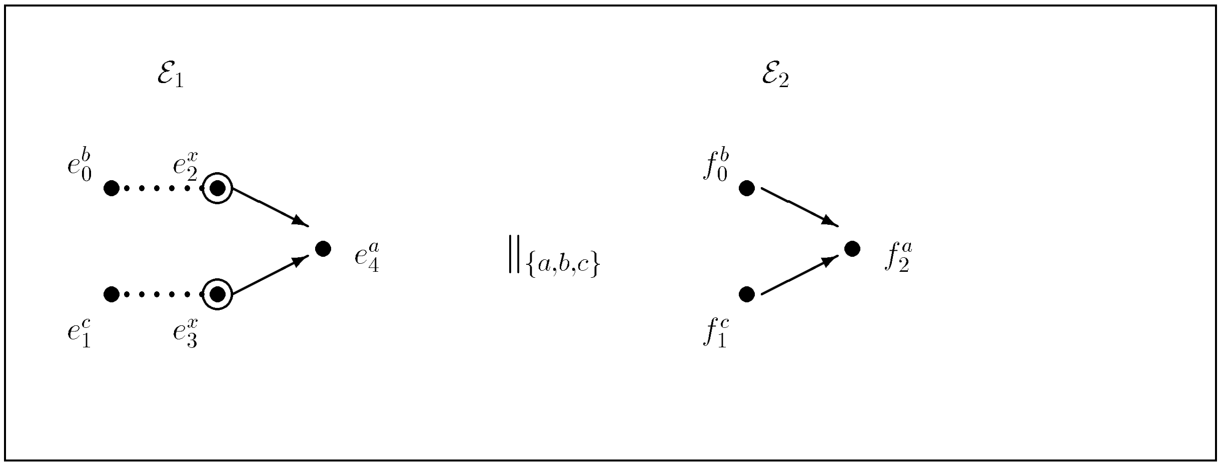

Figure 4: $\mathcal{E}_1$ would not be problematic with this definition but nevertheless critical

We see in Figure 4 that $\mathcal{E}_1$, not problematic with the modified definition, is critical because $X = \{(e_4, f_2), (e_0, f_0), (e_1, f_1)\} \in Conf\,(\mathcal{E}_1 \parallel_{\{a,b,c\}} \mathcal{E}_2)$, and $\Pi_1(X) = \{(e_4, e_0, e_1)\} \notin Conf(\mathcal{E}_1)$.

Proposition 4.2.8: For all terms $P \in \mathbf{L}$: $[\![P]\!]_F$ is not problematic.

Proof: (induction on the syntactical structure of the terms)
Let P_i be Terms in $\mathbf{L}$ and $[\![P_i]\!]_F := \mathcal{E}_i = (E_i, \prec_i, \#_i, l_i)$, $[\![P]\!]_F = \mathcal{E} := (E, \prec, \#, l)$ and $E_1 \cap E_2 = \emptyset$.

- $P = a$, obvious

- $P = P_1; P_2$
  Suppose that $\mathcal{E}$ is problematic, then
  $\exists d^1, ..., d^n : d^i \not\prec f \in E \Rightarrow \{d^1, .., d^n, f\} \subseteq E_1 \vee \{d^1, .., d^n, f\} \subseteq E_2$ and therefore
  $\exists X \in \mathit{Conf}(H(\mathcal{E}_1, \{d^1, .., d^n\}, f)) \setminus \mathit{Conf}(\mathcal{E}_1)$ or $\exists X \in \mathit{Conf}(H(\mathcal{E}_2, \{d^1, .., d^n\}, f)) \setminus$
  $\mathit{Conf}(\mathcal{E}_2)$. Contradiction!

- $P = P_1 + P_2$
  Suppose that $\mathcal{E}$ is problematic, then
  $\exists d^1, ..., d^n : d^i \not\prec f \in E \Rightarrow \{d^1, .., d^n, f\} \subseteq E_1 \vee \{d^1, .., d^n, f\} \subseteq E_2$ and therefore
  $\exists X \in \mathit{Conf}(H(\mathcal{E}_1, \{d^1, .., d^n\}, f)) \setminus \mathit{Conf}(\mathcal{E}_1)$ or $\exists X \in \mathit{Conf}(H(\mathcal{E}_2, \{d^1, .., d^n\}, f)) \setminus$
  $\mathit{Conf}(\mathcal{E}_2)$. Contradiction!

- $P = P_1 \parallel_A P_2$
  By induction hypothesis $\mathcal{E}_1$ and $\mathcal{E}_2$ are not problematic, and it follows
  $\forall X \in \mathit{Conf}(\mathcal{E}_1 \parallel_A \mathcal{E}_2) : \forall i \in \{1, 2\} : \Pi_i(X) \in \mathit{Conf}(\mathcal{E}_i)$.
  Suppose that $\mathcal{E}$ is problematic, then $\exists D := \{d^1 = (d^1_1, d^1_2), ..., d^n = (d^n_1, d^n_2)\}$ and
  $f = (f_1, f_2)$ with $\forall 1 \leq i \leq n : d^i \not\prec f = (f_1, f_2) \in E$ and $\exists Y \in \mathit{Conf}(H(\mathcal{E}, D, f)) \setminus$
  $\mathit{Conf}(\mathcal{E})$. We denote $\prec' := \prec \cup D \times \{f\}$. Recall lemma 4.2.4 and consider
  $X = \{h \in Y \mid h \leq'_Y f\} = \{h \in Y \mid h \leq_Y f \vee \exists d^i \in D \cap Y : h \leq_Y d^i\}$. Then
  $X \in \mathit{Conf}(H(\mathcal{E}, D, f)), X \notin \mathit{Conf}(\mathcal{E})$, $X \setminus \{f\} \in \mathit{Conf}(\mathcal{E})$, and it follows for $i \in \{1, 2\}$
  that $\Pi_i(X \setminus \{f\}) \in \mathit{Conf}(\mathcal{E}_i)$, with $\Pi_i(X \setminus \{f\}) = \begin{cases} \Pi_i(X) \setminus \{f_i\} & \text{if } f_i \neq * \\ \Pi_i(X) & \text{otherwise.} \end{cases}$

  Since $X \notin \mathit{Conf}(\mathcal{E})$, but $X \setminus \{f\} \in \mathit{Conf}(\mathcal{E})$, there exists $e = (e_1, e_2) \notin X$ and
  $d^i \in D \cap X$ such that: (i) $d^i \# e \prec f$ and (ii) $\forall h \in X : \neg(h \# e) \vee \neg(h \prec f)$.

  - Case 1: $d^i_1 \#_1 e_1 \prec_1 f_1 \Rightarrow \Pi_1(X) \notin \mathit{Conf}(\mathcal{E}_1)$.
    Let $W := \{h_1 \in \Pi_1(X) \mid h_1 \not\prec_1 f_1 \;\&\; \exists x_1 \notin \Pi_1(X) : x_1 \prec_1 f_1, x_1 \# h_1\}$ and
    $D := \{f_1, d^1_1, ..., d^n_1\} \setminus \{*\} \cup W$. Since $W \subseteq \Pi_1(\downarrow_X (f_1, f_2))$, both W and D are finite.
    Since $\Pi_1(X) \setminus \{f_1\} \in \mathit{Conf}(\mathcal{E}_1)$ $\Pi_1(X) \in \mathit{Conf}(H(\mathcal{E}_1, D, f_1))$ holds and consequently
    $\mathcal{E}_1$ is problematic. Contradiction!
  - Case 2: $d^i_2 \#_2 e_2 \prec_2 f_2$ analogous to case 1.
  - Case 3: $d^i_1 \#_1 e_1, e_2 \prec_2 f_2$ (we assume $\neg(e_2 \# d^i_2)$)
    a) $(*, e_2) \in X \Rightarrow (e_1, e_2) \# (*, e_2) \prec (f_1, f_2)$, Contradiction to assumption (ii)!
    b) $(*, e_2) \notin X$. Since $X \in \mathit{Conf}(H(\mathcal{E}, \{d^1, .., d^n\}, f))$, there exists $(h_1, h_2) \in X$ such that $h_2 \#_2 e_2$ and $(h_1, h_2) \prec' (f_1, f_2)$. If $(h_1, h_2) \prec (f_1, f_2)$ this would be a contradiction to assumption (ii). So $\exists j : (h_1, h_2) = (d^j_1, d^j_2)$. Now in analogy with case 2 a contradiction follows.
  - Case 4: $d^i_2 \#_2 e_2, e_1 \prec_1 f_1$ in analogy to case 3.
  - Case 5: $d^i_1 = e_1 \neq * \;\&\; d^i_2 \neq e_2$ (we assume $\neg(e_2 \# d^i_2)$)
    Since $(d^i_1, d^i_2) \not\prec (f_1, f_2)$, $e_1 \prec_1 f_1$ is not allowed, one concludes $e_2 \prec_2 f_2$.
    a) $(*, e_2) \in X \Rightarrow (e_1, e_2) \# (*, e_2) \prec (f_1, f_2)$, Contradiction to assumption (ii)!
    b) $(*, e_2) \notin X$ in analogy to case 3b)
  - Case 6: $d^i_2 = e_2 \neq * \;\&\; d^i_1 \neq e_1$ in analogy to case 5.

- $P = P_1[a \rightsquigarrow P_2]$
  We assume that $\mathcal{E}$ is problematic, consequently there exist
  $D := \{d^1 = (d^1_1, d^1_2), ..., d^n = (d^n_1, d^n_2)\}$ and $f = (f_1, f_2)$ with

$\forall 1 \le i \le n : d^i \not\prec f = (f_1, f_2) \in E$ and $\exists Y \in$ *Conf*$(H(\mathcal{E}, D, f)) \setminus$ *Conf*$(\mathcal{E})$. Define $\prec' := \prec \cup D \times \{f\}$. Recall lemma 4.2.4 and consider $X := \{h \in Y \mid h \le'_Y f\} = \{h \in Y \mid h \le_Y f \vee \exists d^i \in D \cap Y : h \le_Y d^i\}$. Then $X \in$ *Conf*$(H(\mathcal{E}, D, f))$ but $X \notin$ *Conf*$(\mathcal{E})$, $X \setminus \{f\} \in$ *Conf*$(\mathcal{E})$.

Since $X \notin$ *Conf*$(\mathcal{E})$, but $X \setminus \{f\} \in$ *Conf*$(\mathcal{E})$, there exists $e = (e_1, e_2) \notin X$ and i such that: (i) $d^i \# e \prec f$ and (ii) $\forall h \in X : \neg(h \# e) \vee \neg(h \prec f)$.

- Case 1: $d^i_1 \#_1 e_1 \prec_1 f_1$

  Suppose $\Pi_1(X) \in$ *Conf*$(\mathcal{E}_1)$, then there exists $h_1 \in \Pi_1(X)$ with $e_1 \#_1 h_1 \prec_1 f_1$ and because of this there exists $(h_1, h_2) \in X$ with $(e_1, e_2) \# (h_1, h_2) \prec (f_1, f_2)$, Contradiction to assumption (ii), consequently $\Pi_1(X) \notin$ *Conf*$(\mathcal{E}_1)$.

  Suppose $\Pi_1(X) \notin$ *Conf*$(H(\mathcal{E}_1, \{d^1_1, ..., d^n_1\}, f_1))$, i.e. $\exists x_1 \notin \Pi_1(X), x_1 \prec_1 f_1$ & $\forall h_1 \in \Pi_1(X) : \neg(x_1 \#_1 h_1) \vee \neg(h_1 \prec'_1 f_1)$, in particular $\forall 1 \le i \le n : \neg(d^i_1 \#_1 x_1)$.

  If there exist different $(f_1, f_2), (f_1, f'_2) \in X$ take $(f_1, f_2) \in X$ with $\nexists y_2 \in E_2$ and $y_2 \prec_2 f_2$. Since $X \in$ *Conf*$(H(\mathcal{E}, \{d^1, ..., d^n\}, f))$ and $(x_1, x_2) \notin X, (x_1, x_2) \prec (f_1, f_2)$, there exists $(h_1, h_2) \in X$ with $(x_1, x_2) \# (h_1, h_2) \prec' (f_1, f_2)$. $x_1 \#_1 h_1$ because otherwise $x_1 \in \Pi_1(X)$.
  If $(h_1, h_2) \prec (f_1, f_2)$ then $h_1 \prec_1 f_1$, since f_2 has no predecessors. So $x_1 \#_1 h_1 \prec_1 f_1$. Contradiction!
  If $(h_1, h_2) \prec' (f_1, f_2)$, then $\exists j : (h_1, h_2) = (d^j_1, d^j_2)$ but then $d^j_1 \#_1 x_1$, Contradiction!

  Therefore $\Pi_1(X) \in$ *Conf*$(H(\mathcal{E}_1, \{d^1_1, ..., d^n_1\}, f_1))$, $\Pi_1(X) \notin$ *Conf*$(\mathcal{E}_1)$, and therefore $\mathcal{E}_1$ is problematic. Contradiction!

- Case 2: $d^i_1 = e_1 = f_1$ & $d^i_2 \#_2 e_2 \prec_2 f_2$

  Look at $X(e_1) = \{y_2 \in E_2 \mid (e_1, y_2) \in X\}$. $X(e_1) \notin$ *Conf*$(\mathcal{E}_2)$, since with assumption (ii) $\forall h_2 \in X(e_1) : \neg(e_2 \# h_2) \vee \neg(h_2 \prec_2 f_2)$.

  Let $D := \{d^1_2, ..., d^n_2\} \setminus \{*\}$. Suppose $X \notin$ *Conf*$(H(\mathcal{E}_2, D, f_2))$. Then $\exists x_2 \notin X(e_1)$ with $x_2 \prec_2 f_2$ and $\forall h_2 \in X(e_1) : \neg(h_2 \#_2 x_2) \vee \neg(h_2 \prec'_2 f_2)$, in particular $\forall d^i_2 : \neg(h_2 \#_2 d^i_2)$. But $(e_1, x_2) \notin X$ and $(e_1, x_2) \prec (e_1, f_2)$, consequently $\exists (h_1, h_2) \in X$ with $(e_1, x_2) \# (h_1, h_2) \prec' (e_1, f_2)$. If $h_1 \#_1 e_1$ then $(h_1, h_2) \notin X$, consequently $h_1 = e_1$ & $h_2 \#_2 x_2$ and $h_2 \notin D$, consequently $(e_1, h_2) \prec (e_1, f_2)$ and $h_2 \prec_2 f_2$, Contradiction!

  We conclude $X \in$ *Conf*$(H(\mathcal{E}_2, D, f_2))$ and $\mathcal{E}_2$ is problematic. Contradiction!

There are no more cases for $P_1[a \rightsquigarrow P_2]$: If $d^i_1 \#_1 e_1$ & $e_1 = f_1$ & $e_2 \prec_2 f_2$ then $d^i \# f$; if $d^i_1 = e_1$ & $d_2 \# e_2$ & $e_1 \prec_1 f_1$ then $d^i \prec f$.

■

Corollary 4.2.9: For all $P_1, P_2 \in \mathbf{L}$ and for all $A \subseteq$ *Act* the following holds: $\Pi_1($*Conf*$(\llbracket P_1 \rrbracket_F \parallel_A \llbracket P_2 \rrbracket_F)) \subseteq$ *Conf*$(\llbracket P_1 \rrbracket_F)$ and $\Pi_2($*Conf*$(\llbracket P_1 \rrbracket_F \parallel_A \llbracket P_2 \rrbracket_F)) \subseteq$ *Conf*$(\llbracket P_2 \rrbracket_F)$.

4.3 Consistency Results

Lemma 4.3.1 For all terms $P_1, P_2 \in \mathbf{L}$ and for all actions $a \in Act$:
$[\![a]\!]_K = \mathcal{C}([\![a]\!]_{F'}) = \mathcal{C}([\![a]\!]_F)$
If $[\![P_i]\!]_K = \mathcal{C}([\![P_i]\!]_{F'}) = \mathcal{C}([\![P_i]\!]_F)$ then:
$[\![P_1]\!]_K ; [\![P_2]\!]_K = \mathcal{C}([\![P_1]\!]_{F'} ; [\![P_2]\!]_{F'}) = \mathcal{C}([\![P_1]\!]_F ; [\![P_2]\!]_F)$
$[\![P_1]\!]_K + [\![P_2]\!]_K = \mathcal{C}([\![P_1]\!]_{F'} + [\![P_2]\!]_{F'}) = \mathcal{C}([\![P_1]\!]_F + [\![P_2]\!]_F)$
$[\![P_1]\!]_K [a \rightsquigarrow [\![P_2]\!]_K] = \mathcal{C}([\![P_1]\!]_{F'} [a \rightsquigarrow [\![P_2]\!]_{F'}]) = \mathcal{C}([\![P_1]\!]_F [a \rightsquigarrow [\![P_2]\!]_F])$
If P_1 and P_2 satisfy the delta axiom then
$[\![P_1]\!]_K \parallel_A [\![P_2]\!]_K = \mathcal{C}([\![P_1]\!]_{F'} \parallel'_A [\![P_2]\!]_{F'}) = \mathcal{C}([\![P_1]\!]_F \parallel_A [\![P_2]\!]_F)$.

Proof: [Co95], proposition 3.43 shows this result for $[\![.]\!]_F$ and $[\![.]\!]_{F'}$

The last point has to be shown for terms that do not satisfy the delta axiom:

Lemma 4.3.2: Let $P_1, P_2 \in \mathbf{L}$, $A \subseteq Act$. If $[\![P_i]\!]_K = \mathcal{C}([\![P_i]\!]_F)$ then
$[\![P_1 \parallel_A P_2]\!]_K = \mathcal{C}([\![P_1]\!]_F \parallel_A [\![P_2]\!]_F)$.

Proof:
Let $(\mathcal{C}, \surd, l) := [\![P_1 \parallel_A P_2]\!]_K = [\![P_1]\!]_K \parallel_A [\![P_2]\!]_K$ (note that we use the same symbol $\parallel_A$ for the domain of flow event structures and the domain of configuration structures).
Let $(\mathcal{C}', \surd', l') := \mathcal{C}([\![P_1]\!]_F \parallel_A [\![P_2]\!]_F)$ and let $\mathcal{E} := (E, \prec, \#, l) = [\![P_1]\!]_F \parallel_A [\![P_2]\!]_F$.
Let $\mathcal{E}_i := (E_i, \prec_i, \#_i, l_i) = [\![P_i]\!]_F$ and $(\mathcal{C}_i, \surd_i, l_i) = [\![P_i]\!]_K$.
According to the assumption $(\mathcal{C}_i, \surd_i, l_i) = \mathcal{C}(\mathcal{E}_i)$.

We want to show that $(\mathcal{C}, \surd, l) = (\mathcal{C}', \surd', l')$.

- $\mathcal{C} \subseteq \mathcal{C}'$

  Since we only take finite configurations into account an induction over the number of events in a configuration is possible. Let $X \in \mathcal{C}$.
  $X = \emptyset \Rightarrow X \in \mathcal{C}'$.
  Let $X = X' \cup \{(e_1, e_2)\}$ and $X' \in \mathcal{C}'$, $(e_1, e_2) \notin X'$. According to the assumption we have $(e_1, e_2) \in E_1 \times_A E_2$ and $\forall i \in \{1, 2\}$: $\Pi_i(X) \in \mathcal{C}_i (= Conf([\![P_i]\!]_F))$ and Π_i is injective on X.
  Suppose $X \notin \mathcal{C}'$
  (i) Suppose X contains conflicts: (e_1, e_2) cannot be self-conflicting therefore
  $\exists (e'_1, e'_2) \in X' : (e'_1, e'_2) \# (e_1, e_2)$.
  Case 1: $e'_1 \#_1 e_1 \Rightarrow \Pi_1(X) \notin Conf([\![P_1]\!]_F)$. Contradiction!
  Case 2: $e'_2 \#_2 e_2 \Rightarrow \Pi_2(X) \notin Conf([\![P_2]\!]_F)$. Contradiction!
  Case 3: $e_1 = e'_1 \neq * \;\&\; e_2 \neq e'_2 \Rightarrow \Pi_1$ is not injective on X. Contradiction!
  Case 4: $e_2 = e'_2 \neq * \;\&\; e_1 \neq e'_1 \Rightarrow \Pi_2$ is not injective on X. Contradiction!
  (ii) Suppose $\leq_X$ isn't a partial order, i.e. X contains cycles with respect to $\prec$. Then $\exists (e'_1, e'_2) \in X'$ with $(e_1, e_2) \prec (e'_1, e'_2)$. Since $X' \in Conf([\![P_1]\!]_F \parallel_A [\![P_2]\!]_F)$, this is only possible if $\exists (e''_1, e''_2) \in X'$ with $(e_1, e_2) \# (e''_1, e''_2) \prec (e'_1, e'_2)$, but then X contains conflicts. Contradiction to (i).
  (iii) All events of X obviously only have finitely many predecessors, since we only take into account finite configurations.
  (iv) Suppose X is not left-closed up to conflicts.
  Let $(e'_1, e'_2) \notin X$, $(e'_1, e'_2) \prec (e_1, e_2)$. W.l.o.g. $e'_1 \prec_1 e_1$. Since $\Pi_1(X) \in Conf([\![P_1]\!]_F)$, there exists $e''_1 \in \Pi_1(X) : e'_1 \#_1 e''_1 \prec_1 e_1$ and $\exists (e''_1, e''_2) \in X$ with
  $(e'_1, e'_2) \# (e''_1, e''_2) \prec (e_1, e_2)$. Contradiction!

- $\mathcal{C}' \subseteq \mathcal{C}$
  Let $X \in \mathcal{C}'$. Then $X \subseteq E_1 \times_A E_2$ and with corollary 4.2.9 $\Pi_i(X) \in Conf(\llbracket P_i \rrbracket_F) = \mathcal{C}_i$. Finally Π_i is injective on X, since otherwise X contained conflicts.

- $\surd' \subseteq \surd$
  Let $X \notin \surd$, d.h. $\Pi_1(X) \notin \surd_1 \vee \Pi_2(X) \notin \surd_2$. W.l.o.g. $\exists e_1 \notin \Pi_1(X)$ with $\forall e_1' \in \Pi_1(X) : \neg(e_1' \# e_1)$, then $(e_1, *) \notin X$ and $\forall (e_1', e_2') \in X : \neg(e_1', e_2') \# (e_1, *)$, consequently $X \notin \surd'$.

- $\surd \subseteq \surd'$. Let $X \notin \surd'$, i.e. $\exists (e_1, e_2) \notin X$ with $\forall (e_1', e_2') \in X : \neg(e_1', e_2') \# (e_1, e_2)$. W.l.o.g. $e_1 \neq *$, consequently $e_1 \notin \Pi_1(X)$ (otherwise $(e_1, e_2') \in X, (e_1, e_2') \# (e_1, e_2)$) and $\forall e_1' \in \Pi_1(X) : \neg(e_1' \#_1 e_1$, consequently $\Pi_1(X) \notin \surd_1$ and we conclude $X \notin \surd$.

- $l = l'$ obvious

Proposition 4.3.3: For all terms $P \in \mathbf{L}$ the following holds: $\llbracket P \rrbracket_K = \mathcal{C}(\llbracket P \rrbracket_F)$.

Proof: induction over the syntactic structure of P with lemma 4.3.1 and lemma 4.3.2.

■

Corollary 4.3.4: $\cong_d$ is a congruence on $\mathbf{L}$.

5 Syntactic Refinement

5.1 Introduction and Motivation

As mentioned above [GGR92] defined a syntactic refinement, compared it with semantic refinement and showed that in case of refining synchronizing actions it coincides with semantic refinement only under fairly restrictive conditions. We will call the syntactic refinement of [GGR92] syntactic substitution and define a new kind of syntactic refinement.

The difficulties in [GGR92] arise by the refinement of synchronizing actions: e.g. $(P_1 \parallel_A P_2)[a \rightsquigarrow Q]$ and $a \in A$. In this case the semantic refinement operator does not necessarily distribute over the parallel composition, which is the case for syntactic substitution.

If one understands the refinement of an action a as the instantiation of a procedure call it is quite natural to understand a synchronizing action in the following way: An action name stands for an agreement among the communicating partners to execute the procedure just once and to distribute the result. This is the way we understand semantic refinement.

In order to simulate this in a syntactic way one possibility is to put another process in parallel that takes charge of the execution of the procedure. Let a_1, a_2 be calling- and returning actions and $*$ an operator for repetition. So the definition of syntactic refinement could be the following:

$(P_1 \parallel_A P_2)[Q/a] := (P_1[(a_1;a_2)/a] \parallel_{A\setminus\{a\}\cup\{a_1,a_2\}} P_2[(a_1;a_2)/a]) \parallel_{\{a_1;a_2\}} (a_1;Q;a_2)^*$

But first it is rather complicated to put a third process in parallel, and secondly it is not quite intuitive, thus a a simplification could be to charge one process with the execution of the synchronizing procedure. The other process then synchronizes with it:

$(P_1 \parallel_A P_2)[Q/a] := (P_1[(a_1;Q;a_2)/a] \parallel_{A\setminus\{a\}\cup\{a_1,a_2\}} P_2[(a_1;a_2)/a])$

If one is not interested in the synchronization points one has to define an appropriate hiding operator $\setminus$ and one gets:

$(P_1 \parallel_A P_2)[Q/a] := (P_1[(a_1;Q;a_2)/a] \parallel_{A\setminus\{a\}\cup\{a_1,a_2\}} P_2[(a_1;a_2)/a]) \setminus \{a_1, a_2\}$

We will give a simple criterion under which circumstances the new syntactic refinement coincides with semantic refinement with respect to a rather strong equivalence relation.

Only well-formed terms will be taken into account.

5.2 A new kind of syntactic refinement

5.2.1 The new definition

With exception to the case $(P_1 \parallel_A P_2)[a \rightsquigarrow Q]$ with $a \in A$, the notion of syntactic refinement in [GGR92] coincides with textual replacement as one could see in definition 2.1.4. Moreover [GGR92] show that syntactic refinement and semantic refinement yield isomorphic flow event structures for all terms in L that do not belong to the same exceptional class.

Lemma 5.2.1 Let $P, P_1, P_2, Q \in \mathbf{L^o}$, $a, b \in Act$, $A \subseteq Act$. Then the following equivalences hold:

1. $a[a \rightsquigarrow Q] \cong_e Q$
2. $b[a \rightsquigarrow Q] \cong_e b$ (if $b \neq a$)
3. $(P_1; P_2)[a \rightsquigarrow Q] \cong_e P_1[a \rightsquigarrow Q]; P_2[a \rightsquigarrow Q]$
4. $(P_1 + P_2)[a \rightsquigarrow Q] \cong_e P_1[a \rightsquigarrow Q] + P_2[a \rightsquigarrow Q]$
5. $(P_1 \parallel_A P_2)[a \rightsquigarrow Q] \cong_e P_1[a \rightsquigarrow Q] \parallel_A P_2[a \rightsquigarrow Q]$ (if $a \notin A$)

(taken from [GGR92], lemma 4.1)

Lemma 5.2.2: Let $P, Q \in \mathbf{L^o}$, $a \in Act$. If $a \notin S(P)$, then: $P[a \rightsquigarrow Q] \cong_e P\{\frac{Q}{a}\}$.

(taken from [GGR92], theorem 4.3)

But for terms with syntactic structure $(P_1 \parallel_A P_2)[a \rightsquigarrow Q]$ with $a \in A$ the syntactic substitution and the semantic refinement coincide only under fairly restrictive conditions.

The syntactic substitution of [GGR92] is defined in this case:
$(P_1 \parallel_A P_2)\{\frac{Q}{a}\} := (P_1 \parallel_{A\setminus\{a\}\cup L(Q)} P_2)$, if $a \in A$.

As one can see in the following example the semantic of $(P_1 \parallel_A P_2)\{\frac{Q}{a}\}$ does not even necessarily preserve interleaving trace equivalence:

Let $P_1 := (a; c \parallel_\emptyset a; c), P_2 := a, Q := (b; b + b), A := \{a\}$.
Then $\langle b.b.c.c\rangle \notin Traces((P_1 \parallel_A P_2)[a \rightsquigarrow Q]) = Traces(((a; c \parallel_\emptyset a; c) \parallel_{\{a\}} a)[a \rightsquigarrow (b; b + b)])$.
But $\langle b.b.c.c\rangle \in Traces(P_1[a \rightsquigarrow Q] \parallel_{A\setminus\{a\}\cup L(Q)} P_2[a \rightsquigarrow Q])$ since with lemma 5.2.1
$P_1[a \rightsquigarrow Q] \parallel_{A\setminus\{a\}\cup L(Q)} P_2[a \rightsquigarrow Q]) \cong_e ((b; b + b); c \parallel_\emptyset (b; b + b); c) \parallel_b (b; b + b)$.

In order to avoid problems arising by executing actions of Q repeatedly and parallel to each other we define syntactic refinement like a procedure call. The idea is that one process executes the procedure and the other only synchronizes at the beginning and the end of the call.

Define syntactic refinement $[Q/a]$ inductively over the syntactic structure of the terms:

Definition 5.2.3 Let $P, P_1, P_2, Q \in \mathbf{L^o}$ and $\{a_1, a_2\} \cap (L(Q) \cup L(P_1) \cup L(P_2) \cup A) = \emptyset$.

$$\begin{array}{rcl} b[Q/a] & := & \begin{cases} Q & \text{if } b = a \\ b & \text{otherwise} \end{cases} \\ (P_1; P_2)[Q/a] & := & P_1[Q/a]; P_2[Q/a] \\ (P_1 + P_2)[Q/a] & := & P_1[Q/a] + P_2[Q/a] \\ (P_1 \parallel_A P_2)[Q/a] & := & P_1[Q/a] \parallel_A P_2[Q/a] \text{ (if } a \notin A) \\ (P_1 \parallel_A P_2)[Q/a] & := & (P_1[(a_1; Q; a_2)/a] \parallel_{A\setminus\{a\}\cup\{a_1,a_2\}} P_2[(a_1; a_2)/a]) \setminus \{a_1, a_2\} \text{ (if } a \in A). \end{array}$$

Note that the result of the syntactic refinement of a term lies in a language $\mathbf{L^o}'$ which consists of $\mathbf{L^o}$ plus a hiding-operator. Apart from the last clause the definition is equivalent to syntactic substitution.

(Note that a definition
$(P_1 \parallel_A P_2)[Q/a] = (P_1[(a_1; Q; a_2)/a] \parallel_{A\setminus\{a\}\cup\{a_1,a_2\}\cup L(Q)} P_2[(a_1; Q; a_2)/a]) \setminus \{a_1, a_2\}$ would not solve the problem. Consider for example $P_1 = P_2 = a, A = \{a\}, Q = c; d + c$. Then a maximal configuration $\{e_0^{a_1}, e_1^c\}$ would exist for the syntactically refined term (before hiding) and not for the semantically refined one.)

With lemma 5.2.1 and 5.2.2 one sees that for all terms $P, Q \in \mathbf{L^o}$ with $a \notin S(Q)$
$P[a \rightsquigarrow Q] \cong_e P[Q/a]$ holds.

Now the conditions when $(P_1 \parallel_A P_2)[Q/a]$ and $(P_1 \parallel_A P_2)[a \rightsquigarrow Q]$ coincide will be studied.

5.2.2 Equivalence of syntactic and semantic refinement

Syntactic refinement without hiding

For simplicity first no hiding operator will be considered. Thus the conditions when $(P_1 \parallel_A P_2)[(a_1;Q;a_2)/a]$ and $(P_1 \parallel_A P_2)[a \rightsquigarrow a_1;Q;a_2]$ are equivalent will be examined. First of all an appropriate equivalence relation has to be chosen.

[GGR92] chose $\cong_c$ – but as one sees in the example below this relation is not suitable for our purposes.

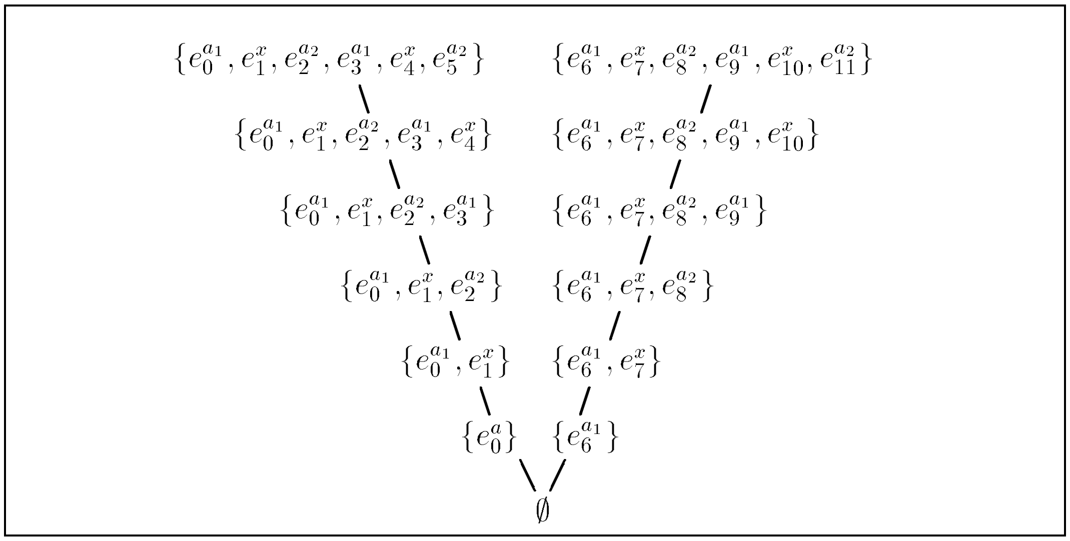
$\{e_0^{a_1}, e_1^x, e_2^{a_2}, e_3^{a_1}, e_4^x, e_5^{a_2}\}$
$\{e_6^{a_1}, e_7^x, e_8^{a_2}, e_9^{a_1}, e_{10}^x, e_{11}^{a_2}\}$
$\{e_0^{a_1}, e_1^x, e_2^{a_2}, e_3^{a_1}, e_4^x\}$
$\{e_6^{a_1}, e_7^x, e_8^{a_2}, e_9^{a_1}, e_{10}^x\}$
$\{e_0^{a_1}, e_1^x, e_2^{a_2}, e_3^{a_1}\}$
$\{e_6^{a_1}, e_7^x, e_8^{a_2}, e_9^{a_1}\}$
$\{e_0^{a_1}, e_1^x, e_2^{a_2}\}$
$\{e_6^{a_1}, e_7^x, e_8^{a_2}\}$
$\{e_0^{a_1}, e_1^x\}$
$\{e_6^{a_1}, e_7^x\}$
$\{e_0^{a}\}$
$\{e_6^{a_1}\}$
$\emptyset$

Figure 5: $\mathit{Conf}((\llbracket P_1 \rrbracket \parallel_A \llbracket P_2 \rrbracket)[a \rightsquigarrow a_1;x;a_2])$

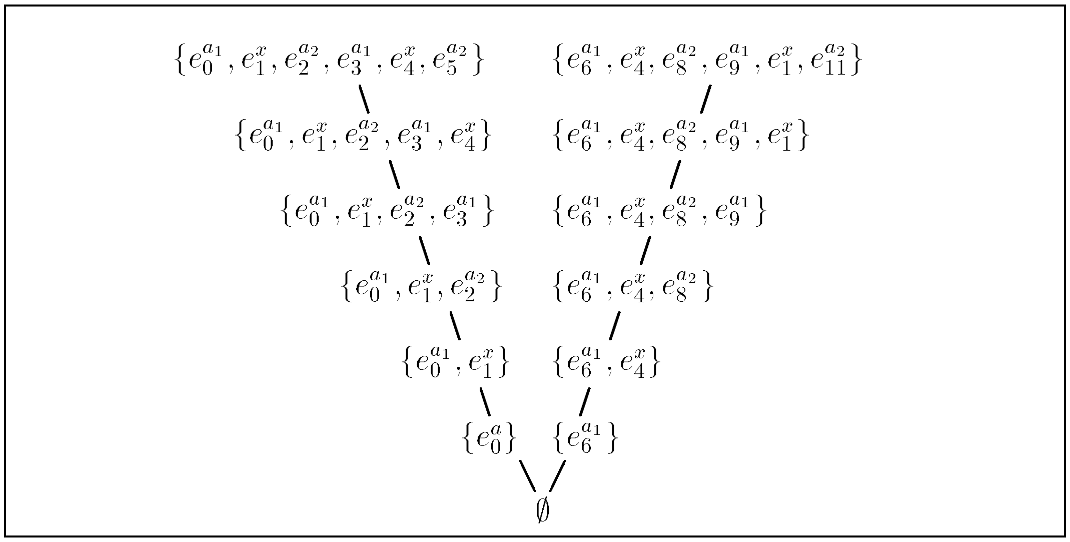
$\{e_0^{a_1}, e_1^x, e_2^{a_2}, e_3^{a_1}, e_4^x, e_5^{a_2}\}$
$\{e_6^{a_1}, e_4^x, e_8^{a_2}, e_9^{a_1}, e_1^x, e_{11}^{a_2}\}$
$\{e_0^{a_1}, e_1^x, e_2^{a_2}, e_3^{a_1}, e_4^x\}$
$\{e_6^{a_1}, e_4^x, e_8^{a_2}, e_9^{a_1}, e_1^x\}$
$\{e_0^{a_1}, e_1^x, e_2^{a_2}, e_3^{a_1}\}$
$\{e_6^{a_1}, e_4^x, e_8^{a_2}, e_9^{a_1}\}$
$\{e_0^{a_1}, e_1^x, e_2^{a_2}\}$
$\{e_6^{a_1}, e_4^x, e_8^{a_2}\}$
$\{e_0^{a_1}, e_1^x\}$
$\{e_6^{a_1}, e_4^x\}$
$\{e_0^{a}\}$
$\{e_6^{a_1}\}$
$\emptyset$

Figure 6: $\mathit{Conf}((\llbracket P_1 \rrbracket \parallel_A \llbracket P_2 \rrbracket)[(a_1;x;a_2)/a])$

Let $P_1 = a \parallel_\emptyset a$, $P_2 = a;a$, $Q = x$, $A = \{a\}$.

In Figure 5 we see that in case of semantic refinement four events (e_1, e_4, e_7, e_{10}) with label x are constructed. As we see in Figure 6 in case of syntactic refinement only two events (e_1, e_4) with label x are constructed.

The reason for this is that in the case of syntactic refinement a procedure call is used – for example e_1 corresponds to the execution of x on the left hand side and e_4 corresponds to the execution of x on the right hand side of P_1. Of course both events have to be contained in each complete configuration. In the case of semantic refinement however there is first a decision which a synchronizes with which one – and depending on this choice x will be "called".

It turns out however that the domains of configurations are equivalent. Thus the equivalence relation $\cong_d$ seems to be natural for this kind of syntactic refinement.

The following example shows that syntactic and semantic refinement do not coincide with respect to $\cong_d$ for all terms. In fact there exist terms for which syntactic and semantic refinement even do not coincide with respect to interleaving trace equivalence ($\approx_{it}$).

Let $P_1 = b; a \parallel_\emptyset c; a$, $P_2 = f; a; d \parallel_\emptyset g; a; e$ and $Q = x$. Then
$(P_1 \parallel_{\{a\}} P_2)[a \rightsquigarrow a_1; Q; a_2] \not\approx_{it} P_1[a \rightsquigarrow a_1; Q; a_2] \parallel_{\{a_1,a_2\}} P_2[a \rightsquigarrow a_1; a_2]$, because
$\langle b, c, f, a_1, x, g, a_1, a_2, e\rangle$ is a trace of the second term but not of the first one.

In this example the problem arisis from a being in parallel to itself in $P_1 \parallel_A P_2$. It will be shown that syntactic and semantic refinement coincide with respect to $\cong_d$ if the refined a is not in parallel to itself:

Definition 5.2.4: Let $P \in \mathbf{L}$. An action $a \in L(P)$ is called *auto-concurrent* if there exist configurations $X, Y, Z \in Conf(\llbracket P \rrbracket)$ with $Y \neq Z$ & $X \to^a Y$ & $X \to^a Z$ and
$Y \cup Z \in Conf(\llbracket P \rrbracket)$

Proposition 5.2.5 Let $P_1, P_2 \in \mathbf{L^o}$, $A \subseteq Act$, $a \in A$. If a is not auto-concurrent in $P_1 \parallel_A P_2$ and $(L(P_1) \cup L(P_2)) \cap L(Q) = \emptyset$ and $(L(P_1) \cup L(P_2) \cup L(Q)) \cap \{a_1, a_2\} = \emptyset$ then
$(P_1 \parallel_A P_2)[a \rightsquigarrow a_1; Q; a_2] \cong_d P_1[a \rightsquigarrow a_1; Q; a_2] \parallel_{A'} P_2[a \rightsquigarrow a_1; a_2]$ holds with
$A' = A \setminus \{a\} \cup \{a_1, a_2\}$.

Proof: see Appendix A

A Hiding-Operator

To complete our definition of syntactic refinement we now have to define a hiding operator. It is very easy to define hiding on prime event structures:

Definition 5.2.6 (hiding on prime event structures)
Let $\mathcal{E} = (E, \leq, \#, l)$ be a prime event structure, $A \subseteq Act$. Define
$\mathcal{E} \setminus_P A := (E', \leq \cap (E' \times E'), \# \cap (E' \times E'), l\lceil E')$ with $E' := \{e \in E \mid l(e) \notin A\}$.

A hiding operator $\setminus_F$ on flow event structures should be consistent with the one on prime event structures, i.e. it is useful to demand: $\mathcal{E} \setminus_F A \cong_d \mathcal{P}(\mathcal{E}) \setminus_P A$ for each flow event structure $\mathcal{E}$.

The definition for hiding on flow event structures is not so easy because one event can have different roles in the event structure. Consider for example the event structure $\mathcal{E}$ in Figure 7:

In fact it is not possible to define an event structure $\mathcal{E}'$ consisting only of the events $\{e_0, e_2, e_3\}$ which is domain isomorphic to $\mathcal{P}(\mathcal{E}) \setminus \{b\}$.

Thus we define hiding on flow event structures according to hiding on prime event structures:

Definition 5.2.7 (hiding on flow event structures)
Let $\mathcal{E} = (E, \prec, \#, l)$ be a flow event structure and $A \subseteq Act$. Define $\mathcal{E} \setminus_F A := \mathcal{P}(\mathcal{E}) \setminus_P A$.

Lemma 5.2.8 Let $\mathcal{E}_1, \mathcal{E}_2$ be two flow event structures with $\mathcal{E}_1 \cong_d \mathcal{E}_2$. Then $\mathcal{E}_1 \setminus_F A \cong_d \mathcal{E}_2 \setminus_F A$.

Proof: Obvious, since the definition is made via the configuration structure.

Now we show that refining an event structure with $[\![a_1; P; a_2]\!]_F$ and then hiding a_1, a_2 leads to the same result as refining it only with $[\![P]\!]_F$:

Lemma 5.2.9 Let $\mathcal{E} = (E, \prec, \#, l)$ and $\mathcal{F} = (E_\mathcal{F}, \prec_\mathcal{F}, \#_\mathcal{F}, l_\mathcal{F})$ be arbitrary event structures with $a_1, a_2 \notin l(E) \cup l_\mathcal{F}(E_\mathcal{F})$. Then $\mathcal{E}[a \rightsquigarrow \mathcal{F}] \cong_d (\mathcal{E}[a \rightsquigarrow \mathbf{a_1}; \mathcal{F}; \mathbf{a_2}]) \setminus_F \{a_1, a_2\}$ (with $\mathbf{a_i}$ being the event structures $(\{x_i\}, \emptyset, \emptyset, \{(x_i, a_i)\})$.

Proof:
Let $\{x_1, x_2\} \notin E_\mathcal{F}$. Then $\mathcal{E}[a \rightsquigarrow \mathbf{a_1}; \mathcal{F}; \mathbf{a_2}] = (E_1, \prec_1, \#_1, l_1) = \mathcal{E}_1$ with
$E_1 := \{(e, *) \mid e \in E, l(e) \neq a\} \cup \{(e, f) \mid e \in E, l(e) = a, f \in E_\mathcal{F} \cup \{x_1, x_2\}\}$,
$(e, f) \prec_1 (e', f') :\Leftrightarrow (e \prec e') \vee (e = e' \;\&\; (f \prec_\mathcal{F} f' \vee f = x_1 \vee f' = x_2))$,
$(e, f) \#_1 (e', f') :\Leftrightarrow (e \# e') \vee (e = e' \;\&\; f \#_\mathcal{F} f'))$,

$$l_1(e, f) := \begin{cases} l(e) & \text{if } f = * \\ l_\mathcal{F}(f) & \text{if } f \in E_\mathcal{F} \\ a_1 & \text{if } f = x_1 \\ a_2 & \text{if } f = x_2 \end{cases}$$

Let $\mathcal{E}_1' := \mathcal{E}[a \rightsquigarrow \mathbf{a_1}; \mathcal{F}; \mathbf{a_2}] \setminus_F \{a_1, a_2\} = \mathcal{P}(\mathcal{E}_1) \setminus_P A = (E_1', \leq_1', \#_1', l_1')$ with
$E_1' := \{\downarrow_X (e, f) \mid (e, f) \in X \in Conf(\mathcal{E}_1), f \notin \{x_1, x_2\}\}$,
$X \leq_1' Y :\Leftrightarrow X \subseteq Y$,
$X \#_1' Y :\Leftrightarrow X \cup Y \notin Conf(\mathcal{E}_1)$,
$l_1'(\downarrow_X (e, f)) = l_1(e, f)$ (see lemma 2.6.7).

Let $\mathcal{E}_2 := \mathcal{E}[a \rightsquigarrow \mathcal{F}] = (E_2, \prec_2, \#_2, l_2)$ with
$E_2 := \{(e, *) \mid e \in E, l(e) \neq a\} \cup \{(e, f) \mid e \in E, l(e) = a, f \in E_\mathcal{F}\}$,
$(e, f) \prec_2 (e', f') :\Leftrightarrow (e \prec e') \vee (e = e' \;\&\; f \prec_\mathcal{F} f')$,
$(e, f) \#_2 (e', f') :\Leftrightarrow (e \# e') \vee (e = e' \;\&\; f \#_\mathcal{F} f')$,

$$l_2(e, f) := \begin{cases} l(e) & \text{if } f = * \\ l_\mathcal{F}(f) & \text{otherwise} \end{cases}$$

and let $\mathcal{E}_2' := \mathcal{P}(\mathcal{E}_2) = (E_2', \leq_2', \#_2', l_2')$ with
$E_2' = \{\downarrow_X (e, f) \mid (e, f) \in X \in Conf(\mathcal{E}_2)\}$,
$X \leq_2' Y :\Leftrightarrow X \subseteq Y$,

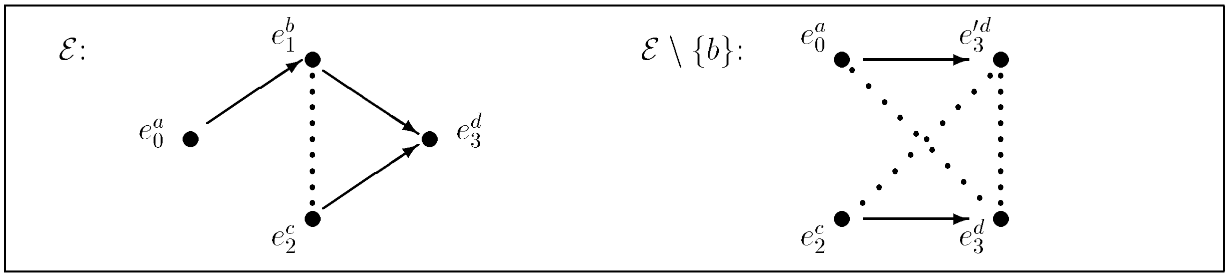

Figure 7: A flow event structure $\mathcal{E}$ and $\mathcal{E} \setminus \{b\}$

$X \#_2' Y :\Leftrightarrow X \cup Y \notin Conf(\mathcal{E}_2)$,
$l_2'(\downarrow_X (e, f)) := l_2(e, f)$.

Now we show that $\mathcal{E}_1' \cong_e \mathcal{E}_2'$, i.e. $\mathcal{E}[a \rightsquigarrow \mathbf{a_1}; \mathcal{F}; \mathbf{a_2}] \setminus_F \{a_1, a_2\} \cong_e \mathcal{P}(\mathcal{E}[a \rightsquigarrow \mathcal{F}])$. With remark 2.6.6 and lemma 2.6.7 then follows $\mathcal{E}[a \rightsquigarrow \mathbf{a_1}; \mathcal{F}; \mathbf{a_2}] \setminus_F \{a_1, a_2\} \cong_d \mathcal{E}[a \rightsquigarrow \mathcal{F}]$.

Define $h : E_1' \to E_2'$: For each $X = \downarrow_X (e, f) \in E_1'$ define
$h(\downarrow_X (e, f)) := \{(e', f') \in X \mid f' \notin \{x_1, x_2\}\} = X' = \downarrow_{X'} (e, f)$.

h is well-defined because for all $X \in Conf(\mathcal{E}_1) : \{(e', f') \in X \mid f' \notin \{x_1, x_2\}\} \in Conf(\mathcal{E}_2)$.
h is injective:
Suppose exist $\downarrow_{X_1} (e_1, f_1) \neq \downarrow_{X_2} (e_2, f_2)$ in E_1' such that $h(\downarrow_{X_1} (e_1, f_1)) = h(\downarrow_{X_2} (e_2, f_2))$.
Then $(e_1, f_1) = (e_2, f_2)$ and with lemma 3.2.5 exists
$(e_1', f_1') \in \downarrow_{X_1} (e_1, f_1), (e_2', f_2') \in \downarrow_{X_2} (e_2, f_2)$ with $(e_1', f_1') \#_1 (e_2', f_2')$. Since $h(\downarrow_{X_1} (e_1, f_1))$ contains no conflicts $f_1' \in \{x_1, x_2\} \vee f_2' \in \{x_1, x_2\}$ – but then $e_1' \# e_2'$. Contradiction!
h is surjective:
Let $Y = \downarrow_Y (e, f) \in E_2'$. Define $h^{-1}(Y) := X =$
$Y \cup \{(e', x_1) \mid \exists f' \in E_{\mathcal{F}} : (e', f') \in Y\} \cup \{(e', x_2) \mid \exists f' \in E_{\mathcal{F}} : (e', f') \in Y \ \& \ \exists (e'', f'') \in Y : e' \prec e''\}$.
As one easily verifies $X \in Conf(\mathcal{E}_1)$, $X = \downarrow_X (e, f) \in E_1'$ and $h(X) = \downarrow_Y (e, f)$.

h is an isomorphism:
Obviously $\downarrow_X (e, f) \leq_1' \downarrow_{X'} (e', f') \Leftrightarrow h(\downarrow_X (e, f)) \leq_2' h(\downarrow_{X'} (e', f'))$ and
$\downarrow_X (e, f) \#_1' \downarrow_{X'} (e', f') \Leftrightarrow h(\downarrow_X (e, f)) \#_2' h(\downarrow_{X'} (e', f'))$ and
$\forall \downarrow_X (e, f) \in E_1' : l_1'(\downarrow_X (e, f)) = l_1(e, f) = l_2(e, f) = l_2'(h(\downarrow_X (e, f)))$.

Thus $\mathcal{E}[a \rightsquigarrow \mathbf{a_1}; \mathcal{F}; \mathbf{a_2}] \setminus_F \{a_1, a_2\} \cong_d \mathcal{E}[a \rightsquigarrow \mathcal{F}]$.

■

We now show the consistency of syntactic and semantic refinement:

Proposition 5.2.10 Let $P_1, P_2 \in \mathbf{L^o}$, $A \subseteq Act$, $a \in A$. If a is not auto-concurrent in $P_1 \parallel_A P_2$ and $P_1[a \rightsquigarrow a_1; Q; a_2] \cong_d P_1[(a_1; Q; a_2)/a]$ and $P_2[a \rightsquigarrow a_1; a_2] \cong_d P_2[(a_1; a_2)/a]$ then $(P_1 \parallel_A P_2)[a \rightsquigarrow Q] \cong_d (P_1 \parallel_A P_2)[Q/a]$ holds.

Proof:
With lemma 5.2.5 we know that
$(P_1 \parallel_A P_2)[a \rightsquigarrow a_1; Q; a_2] \cong_d P_1[a \rightsquigarrow a_1; Q; a_2] \parallel_A P_2[a \rightsquigarrow a_1; a_2]$.
And with lemma 5.2.8 we therefore conclude
$[\![(P_1 \parallel_A P_2)[a \rightsquigarrow a_1; Q; a_2]]\!] \setminus_F \{a_1, a_2\} \cong_d [\![P_1[a \rightsquigarrow a_1; Q; a_2] \parallel_A P_2[a \rightsquigarrow a_1; a_2]]\!] \setminus_F \{a_1, a_2\}$.

With lemma 5.2.9 we know that
$[\![(P_1 \parallel_A P_2)[a \rightsquigarrow a_1; Q; a_2]]\!] \setminus_F \{a_1, a_2\} \cong_d [\![(P_1 \parallel_A P_2)[a \rightsquigarrow Q]]\!]$ and therefore we see that
$[\![(P_1 \parallel_A P_2)[Q/a] = [\![P_1[a \rightsquigarrow a_1; Q; a_2] \parallel_A P_2[a \rightsquigarrow a_1; a_2]]\!] \setminus_F \{a_1, a_2\} \cong_d [\![(P_1 \parallel_A P_2)[a \rightsquigarrow Q]]\!]$.

■

Proposition 5.2.11 Let $P \in \mathbf{L^o}$. If P does not contain a term $P_1 \parallel_A P_2, a \in A$ with a auto-concurrent in $P_1 \parallel_A P_2$, then $P[a \rightsquigarrow Q] \cong_d P[Q/a]$ holds.

Proof: induction over the syntactic structure of P:
For $P = a$, $P = b \neq a$, okay.
For $P = P_1 * P_2$ with $* \in \{;, +, \parallel_A\}$, $(a \notin A)$ lemma 5.2.1 claims that
$(P_1 * P_2)[a \rightsquigarrow Q] \cong_e P_1[a \rightsquigarrow Q] * P_2[a \rightsquigarrow Q]$. With the induction hypothesis one concludes

$P_i[a \rightsquigarrow Q] \cong_d P_i[Q/a]$ if P_1, P_2 do not contain forbidden terms. With definition 5.2.3 one sees $P_1[Q/a] * P_2[Q/a] \cong_d (P_1 * P_2)[Q/a]$ and therefore $(P_1 * P_2)[a \rightsquigarrow Q] \cong_d (P_1 * P_2)[Q/a]$.

If $P = P_1 \parallel_A P_2$, $a \in A$, and if a is not auto-concurrent in P we conclude with proposition 5.2.10 and with the induction hypothesis that $P[a \rightsquigarrow Q] \cong_d P[Q/a]$.

■

5.3 Conclusion and comparison with [GGR92]

We showed that $(P_1 \parallel_A P_2)[a \rightsquigarrow Q]$ with $a \in A$ coincides up to $\cong_d$ with $(P_1[a \rightsquigarrow a_1; Q; a_2] \parallel_{A \setminus \{a\} \cup \{a_1, a_2\}} P_2[a \rightsquigarrow a_1; a_2]) \setminus \{a_1, a_2\}$ if $a \in A$ and a is not auto-concurrent in $P_1 \parallel_A P_2$.

This result is more powerful than the one for syntactic substitution in [GGR92]. [GGR92] showed $(P_1 \parallel_A P_2)[a \rightsquigarrow Q] \cong_c (P_1 \parallel_A P_2)\{\frac{Q}{a}\}$ under the hypothesis that either Q atomic or
Q deterministic and a two-way-sequential in $P_1 \parallel_A P_2$ or
Q distinct and a not auto-concurrent in $P_1 \parallel_A P_2$.

Q is called *deterministic* iff $\forall a \in L(Q) : \nexists F, G \neq H \in Conf(\llbracket Q \rrbracket)$ with $F \rightarrow^a G$ & $F \rightarrow^a H$.

Q is called *atomic* iff Q is deterministic and each action in Q is *initial-only*, i.e. $\forall a \in L(Q), F, G \in Conf(\llbracket Q \rrbracket)$ with $F \rightarrow^a G$: $F = \emptyset$.

Q is called *distinct* iff Q is deterministic and each initial action in Q is initial-only (an action is called *initial* in Q iff $\exists F \in Conf(\llbracket Q \rrbracket)$ with $\emptyset \rightarrow^a F$).

a being *two-way-sequential* in $P_1 \parallel_A P_2$ is a bit weaker than the requirement of a being not auto-concurrent in P_1 and not being auto-concurrent in P_2.

In each case Q has to satisfy some restrictive hypothesis. This is not the case in our version. All terms satisfying the conditions of [GGR92] also satisfy the condition of a not being auto-concurrent in $P_1 \parallel_A P_2$. Therefore the new definition of syntactic refinement is more powerful than the syntactic substitution in [GGR92].

[GGR92] chose an equivalence relation stronger than the one we use. But the following examples show that the syntactic substitution of [GGR92] easily violates the equivalence $\cong_d$ if the conditions above are not satisfied.

One example we saw already above was $P_1 = (a; c \parallel_\emptyset a; c), P_2 = a, Q = (b; b + b), A = \{a\}$.
$\langle b.b.c.c \rangle$ is no trace of $(P_1 \parallel_A P_2)[a \rightsquigarrow Q]$, but it is a trace of $(P_1 \parallel_A P_2)\{\frac{Q}{a}\}$ because
$(P_1 \parallel_A P_2)\{\frac{Q}{a}\} \cong_e ((b; b + b); c \parallel_\emptyset (b; b + b); c) \parallel_b (b; b + b)$.
Thus $(P_1 \parallel_A P_2)[a \rightsquigarrow Q] \not\approx_{it} (P_1 \parallel_A P_2)\{\frac{Q}{a}\}$.
On the other hand $(P_1 \parallel_A P_2)[Q/a] \cong_d (P_1 \parallel_A P_2)[a \rightsquigarrow Q]$ (because a is auto-concurrent in P_1 but not in P_2 and therefore proposition 5.2.11 can be used).

If Q is non-deterministic the conditions of [GGR92] are not satisfied. Consider for example $P_1 = P_2 = a$, $A = \{a\}, Q = c; d + c$

As Figure 8 shows: $(P_1 \parallel_A P_2)[a \rightsquigarrow Q] \rrbracket) \ncong_c (P_1 \parallel_A P_2)\{\frac{Q}{a}\}$ and
$(P_1 \parallel_A P_2)[a \rightsquigarrow Q] \rrbracket) \ncong_d (P_1 \parallel_A P_2)\{\frac{Q}{a}\}$
(but $(P_1 \parallel_A P_2)[a \rightsquigarrow Q] \rrbracket) \cong_d (P_1 \parallel_A P_2)[Q/a]$).

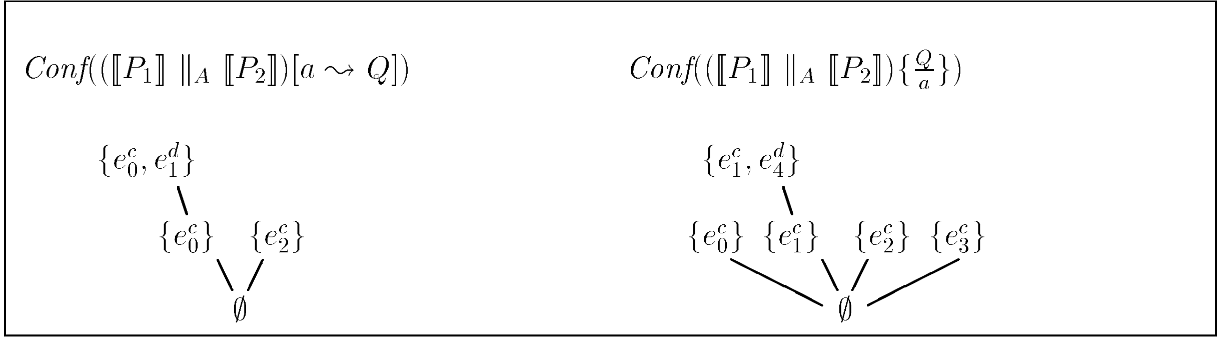
$Conf(([\![P_1]\!] \parallel_A [\![P_2]\!])[a \rightsquigarrow Q])$
$\{e_0^c, e_1^d\}$
$\{e_0^c\}$
$\{e_2^c\}$
$\emptyset$
$Conf(([\![P_1]\!] \parallel_A [\![P_2]\!])\{\frac{Q}{a}\})$
$\{e_1^c, e_4^d\}$
$\{e_0^c\}$
$\{e_1^c\}$
$\{e_2^c\}$
$\{e_3^c\}$
$\emptyset$

Figure 8: $([\![P_1]\!] \parallel_A [\![P_2]\!])[a \rightsquigarrow Q]$ versus $([\![P_1]\!] \parallel_A [\![P_2]\!])\{\frac{Q}{a}\}$

But as one sees in Figure 9: $(P_1 \parallel_A P_2)[a \rightsquigarrow Q] \cong_d (P_1 \parallel_A P_2)[Q/a]$ holds.

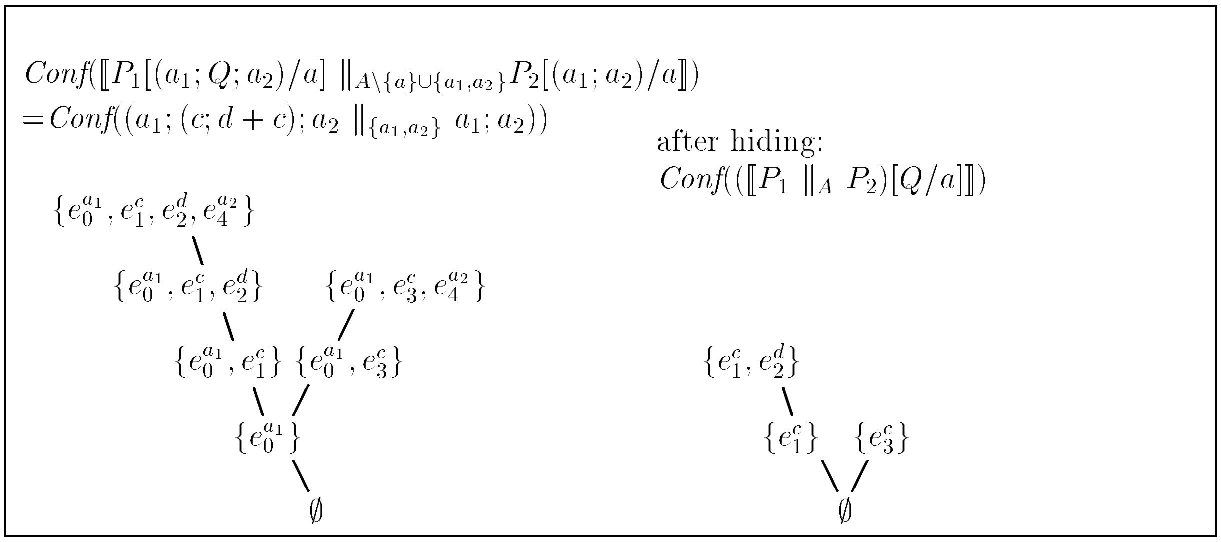
$Conf([\![P_1[(a_1;Q;a_2)/a] \parallel_{A\setminus\{a\}\cup\{a_1,a_2\}} P_2[(a_1;a_2)/a]]\!])$
$= Conf((a_1;(c;d+c);a_2 \parallel_{\{a_1,a_2\}} a_1;a_2))$
after hiding:
$Conf(([\![P_1 \parallel_A P_2)[Q/a]]\!])$
$\{e_0^{a_1}, e_1^c, e_2^d, e_4^{a_2}\}$
$\{e_0^{a_1}, e_1^c, e_2^d\}$
$\{e_0^{a_1}, e_3^c, e_4^{a_2}\}$
$\{e_0^{a_1}, e_1^c\}$
$\{e_0^{a_1}, e_3^c\}$
$\{e_0^{a_1}\}$
$\emptyset$
$\{e_1^c, e_2^d\}$
$\{e_1^c\}$
$\{e_3^c\}$
$\emptyset$

Figure 9: $([\![P_1[(a_1;Q;a_2)/a] \parallel_{A\setminus\{a\}\cup\{a_1,a_2\}} P_2[(a_1;a_2)/a]]\!])$ and $([\![P_1 \parallel_A P_2]\!])[Q/a]$

5.4 Outlook and Future Work

It would be reasonable to give a definition for syntactic refinement including the case of a being auto-concurrent in $P_1 \parallel_A P_2$. With this one could develop a syntactic refinement that always preserves $\cong_d$-equivalence to semantical refined terms – but possibly leads to complicated terms.

Another possiblity is to check if there is a better version of syntactic refinement, i.e. one such that a reasonable equivalence relation is always preserved between syntactic and semantic refinement. One could reasonably use the following condition suggested in [GGR92]:

$P_1[a \rightsquigarrow Q_1] \parallel_{A \setminus \{a\} \cup A'} P_2[a \rightsquigarrow Q_2]$ with $(Q_1 \parallel_{A'} Q_2) \approx Q$ for an appropriate equivalence relation $\approx$.

With the help of such a condition it might on the other hand be also possible to show that the results obtained so far are optimal and that there is no inherent symmetry between syntactic and semantic refinement.

A Proof of Proposition 5.2.5

A.1 Definitions

Let $P_1, P_2, Q \in \mathbf{L^o}$. We show: If $a \in A$, $a_1, a_2 \notin L(P_1) \cup L(P_2) \cup L(Q)$, $(L(P_1) \cup L(P_2)) \cap L(Q) = \emptyset$ and a not auto-concurrent in $P_1 \parallel_A P_2$, then $(P_1 \parallel_A P_2)[a \rightsquigarrow a_1; Q; a_2] \cong_d P_1[a \rightsquigarrow a_1; Q; a_2] \parallel_{A \setminus \{a\} \cup \{a_1, a_2\}} P_2[a \rightsquigarrow a_1; a_2]$ holds.

Let $P := P_1 \parallel_A P_2$.

Let $\mathcal{E}_1 = (E_1, \prec_1, \#_1, l_1) = [\![P_1]\!], \mathcal{E}_2 = (E_2, \prec_2, \#_2, l_2) = [\![P_2]\!]$ and $\mathcal{E}_Q = (E_Q, \prec_Q, \#_Q, l_Q) = [\![Q]\!]$.

Define $E_1^a = \{e_1 \in E_1 \mid l_1(e_1) = a\}$ and $E_1^{-a} = \{e_1 \in E_1 \mid l_1(e_1) \neq a\}$, E_2^a and E_2^{-a} likewise.

First of all we state general presuppositions: Let $Q' = a_1; Q; a_2$, then
$[\![Q']\!] = [\![a_1; Q; a_2]\!] = (E_{Q'}, \prec_{Q'}, \#_{Q'}, l_{Q'})$ with
$E_{Q'} = E_Q \cup \{q_1, q_2\}$, (with $q_1, q_2 \notin L(Q)$)
$\prec_{Q'} = \prec_Q \cup \{q_1\} \times E_Q \cup E_Q \times \{q_2\} \cup \{(q_1, q_2)\}$,
$\#_{Q'} = \#_Q$,
$l_{Q'} = l_Q \cup \{(q_1, a_1), (q_2, a_2)\})$

A.1.1 Semantic Refinement

$[\![P]\!] = [\![P_1]\!] \parallel_A [\![P_2]\!] = (E_P, \prec_P, \#_P, l_P)$ with

$$\begin{array}{ll}
E_P = & (E_1 \times \{\star\}) \cup (\{\star\} \times E_2) \cup \{(e_1, e_2) \in E_1 \times E_2 \mid l_1(e_1) = l_2(e_2) \in A\}, \\
\prec_P = & \{((e_1, e_2), (e_1', e_2')) \mid (e_1, e_1') \in \prec_1 \vee (e_2, e_2') \in \prec_2\} \\
\#_P = & \{((e_1, e_2), (e_1', e_2')) \mid (e_1 \#_1 e_1') \vee\ (e_2 \#_2 e_2') \vee\ (e_1 = e_1' \neq \star\ \&\ e_2 \neq e_2') \vee \\
& (e_2 = e_2' \neq \star\ \&\ e_1 \neq e_1')\} \cup \\
& \{((*, e_2), (*, e_2)) \mid l_2(e_2) \in A\} \cup \{((e_1, *), (e_1, *)) \mid l_1(e_1) \in A\} \\
l_P = & \{((e_1, *), l_1(e_1)) \mid e_1 \in E_1\} \cup \{((e_1, e_2), l_2(e_2)) \mid e_2 \neq *\}
\end{array}$$

Then let $P_R := P[a \rightsquigarrow Q'] = (P_1 \parallel_A P_2)[a \rightsquigarrow a_1; Q; a_2]$

$[\![P_R]\!] = (E_R, \prec_R, \#_R, l_R)$ with

$$\begin{array}{ll}
E_R = & X_0 \cup X_1 \cup X_2 \cup X_3 \cup X_4 \cup X_5 \cup X_6 \cup X_7 \cup X_8 \text{ with} \\
& X_0 = \{((e_1, *), *) \mid e_1 \in E_1^{-a}, l_1(e_1) \in A\} \\
& X_1 = \{((e_1, *), *) \mid e_1 \in E_1^{-a}, l_1(e_1) \notin A\} \\
& X_2 = \{((e_1, e_2), q) \mid e_1 \in E_1^a, e_2 \in E_2^a, q \in E_Q\} \\
& X_3 = \{((e_1, *), q') \mid e_1 \in E_1^a, q' \in E_Q \cup \{q_1, q_2\}\} \\
& X_4 = \{((*, e_2), *) \mid e_2 \in E_2^{-a}, l_2(e_2) \in A\} \\
& X_5 = \{((*, e_2), *) \mid e_2 \in E_2^{-a}, l_2(e_2) \notin A\} \\
& X_6 = \{((*, e_2), q') \mid e_2 \in E_2^a, q' \in E_Q \cup \{q_1, q_2\}\} \\
& X_7 = \{((e_1, e_2), *) \mid e_1 \in E_1^{-a}, e_2 \in E_2^{-a}, l_1(e_1) = l_2(e_2) \in A\} \\
& X_8 = \{((e_1, e_2), q_i) \mid e_1 \in E_1^a, e_2 \in E_2^a, i \in \{1, 2\}\}
\end{array}$$

$$\begin{array}{ll}
\prec_R = & \{(((e_1,e_2),q),((e'_1,e'_2),q') \mid (e_1 \prec_1 e'_1)(\text{FRa}) \\
& \vee(e_2 \prec_2 e'_2)\ (\text{FRb}) \\
& \vee(e_1 = e'_1\ \&\ e_2 = e'_2\ \&\ q \prec_{Q'} q')\ (\text{FRc})\ \} \\
\#_R = & \{(((e_1,e_2),q),((e'_1,e'_2),q')) \mid (e_1 \#_1 e'_1)\ (\text{KRa}) \\
& \vee(e_2 \#_2 e'_2)\ (\text{KRb}) \\
& \vee(e_1 = e'_1 \neq \star\ \&\ e_2 \neq e'_2)\ (\text{KRc}) \\
& \vee(e_2 = e'_2 \neq \star\ \&\ e_1 \neq e'_1)\ (\text{KRd}) \\
& \vee(e_1 = e'_1 = \star\ \&\ e_2 = e'_2\ \&\ l_2(e_2) \in A)\ (\text{KRe}) \\
& \vee(e_2 = e'_2 = \star\ \&\ e_1 = e'_1\ \&\ l_1(e_1) \in A)\ (\text{KRf}) \\
& \vee(e_1 = e'_1\ \&\ e_2 = e'_2\ \&\ q \#_Q q')\ (\text{KRg})\}
\end{array}$$

$$l_R((e_1,e_2),q) = \begin{cases} l_Q(q) & \text{if } q \in E_Q \\ a_1 & \text{if } q = q_1 \\ a_2 & \text{if } q = q_2 \\ l_2(e_2) & \text{if } e_1 = * \ \&\ e_2 \in E_2^{-a} \\ l_1(e_1) & \text{otherwise} \end{cases}$$

The sets X_0, X_3, X_4, X_6 only contain self-conflicting events. The other sets do not contain any self-conflicting events.

A.1.2 Syntactic Refinement

Let $P'_1 = P_1[a \leadsto a_1; Q; a_2]$ and $P'_2 = P_2[a \leadsto a_1; a_2]$.

Then $[\![P'_1]\!] = (E'_1, \prec'_1, \#'_1, l'_1), [\![P'_2]\!] = (E'_2, \prec'_2, \#'_2, l'_2)$ with:

$$\begin{array}{ll}
E'_1 = & \{(e,*) \mid e \in E_1^{-a}\} \cup \{(e,q) \mid e \in E_1^a, q \in E_Q \cup \{q_1,q_2\}\} \\
\prec'_1 = & \{((e,q),(e',q')) \mid (e \prec_1 e') \vee (e = e'\ \&\ (q \prec_{Q'} q'))\} \\
\#'_1 = & \{((e,q),(e',q')) \mid (e \#_1 e') \vee (e = e'\ \&\ q \#_Q q')\}
\end{array}$$

$$l'_1((e,q)) = \begin{cases} l_1(e) & \text{if } e \in E_1^{-a} \\ a_1 & \text{if } e \in E_1^a\ \&\ q = q_1 \\ a_2 & \text{if } e \in E_1^a\ \&\ q = q_2 \\ l_Q(q) & \text{if } e \in E_1^a\ \&\ q \in E_Q \end{cases}$$

and $[\![P'_2]\!] = (E'_2, \prec'_2, \#'_2, l'_2)$ with

$$\begin{array}{ll}
E'_2 = & \{(e,*) \mid e \in E_2^{-a}\} \cup \{(e,q) \mid e \in E_2^a, q \in \{q_1,q_2\}\} \\
\prec'_2 = & \{((e,q),(e',q')) \mid (e \prec_2 e') \vee (e = e'\ \&\ q = q_1 \& q' = q_2)\} \\
\#'_2 = & \{((e,q),(e',q')) \mid e \# e'\}
\end{array}$$

$$l'_2((e,q)) = \begin{cases} l_2(e) & \text{if } e \in E_2^{-a} \\ a_1 & \text{if } e \in E_2^a, q = q_1 \\ a_2 & \text{if } e \in E_2^a, q = q_2 \end{cases}$$

Let $P_S := P'_1 \parallel_{A \setminus \{a\} \cup \{a_1,a_2\}} P'_2$. Then $[\![P_S]\!] = (E_S, \prec_S, \#_S, l_S)$ with

$E_S = \{((e_1,*),*) \mid e_1 \in E_1^{-a}\}$
$\cup\{((e_1,q),*) \mid e_1 \in E_1^a, q \in E_Q \cup \{q_1,q_2\}\}$
$\cup\{(*,(e_2,*)) \mid e_2 \in E_2^{-a}\}$
$\cup\{(*,(e_2,q)) \mid e_2 \in E_2^a, q \in \{q_1,q_2\}$
$\cup\{((e_1,*),(e_2,*)) \mid e_1 \in E_1^{-a}, e_2 \in E_2^{-a}, l_1(e_1) = l_2(e_2) \in A\}$
$\cup\{((e_1,q_1),(e_2,q_1)) \mid e_1 \in E_1^a, e_2 \in E_2^a\}$
$\cup\{((e_1,q_2),(e_2,q_2)) \mid e_1 \in E_1^a, e_2 \in E_2^a\}$
Rename the events:

$$
\begin{array}{ll}
E_S = & Y_0 \cup Y_1 \cup Y_2 \cup Y_3 \cup Y_4 \cup Y_5 \cup Y_6 \cup Y_7 \cup Y_8, \text{ where} \\
& Y_0 = \{((e_1,*),*) \mid e_1 \in E_1^{-a}, l_1(e_1) \in A\} \\
& Y_1 = \{((e_1,*),*) \mid e_1 \in E_1^{-a}, l_1(e_1) \notin A\} \\
& Y_2 = \{((e_1,*),q) \mid e_1 \in E_1^{a}, q \in E_Q\} \\
& Y_3 = \{((e_1,*),q_i) \mid e_1 \in E_1^{a}, i \in \{1,2\}\} \\
& Y_4 = \{((*,e_2),*) \mid e_2 \in E_2^{-a}, l_2(e_2) \in A\} \\
& Y_5 = \{((*,e_2),*) \mid e_2 \in E_2^{-a}, l_2(e_2) \notin A\} \\
& Y_6 = \{((*,e_2),q_i) \mid e_2 \in E_2^{a}, i \in \{1,2\} \\
& Y_7 = \{((e_1,e_2),*) \mid e_1 \in E_1^{-a}, e_2 \in E_2^{-a}, l_1(e_1) = l_2(e_2) \in A\} \\
& Y_8 = \{((e_1,e_2),q_i) \mid e_1 \in E_1^{a}, e_2 \in E_2^{a}, i \in \{1,2\}\} \\
\prec_S = & \{(((e_1,e_2),q),((e_1',e_2'),q')) \mid (e_1 \prec_1 e_1') (\text{FSa}) \\
& \vee (e_2 \prec_2 e_2') \text{ (FSb)} \\
& \vee (e_1 = e_1' \neq \star \ \& \ (q \prec_{Q'} q') \text{ (FSc)} \\
& \vee (e_2 = e_2' \neq \star \ \& \ q = q_1, q' = q_2) \text{ (FSd) } \} \\
\#_S = & \{(((e_1,e_2),q),((e_1',e_2'),q')) \mid (e_1 \#_1 e_1') \text{ (KSa)} \\
& \vee (e_2 \#_2 e_2') \text{ (KSb)} \\
& \vee (e_1 = e_1' \neq * \ \& \ q = q' \ \& \ e_2 \neq e_2') \text{ (KSc)} \\
& \vee (e_2 = e_2' \neq * \ \& \ q = q' \ \& \ e_1 \neq e_1') \text{ (KSd)} \\
& \vee (e_1 = e_1' = * \ \& \ q = q' \ \& \ e_2 = e_2' \ \& \ l_2(e_2) \in A) \text{ (KSe)} \\
& \vee (e_2 = e_2' = * \ \& \ q = q' \ \& \ e_1 = e_1' \ \& \ (l_1(e_1) \in A \setminus \{a\} \vee q, q' \in \{q_1, q_2\})) \text{ (KSf)} \\
& \vee (e_1 = e_1' \neq * \ \& \ q \#_Q q') \text{ (KSg)}\}
\end{array}
$$

$$
l_S(((e_1,e_2),q)) = \begin{cases} l_Q(q) & \text{if } q \in E_Q \\ a_1 & \text{if } q = q_1 \\ a_2 & \text{if } q = q_2 \\ l_2(e_2) & \text{if } e_1 = * \ \& \ e_2 \in E_1^{-a} \\ l_1(e_1) & \text{otherwise} \end{cases}
$$

The sets Y_0, Y_3, Y_4, Y_6 only contain self-conflicting events, the other sets do not contain any self-conflicting event.

A.2 Comparison of $\mathcal{E}_R$ and $\mathcal{E}_S$

As one can easily see the sets X_0 and Y_0, X_1 and Y_1, X_3 and Y_3, X_4 and Y_4, X_5 and Y_5, X_6 and Y_6, X_7 and Y_7 and X_8 and Y_8 correspond to each other. Even the following identities hold: $X_0 = Y_0$, $X_1 = Y_1$, $X_4 = Y_4$, $X_5 = Y_5$, $X_7 = Y_7$, $X_8 = Y_8$, (instead of $=$ one could use $\cong$ (set-isomorphism) if the events were renamed).

The sets X_3 and X_6 contain more events than Y_3 and Y_6.

The main difference lies between X_2 and Y_2: X_2 can contain much more events than Y_2 since any combinations between events labelled with a in E_1 and E_2 are allowed. Syntactic refinement does not lead to any combination with events of E_2.

Let $C_R := \mathit{Conf}(\mathcal{E}_R)$ and $C_S := \mathit{Conf}(\mathcal{E}_S)$ (and $E_{C_R} = \bigcup_{X \in C_R} X, E_{C_S} = \bigcup_{X \in C_S} X$). In order to show that $\mathcal{E}_R \cong_d \mathcal{E}_S$, one has to find a bijection $f : C_R \to C_S$ such that
$\forall X, Y \in C_R : X \subseteq Y \Rightarrow f(X) \subseteq f(Y)$.

First of all define f on E_R and then lift f to sets. Note that $f : E_R \to E_S$ is not bijective.

Let $x = ((e_1, e_2), q) \in E_R$.

$$\text{Then } f(x) := \begin{cases} x & \text{if } x \in X_0 \cup X_1 \cup X_4 \cup X_5 \cup X_7 \cup X_8 \\ x & \text{if } x \in X_3 \cup X_6 \ \& \ q \in \{q_1, q_2\} \\ ((e_1, e_2), q_1) & \text{if } x \in X_3 \cup X_6 \ \& \ q \in E_Q \\ ((e_1, *, q) & \text{if } x \in X_2 \end{cases}$$

Note that f is even defined on events that are not contained in E_{C_R}. As you see f is a well-defined surjective mapping from E_{C_R} to E_{C_S}.

We want to show that $f : C_R \to C_S$ is an isomorphism of domains. if a is not auto-concurrent in $P_1 \parallel_A P_2$. We will show the following:

- $f : C_R \to C_S$ (i.e. all images of f are configurations of $\mathcal{E}_S$: $\forall X \in C_R : f(X) \in C_S$)
- $\forall x \in E_{C_R} : l_R(x) = l_S(f(x))$
- $\forall X, Y \in C_R : X \subseteq Y \Leftrightarrow f(X) \subseteq f(Y)$
- If a is not auto-concurrent in $P_1 \parallel_A P_2$ then $f : C_R \to C_S$ is bijective.

A.3 $\forall X \in C_R : f(X) \in C_S$

Let $y \in E_S$. Then $f^{-1}(y) = \{x \in E_R \mid f(x) = y\}$. We have $\forall 0 \leq i \leq 8 : x \in X_i \Leftrightarrow f(x) \in Y_i$ and $y \in Y_i \Leftrightarrow f^{-1}(y) \subseteq X_i$.

Let X be a configuration of $\mathcal{E}_R$, i.e. X satisfies the conditions (i),(ii),(iii), (iv): Then $f(X)$ also satisfies these conditions:

(i) $\forall y, y' \in f(X) : \neg(y \#_S y')$

(ii) $\leq_{f(X)} = (\prec \cap (f(X) \times f(X)))^*$ is a partial order

(iii) $\forall y \in f(X)$: $\{y' \in f(X) \mid y' \leq_{f(X)} y\}$ is finite

(iv) $\forall y \in f(X), \forall y' \in E_S \setminus f(X)$: $y' \prec_S y \Rightarrow \exists y'' \in f(X) : y' \#_S y'' \prec_S y$.

A.3.1 (i)

Suppose $\exists y = ((e_1, e_2), q), y' = ((e_1', e_2'), q') \in f(X)$ such that $y \#_S y'$. Note that $\forall y, y' \in f(X) : \exists x \in f^{-1}(y), x' \in f^{-1}(y')$ with $x, x' \in X$. One of the following must hold:

$(e_1 \#_1 e_1')$ (KSa)
$\Rightarrow \forall x \in f^{-1}(y) \forall x' \in f^{-1}(y') : x \#_R x'$ (KRa)
Therefore $\exists x, x' \in X$ with $x \#_R x'$. Contradiction!

or $(e_2 \#_2 e_2')$ (KSb)
$\Rightarrow \forall x = ((e_1, e_2), q_x) \in f^{-1}(y) \forall x' = ((e_1', e_2'), q_x') \in f^{-1}(y') : x \#_R x'$ (KRb)
Therefore $\exists x, x' \in X$ with $x \#_R x'$. Contradiction!

or $(e_1 = e'_1 \neq * \ \&\ q = q' \ \&\ e_2 \neq e'_2)$ (KSc)

Then $e_2 \neq * \vee e'_2 \neq *$. Let w.l.o.g. $e_2 \neq * \Rightarrow e'_2 \neq *$ (otherwise $q \neq q'$)
$\Rightarrow \forall x \in f^{-1}(y) : x = ((e_1, e_2), q) \forall x' \in f^{-1}(y') : x' = ((e_1, e'_2), q)$ and with (KRc) $x \#_R x'$.

Therefore $\exists x, x' \in X$ with $x \#_R x'$. Contradiction!

or $(e_2 = e'_2 \neq * \ \&\ q = q' \ \&\ e_1 \neq e'_1)$ (KSd)

$\Rightarrow \forall x \in f^{-1}(y) \exists q_x : x = ((e_1, e_2), q_x) \forall x' \in f^{-1}(y') \exists q'_x : x' = ((e'_1, e_2), q'_x)$ and with (KRd) $x \#_R x'$

Also $\exists x, x' \in X$ with $x \#_R x'$. Contradiction!

or $(e_1 = e'_1 = * \ \&\ q = q' \ \&\ e_2 = e'_2 \ \&\ l_2(e_2) \in A)$ (KSe)

$\Rightarrow y = y', \forall x \in f^{-1}(y) \exists q_x : x = ((*, e_2), q_x)$. With (KRe) follows $x \#_R x$.

Therefore $\exists x \in X$ with $x \#_R x$. Contradiction!

or $(e_2 = e'_2 = * \ \&\ q = q' \ \&\ e_1 = e'_1 \ \&\ (l_1(e_1) \in A \setminus \{a\} \vee q, q' \in \{q_1, q_2\}))$ (KSf)

$\Rightarrow y = y', \forall x \in f^{-1}(y) \exists q_x : x = ((e_1, *), q_x)$ with $l_1(e_1) \in A$. With (KRf) follows $x \#_R x$

Therefore $\exists x \in X$ with $x \#_R x$. Contradiction!

or $(e_1 = e'_1 \neq * \ \&\ q \#_Q q')$ (KSg)

$\Rightarrow y, y' \in Y_2, \forall x \in f^{-1}(y) \exists e_{2x} \in E_2 : x = ((e_1, e_{2x}), q), \forall x' \in f^{-1}(y') \exists e'_{2x} \in E_2 : x' = ((e_1, e'_{2x}), q)$ If $e_{2x} = e'_{2x}$, then with (KRg) $x \#_R x'$.
If $e_{2x} \neq e'_{2x}$, then follows with (KRc) $x \#_R x'$

Therefore $\exists x, x' \in X$ with $x \#_R x'$. Contradiction!

A.3.2 (ii)

We have to show that $\leq_{f(X)} = (\prec \cap (f(X) \times f(X)))^*$ is a partial order. It suffices to show that $\forall y, y' \in f(X) : (y \leq_{f(X)} y' \ \&\ y' \leq_{f(X)} y) \Rightarrow y = y'$. For this it's enough to show that no sequence $y = y_1 \prec_S ... \prec_S y_n = y \in f(X)$ with $n \geq 2$ exists.

Suppose there exists a sequence $y_1, ..., y_n$, $n \geq 2$. Consider the set $\{x_1, .., x_n\} \subseteq X$ with $f(x_1) = y_1, ..., f(x_n) = y_n$. If $\forall x \in X : f(x) \prec_S f(x') \Rightarrow x \prec_R x'$, then $x_1 \prec_R ... \prec_R x_n$ would also be a cycle and therefore X would be no configuration. Contradiction!

So we will show for all events $x, x' \in X$ that $f(x) \prec_S f(x') \Rightarrow x \prec_R x'$. Since x is self-conflicting iff $f(x)$ is self-conflicting one does not have to consider $y, y' \in Y_0 \cup Y_3 \cup Y_4 \cup Y_6$. Let $x, x' \in X$ with $x = ((e_1, e_{2x}, q_x), x' = ((e'_1, e'_{2x}, q'_x)$, and $y = f(x) = ((e_1, e_2), q), y' = f(x') = ((e'_1, e'_2), q')$. Then $f(x) \prec_S f(x')$ iff

$e_1 \prec_1 e'_1$

$\Rightarrow x \prec_R x'$ (FRa)

or $e_2 \prec_1 e'_2$

$\Rightarrow (e_{2x} = e_2 \ \&\ e'_{2x} = e'_2) \Rightarrow (x \prec_R x')$ (FRb) or $e_1 = e'_1 \neq * \;\&\; q \prec_{Q'} q'$

Case 1: $y \in Y_2$, d.h. $y = ((e_1, *), q)$ with $q \in E_Q$ ($\Rightarrow x = ((e_1, e_{2x}), q)$ with $e_{2x} \in E_2^a$)

If $y' \in Y_2$ then $x' = ((e_1, e'_{2x}), q')$ with $e'_{2x} \in E_2^a$. If $e_{2x} \neq e'_{2x}$ then $x \#_R x'$ (KRc) and therefore $x, x' \notin X$. If $e_{2x} = e'_{2x}$ then $x \prec_R x'$ (FRc).

If $y' \in Y_8$ then $y' = x' = ((e_1, e'_{2x}, q_2)$. In analogy to this reasoning we conclude $e_{2x} = e'_{2x}$ and therefore $x \prec_R x'$ (FRc)

(If $y' \in Y_3$ then $x' = (e'_1, *, q_2)$ and therefore $\neg(x \prec_R x')$ – thus the assumption is not true for arbitrary events)

Case 2:$y \in Y_8$, i.e. $y = ((e_1, e_2), q_1) = x$

If $y' \in Y_2$ with $y' = ((e_1, *), q'), q' \in E_Q$ then $x' = ((e_1, e'_{2x}, q')$ and therefore $e_2 = e'_{2x}$ and $x \prec_R x'$ (FRc)

If $y' \in Y_8$ with $y' = ((e_1, e'_2), q_2) = x'$ then with the same reasoning $x \prec_R x'$ (FRc).

or $e_2 = e'_2 \neq * \;\&\; q = q_1, q' = q_2$

$\Rightarrow (y, y' \in Y_8) \Rightarrow (x = y, x' = y') \Rightarrow (x \prec_R x')$ (FRc)

A.3.3 (iii)

$\forall y \in f(X)$ ist $\{y' \in f(X) \mid y' \leq_{f(X)} y\}$ is finite since only finite event structures are taken into account.

A.3.4 (iv)

We show that $\forall y \in f(X), \forall y' \in E_S \setminus f(X)$: $y' \prec_S y \Rightarrow \exists y'' \in f(X)$ with $y' \#_S y'' \prec_S y$.

Let $y = ((e_1, e_{2y}), q_y) \in f(X), y' = ((e'_1, e'_{2y}), q'_y) \in E_S \setminus f(X), y' \prec_S y$. Let $f^{-1}(y) = \{x \in E_R \mid f(x) = y\}$, $f^{-1}(y') = \{x' \in E_R \mid f(x') = y\}$. Four cases have to be examined:
FSa) $\mathbf{e'_1 \prec_1 e_1}$
FSb) $\mathbf{e'_{2y} \prec_2 e_{2y}}$
FSc) $\mathbf{e'_1 = e_1 \neq * \;\&\; q'_y \prec_{Q'} q_y}$
FSd) $\mathbf{e'_{2y} = e_{2y} \neq * \;\&\; q'_y = q_1 \;\&\; q_y = q_2}$

- $\mathbf{e'_1 \prec_1 e_1}$ (FSa)

  $\forall x' \in f^{-1}(y') : x' \notin X$ and $\forall x' \in f^{-1}(y') \forall x \in f^{-1}(y) : x' \prec_R x$. Let $x' = ((e'_1, e'_2), q') \in f^{-1}(y')$ and $x \in f^{-1}(y) \cap X$. Then $x' \in X_0 \cup X_1 \cup X_2 \cup X_3 \cup X_7 \cup X_8$ holds.
  If $x' \in X_2$ choose $\bar{x}' = ((e'_1, *), q') \in X_3$.
  If $x' \in X_7$ choose $\bar{x}' = ((e'_1, *), *) \in X_0$.
  If $x' \in X_8$, choose $\bar{x}' = ((e'_1, *), q') \in X_3$.
  Otherwise choose $\bar{x}' = x'$. Then $\bar{x}' = ((e'_1, *), q') \in X_0 \cup X_1 \cup X_3$

  Now $\bar{x}' \notin X$ holds and $\bar{x}' \prec_R x$. Then $\exists x'' = ((e''_1, e''_2), q'') \in X, \bar{x}' \#_R x'' \prec_R x$, and so:
  (KRa) $e''_1 \#_1 e'_1$ or
  (KRb) (impossible) or

(KRc) $e_1'' = e_1' \neq * \;\&\; e_2'' \neq *$ or
(KRd) (impossible) or
(KRe) (impossible) or
(KRf) (impossible) or
(KRg) $e_1'' = e_1' \;\&\; e_2'' = * \;\&\; q'' \#_Q q'$

- $e_1'' \#_1 e_1'$ (KRa)
  With lemma A-2a there exists $\tilde{x} = ((\tilde{e}_1, \tilde{e}_2), \tilde{q}) \in X$ with $\tilde{e}_1 \#_1 e_1' \;\&\; \tilde{e}_1 \prec_1 e_1$. Then we conclude $f(\tilde{x}) \prec_S f(x) \;\&\; f(\tilde{x}) \#_s f(\bar{x}) \;\&\; f(\tilde{x}) \#_s f(x')$
- $e_1'' = e_1' \neq * \;\&\; e_2'' \neq *$ (KRc)
  Obviously $x'' = ((e_1', e_2''), q'')$ and $f(x'') \prec_S f(x)$. If $e_2'' \in E_2^a$ (this is the case iff $q'' \neq *$) we conclude with lemma A-1a and A-1b that $x_1 = ((e_1', e_2''), q_1) \in X$ and $x_2 = ((e_1', e_2''), q_2) \in X$. And therefore $f(x_1) \prec_S f(x)$, $f(x_2) \prec_S f(x)$.
  Case 1 $\bar{x}' = x' \in X_0 \cup X_1$, therefore $x' = ((e_1', *), *), e_1' \in E_1^{-a}$, and $f(x') = x'$. Since $e_1' = e_1'' \in E_1^{-a}$, $q'' = *$ follows, hence $x'' = ((e_1', e_2''), *) \in X_7$ and $f(x'') = x''$. With (KSc) $f(x') \#_S f(x'')$ follows. So choose $y'' = f(x'')$.
  Case 2 $\bar{x}' \in X_3, x' \in X_2$
  Consequently $e_1' \in E_1^a$ and $e_2'' \in E_2^a$. Suppose $\exists \tilde{e}_2 \in E_2 : ((e_1', \tilde{e}_2), q') \in X$ Then $\tilde{e}_2 = e_2''$ holds, since otherwise $((e_1', \tilde{e}_2), q') \#_R x'' = ((e_1', e_2''), q'')$ with (KRc)). Thus $((e_1', e_2''), q') \in X$. Since $x' \in X_2$ $q' \in E_Q$ and therefore $((e_1', \tilde{e}_2), q') \in X_2$. But then $f(x') = f(((e_1', \tilde{e}_2), q'))$. Contradiction!
  Thus $\forall \tilde{e}_2 \in E_2 : ((e_1', \tilde{e}_2), q') \notin X$. With lemma A-1c one concludes $\exists \tilde{x} = ((e_1', e_2''), \tilde{q}) \in X$ such that $\tilde{q} \#_Q q'$. Then $f(\tilde{x}) = ((e_1', *), \tilde{q}) \prec_S f(x)$ and $f(x') = ((e_1', *), q')$ and therefore $f(x') \#_S f(\tilde{x})$ (KSg). So choose $y'' = f(\tilde{x})$.
  Case 3 $\bar{x}' = x' \in X_3$, thus $x' = ((e_1', *), q')$ with $e_1' \in E_1^a$. And $x'' = ((e_1', e_2''), q'') \in X_2 \cup X_8$. $f(x') = ((e_1, *), q_1) \vee f(x') = ((e_1, *), q_2)$. $e_2'' \in E_2^a \Rightarrow x_1, x_2 \in X$. Therefore either $f(x') \#_S f(x_1)$ or $f(x') \#_s f(x_2)$. Depending on x' choose $y'' = f(x_1)$ or $y'' = f(x_2)$.
  Case 4 $\bar{x}' \in X_0, x' \in X_7$. Then $f(x') = x' = ((e_1', e_2'), *)$, $e_1' \in E_1^{-a}$ and thus $f(x'') = x'' = ((e_1', e_2''), *) \in X_7$.
  $e_2'' \neq e_2'$, (otherwise $x' = x''$), consequently with (KSc) $f(x'') \#_S f(x')$. Choose $y'' = f(x'')$.
  Case 5 $\bar{x}' \in X_3, x' \in X_8$. Then $f(x') = x' = ((e_1', e_2'), q')$ with $e_1' \in E_1^a, q' \in \{q_1, q_2\}$ and $x'' = ((e_1', e_2''), q'') \in X_2 \cup X_8$. $e_2'' \in E_2^a \Rightarrow x_1, x_2 \in X$. If $e_2'' = e_2'$, then $x' = x_1 \vee x' = x_2$ – this is impossible and therefore $e_2'' \neq e_2'$ and consequently $f(x') \#_S f(x_1) \vee f(x') \#_S f(x_2)$. Depending on q' choose $y'' = f(x_1)$ or $y'' = f(x_2)$.
- $e_1'' = e_1' \;\&\; e_2'' = * \;\&\; q'' \#_Q q'$ (KRg)
  In this case $x'' \in X_3$ and thus $x'' \#_R x''$ and $x'' \notin X$. Contradiction!

- $\mathbf{e'_{2y} \prec_2 e_{2y}}$ (FSb)
  $\forall x' \in f^{-1}(y') : x' \notin X$ and $\forall x' \in f^{-1}(y') \forall x \in f^{-1}(y) : x' \prec_R x$. Let $x' = ((e_1', e_2'), q') \in f^{-1}(y')$ and $x \in f^{-1}(y) \cap X$. Then $x' \in X_4 \cup X_5 \cup X_6 \cup X_7 \cup X_8$ holds.
  If $x' \in X_7$, choose $\bar{x}' = ((*, e_2'), *) \in X_4$.
  If $x' \in X_8$, choose $\bar{x}' = ((*, e_2'), q') \in X_6$.

Otherwise choose $\bar{x}' = x'$. Then $\bar{x}' = ((*, e_2'), q') \in X_4 \cup X_5 \cup X_6$ holds.

But $\bar{x}' \notin X$ and $\bar{x}' \prec_R x$. Therefore $\exists x'' = ((e_1'', e_2''), q'') \in X, \bar{x}' \#_R x'' \prec_R x$, thus:
(KRa) (impossible) or
(KRb) $e_2'' \#_2 e_2'$ or
(KRc) (impossible) or
(KRd) $e_2'' = e_2' \neq * \;\&\; e_1'' \neq *$ or
(KRe) (impossible) or
(KRf) (impossible) or
(KRg) $e_1'' = * \;\&\; e_2'' = e_2' \;\&\; q'' \#_Q q'$

- $e_2'' \#_2 e_2'$ (KRb)
  With lemma A-2b there exists $\tilde{x} = ((\tilde{e}_1, \tilde{e}_2), \tilde{q}) \in X$ with $\tilde{e}_2 \#_2 e_2' \;\&\; \tilde{e}_2 \prec_2 e_2$. Then $f(\tilde{x}) \prec_S f(x) \;\&\; f(\tilde{x}) \#_s f(\bar{x}) \;\&\; f(\tilde{x}) \#_s f(x')$ holds. Choose $y'' = f(\tilde{x})$.
- $e_2'' = e_2' \neq * \;\&\; e_1'' \neq *$ (KRd)
  In any case $f(x'') \prec_S f(x)$ and $l_2(e_2') \in A$ and
  $x' = ((e_1', e_2'), q') \in X_4 \cup x_6 \cup X_7 \cup X_8$.
  Case 1 $q' = *$, also $x' = ((e_1', e_2'), *) \in X_4 \cup X_7 \Rightarrow e_2' \in E_2^{-a}$,
  $x'' = ((e_1'', e_2'), *) \in X_7 \Rightarrow f(x') = x', f(x'') = x''$ and with (KSd) $f(x'') \#_S f(x')$.
  Thus choose $y'' = f(x'')$.
  Case 2 $q' \in E_Q \cup \{q_1, q_2\}$, also $x' \in X_6 \cup X_8$ and therefore
  $$f(x') = \begin{cases} ((e_1', e_2'), q_1) & \text{if } q \in E_Q \cup \{q_1\} \\ ((e_1', e_2'), q_2) & \text{otherwise} \end{cases}$$
  Moreover $e_2' \in E_2^a \Rightarrow x'' = ((e_1'', e_2'), q'') \in X_8$ With lemma A-1a and A-1b we conclude $x_1 = f(x_1) = ((e_1'', e_2'), q_1) \in X$ and $x_2 = f(x_2) = ((e_1'', e_2'), q_2) \in X$ and $f(x_1), f(x_2) \prec_S f(x)$.
  With (KSd): $f(x') \#_S f(x_1) \vee f(x') \#_S f(x_2)$. Depending on q' choose $y'' = f(x_1)$ or $y'' = f(x_2)$.
- $e_1'' = * \;\&\; e_2'' = e_2' \;\&\; q'' \#_Q q'$ (KRg)
  In this case x'' would be in X_6 and therefore $x'' \#_R x''$ and thus $x'' \notin X$.
  Contradiction!

- $\mathbf{e_1' = e_1 \neq * \;\&\; q_y' \prec_{Q'} q_y}$ (FSc)

In this case $e_1 \in E_1^a$ and therefore $y \in Y_2 \cup Y_8$ and $y' \in Y_2 \cup Y_3 \cup Y_8$. Then there exists $e_2 \in E_2^a$ with $x = ((e_1, e_2), q) \in X$ and $f(x) = f(((e_1, e_2), q)) = y$, so that
$x \in X \cap f^{-1}(y)$. With lemma A-1a conclude that $x_1 = ((e_1, e_2), q_1) \in X$ and therefore $f(x_1) = ((e_1, e_2), q_1) \in f(X)$ and with (FSc) $f(x_1) \prec_S y$.

Case 1: $y' \in Y_3 \Rightarrow y' = ((e_1, *), q_1)$
With (KSc): $f(x_1) \#_S y'$. Choose $y'' = f(x_1)$

Case 2: $y' \in Y_8 \Rightarrow y' = ((e_1, e_{2y}'), q_1)$

Since $y' \notin f(X)$ and $f(x_1) \in f(X)$ we conclude: $e_{2y}' \neq e_2$ and with (KSc) $f(x_1) \#_S y'$.
Thus choose $y'' = f(x_1)$.

Case 3: $y' \in Y_2 \Rightarrow y' = ((e_1, *), q')$
Then $f^{-1}(y') = \{((e_1, e_2'), q') \mid e_2' \in E_2^a\}$. Therefore $x' = ((e_1, e_2), q') \in f^{-1}(y')$ and
$x' \prec_R x$.
There exists $x'' \in X$ with $x'' \prec_R x, \neg(x'' \#_R x), x'' \#_R x'$. This is only possible with (KRg), thus $x'' = ((e_1, e_2), q'')$ for some $q'' \in E_Q$ with $q'' \#_Q q'$. In order that $x'' \prec_R x$,

$q'' \prec_{Q'} q$ must hold. With this we conclude $x'' \in X_2$ and $f(x'') = ((e_1, *), q'')$ with $f(x'') \#_S y'$ (KSg) and $f(x'') \prec_S y$ with (FSc). Choose $y'' = f(x'')$

- $\mathbf{e'_{2y} = e_{2y} \neq * \ \& \ q'_y = q_1 \ \& \ q_y = q_2}$ (FSd)

  So $y \in Y_8$ and $y' \in Y_6 \cup Y_8$. Therefore $x = ((e_1, e_{2y}, q_2) \in f^{-1}(y) \cap X$ and with lemma A-1a $x_1 = ((e_1, e_{2y}), q_1) \in X$ and $f(x_1) = ((e_1, e_{2y}), q_1) \in f(X)$. With (FSd) $f(x_1) \prec_S y$.

  Case 1: $y' \in Y_6 \Rightarrow y' = ((*, e_{2y}), q_1)$
  With (KSd) $y' \#_S f(x_1)$, thus choose $y'' = f(x_1)$.

  Case 2: $y' \in Y_8 \Rightarrow y' = ((e'_1, e_{2y}), q_1)$
  Since $y' \notin f(X)$ and $f(x_1) \in f(X)$ we conclude $e'_1 \neq e_1$. With (KSd) $f(x_1) \#_S y'$, thus choose $y'' = f(x_1)$.

A.4 $\forall x \in E_{C_R} : l_R(x) = l_S(f(x))$

Let $x = ((e_1, e_2), q) \in C_R$. Since $f(x) = x$, if $x \notin X_2$, $f(x) = ((e_1, e'_2), q)$ follows. Therefore:

$$\begin{array}{llll}
l_R(x) = & l_Q(q) & = l_S(f(x)) & \text{if } q \in E_Q \\
l_R(x) = & a_1 & = l_S(f(x)) & \text{if } q = q_1 \\
l_R(x) = & a_2 & = l_S(f(x)) & \text{if } q = q_2 \\
l_R(x) = & l_2(e_2) & = l_S(f(x)) & \text{if } e_1 = * \ \& \ e_2 \in E_2^{-a}\text{, therefore } e'_2 = e_2 \\
l_R(x) = & l_1(e_1) & = l_S(f(x)) & \text{otherwise}
\end{array}$$

A.5 $\forall X, Y \in C_R : X \subseteq Y \Leftrightarrow f(X) \subseteq f(Y)$

$X \subseteq Y \Rightarrow f(X) \subseteq f(Y)$ evident.

Let $f(X) \subseteq f(Y)$. Suppose $X \not\subseteq Y$, i.e. $\exists x \in X : x \notin Y$, but $f(x) \in f(Y)$. Then $\exists y \in Y$ with $f(x) = f(y)$, but $x \neq y$.

Since $\forall x \in C_R : x \neq f(x) \Rightarrow x \in X_2$, it follows that $x = ((e_1, e_2), q) \in X_2$ and therefore $x \neq y$ but as $f(x) = f(y)$ conclude $y = ((e_1, e'_2), q)$. With lemma A-1a $x_1 = ((e_1, e_2), q_1) \in X$ and $y_1 = ((e_1, e'_2), q_1) \in Y$. But then $x_1 \in Y$, Contradiction, since $x_1 \#_R y_1$.

A.6 f bijective if a is not auto-concurrent in $P_1 \parallel_A P_2$

A.6.1 f injective

Let $X, X' \in C_R, X \neq X'$. Then $\exists x \in X : x \notin X' \vee \exists x' \in X' : x' \notin X$. Let w.l.o.g. $x \in X : x \notin X'$.

We show: $\exists y \in f(X) : y \notin f(X')$

$x \in X_1 \cup X_2 \cup X_5 \cup X_7 \cup X_8$, since x is not self-conflicting.

If $x \in X_1 \cup X_5 \cup X_7 \cup X_8$, we have $f(x) = x, f^{-1}(f(x)) = \{x\}$. Thus if $y = f(x) \in X'$, there exists $x'' \in f^{-1}(y) \in X'$, since $f^{-1}(y) = \{x\}$, so $x \in X'$. Contradiction!

Since $x = ((e_1, e_2), q) \in X_2$, we have $f(x) = ((e_1, *), q)$ and $f^{-1}(f(x)) = \{((e_1, e'_2), q) \mid e'_2 \in E_2^a\}$.

With lemma A-1a also $x_1 = ((e_1, e_2), q_1) \in X$. Since $x_1 \in X_8$ we conclude as above that $x \notin X' \Rightarrow f(x) \notin f(X')$, and thus $X \neq X' \Rightarrow f(X) \neq f(X')$.

A.6.2 f surjective

We want to show that $\forall Y \in C_S \exists X \in C_R$ with $f(X) = Y$.

For each $Y \in C_S$ define $g_Y : Y \to C_R$ with
$g_Y(y) = \begin{cases} ((e_1, h(y, Y)), q) & \text{if } y = ((e_1, *), q) \in Y_2 \\ y & \text{otherwise} \end{cases}$
(h from lemma A-3 is used here.)

Define $g : C_S \to C_R$: $g(Y) = \{g_Y(y) \mid y \in Y\}$.

We see

- $\forall Y \in C_S$ g_Y is well-defined and therefore $\forall y \in Y : g_Y(y) \in E_R$, therefore $\forall x \in g(Y) : x \in E_R$.
- $\forall Y \in C_S : \forall x \in g(Y) : g^{-1}(x) = f(x)$ (with f as above)
- $\forall Y \in C_S : \forall x \in g(Y), x \notin X_2 : g^{-1}(x) = x$ and $\forall y \in Y : g_Y^{-1}(g_Y(y)) = y$

If we can show that

- $g : C_S \to C_R$ well-defined
- $f \circ g = id_{C_S}$

then f is surjective. But we can show this only for terms P_1, P_2 with a not being auto-concurrent in $P_1 \parallel_A P_2$.

Let us assume for the rest of this paragraph that a is not auto-concurrent in $P_1 \parallel_A P_2$.

$f \circ g = id_{C_S}$ obvious

$\mathbf{g : C_S \to C_R}$ is well-defined We will show $\forall Y \in C_S : g(Y) \in C_R$. Let $Y \in C_S$.

(i) Suppose $\exists y = ((e_1, e_2), q), y' = ((e'_1, e'_2), q') \in Y : x = ((e_1, e_{2x}), q) = g_Y(y) \#_R x' = ((e'_1, e'_{2x}), q') = g_Y(y')$. With lemma A-3a and A-3b choose
$z = \begin{cases} ((e_1, h(y, Y)), q_1) & \text{if } y \in Y_2 \\ ((e_1, e_2), q_1) & \text{if } q = q_2 \\ y & \text{otherwise} \end{cases}, z' = \begin{cases} ((e'_1, h(y', Y)), q_1) & \text{if } y' \in Y_2 \\ ((e'_1, e'_2), q_1) & \text{if } q' = q_2 \\ y & \text{otherwise} \end{cases}$, Then $z, z' \in Y$.

Let $x \#_R x'$. Because of

KRa) $\Rightarrow y \#_S y'$
KRb) $\Rightarrow z \#_S z'$
KRc) $\Rightarrow z \#_S z'$
KRd) $\Rightarrow z \#_S z'$
KRe) and KRf) impossible and

KRg) $\Rightarrow y \#_S y'$

Thus Y contains a conflict. Contradiction!

(ii) Suppose there exists a sequence $y = y_1, ..., y_n = y \in Y$ with $g_Y(y_1) \prec_R g_Y(y_2) \prec_R ... \prec_R g_Y(y_n)$. Choose the following sequence in Y: $z_1, ..., z_n$ with
$$z_i = \begin{cases} ((e_{1i}, h(y_i, Y)), q_i) & \text{if } y_i = ((e_{1i}, *), q_i) \in Y_2 \\ y_i & \text{otherwise} \end{cases}$$
Then $\forall 1 \leq i \leq n : z_i \prec_S z_{i+1}$ (and $z_1 = z_n$, as KRa) and KRb) lead to KSa) and KSb), and KRc) implies KSc).

Therefore Y is no configuration of $\mathcal{E}_S$. Contradiction!

(iii) obvious

(iv) $g(Y)$ is left-closed up to conflicts, i.e. $\forall x \in g(Y), \forall x' \in E_R \setminus g(Y)$: $x' \prec_R x \Rightarrow \exists x'' \in g(Y)$ with $x' \#_R x'' \prec_R x$.

Let $y \in Y$, $x' \in E_R \setminus g(Y), x' \prec_R g_Y(y)$.

Choose $z' = \begin{cases} ((e_1', e_2'), q_2) & \text{if } x' = ((e_1', e_2'), q') \in X_2 \;\&\; x' \prec_R g_Y(y) \text{ (FRb)} \\ x' & \text{otherwise} \end{cases}$

If $y = ((e_1, *), q) \in Y_2$ we will write y_1 for the unique event $((e_1, e_2), q_1) \in Y$. Then $f(z') \notin Y$ and $f(z') \prec_S y \vee f(z') \prec_S y_1$.

Since Y is a configuration there exists $y'' \in Y$ with $y'' \prec_S y \vee y'' \prec_S y_1$ and $y'' \#_S f(z')$.

With lemma A-4 it follows that $g_Y(y'') \prec_R g_Y(y) \vee g_Y(y'') \prec_R g_Y(y_1)$.

Now we have to show that $g_Y(y'') \#_R x'$. If $y'' \#_S f(z')$ holds because of KSa),KSb),KSc) or KSd) then obviously $g_Y(y'') \#_R x'$. If $y'' \#_S f(z')$ holds because of KSg) (i.e. $e_1'' = e_1 \neq * \;\&\; q'' \#_Q q$) then either $e_2'' = e_2'$ and therefore $g_Y(y'') \#_R x'$ with KRg) or $e_2'' \neq e_2'$ and therefore with KRc) $g_Y(y'') \#_R x'$.

■

A.7 Some lemmata and propositions

$\mathcal{E}_R, \mathcal{E}_1 = [\![P_1]\!], \mathcal{E}_2 = [\![P_2]\!]$ are defined as before.

Proposition A Let $X \in$ *Conf*$(\mathcal{E}_R)$. Then $\Pi_1(X) \in$*Conf*$(\mathcal{E}_1)$ and $\Pi_2(X) \in$*Conf*$(\mathcal{E}_2)$.

Proof:
With Lemma 4.3.1 X':=$\{(e_1, e_2) \mid \exists q : ((e_1, e_2), q) \in X\} \in$*Conf*$([\![P_1]\!] \parallel_A [\![P_2]\!])$. Obviously $\Pi_1(X') = \Pi_1(X) \in$*Conf*$([\![P_1]\!])$ and $\Pi_2(X') = \Pi_2(X) \in$*Conf*$([\![P_2]\!])$.

Lemma A-1 Let $X \in$ *Conf*$(\mathcal{E}_R)$, $x = ((e_1, e_2), q) \in X$.

a) If $e_1 \in E_1^a, e_2 \in E_2^a, q \in E_Q \cup \{q_1, q_2\}$, then $x_1 := ((e_1, e_2), q_1) \in X$.

Proof:
$((e_1, e_2), q_1) \prec_R x$. Suppose $x_1 \notin X$. Then (iv) implies $\exists x' = ((e_1', e_2'), q') \in X$ with $x' \#_R ((e_1, e_2), q_1)$ and $\neg(x' \#_R x = ((e_1, e_2), q))$. But this is excluded by (KRa),(KRb),(KRc),(KRd),(KRe),(KRf),(KRg). So only (KRg) remains for $x' \#_R x_1$: $e_1' = e_1, e_2' = e_2, q' \#_Q q_1$. But $q' \#_Q q_1$ is impossible. Contradiction!

b) If $e_1 \in E_1^a, e_2 \in E_2^a, q \in E_Q \cup \{q_1, q_2\}$, and $\exists x' = ((e_1', e_2'), q') \in X$ with $e_1 \prec_1 e_1' \vee e_2' \prec_2 e_2'$, then $x_2 = ((e_1, e_2), q_2) \in X$.

Proof:
Suppose $((e_1, e_2), q_2) \notin X$. Since $((e_1, e_2), q_2) \prec_R x'$, (iv) implies $\exists \bar{x} = ((\bar{e}_1, \bar{e}_2), \bar{q}) \in X$ with $\bar{x} \prec_R x'$ and $\bar{x} \#_R ((e_1, e_2), q_2)$. Since $x = ((e_1, e_2), q) \in X$ $\bar{x} \#_R ((e_1, e_2), q)$ is not allowed. Thus all possibilities for conflict are forbidden. Contradiction! So $x_2 \in X$.

c) Let $e_1' \in E_1^a, q' \in E_Q$ and $\forall e_2' \in E_2^a : ((e_1', e_2'), q') \notin X$.
If there exists $x'' = ((e_1', e_2''), q'') \in X$ with $e_2'' \in E_2^a, q'' \in E_Q \cup \{q_1, q_2\}$ and $e_1' \prec_1 e_1$ or $e_2'' \prec_2 e_2$ holds,

then there exists $\tilde{q} \in E_Q$ such that $((e_1', e_2''), \tilde{q}) \in X$ and $q' \#_Q \tilde{q}$.

Proof:
With lemma A-1b one sees that $((e_1', e_2''), q_2) \in X$.
Since $q' \prec_{Q'} q_2$ we have $((e_1', e_2'), q') \prec_R ((e_1', e_2'), q_2)$. Since $((e_1', e_2'), q') \notin X$, (iv) implies that $\exists x'' \in X$ with $x'' \prec_R ((e_1', e_2'), q_2)$ and $x'' \#_R ((e_1', e_2'), q')$ and $\neg(x'' \#_R ((e_1', e_2'), q_2))$. This is only compatible with (KRg), thus $\exists q'' \in E_Q$ with $q'' \#_Q q'$ and $x'' = ((e_1', e_2'), q'') \in X$.

Lemma A-2 Let $X \in \mathit{Conf}(\mathcal{E}_R)$.

a) If $\exists x = ((e_1, e_2), q) \in X$ and $x' = ((e_1', e_2'), q') E_R \setminus X$ with $e_1, e_1' \in E_1$, then $\exists \tilde{x} = ((\tilde{e}_1, \tilde{e}_2), \tilde{q}) \in X$ with $e_1' \#_1 \tilde{e}_1 \prec_1 e_1$.

Proof follows directly from proposition A.

b) If $\exists x = ((e_1, e_2), q) \in X$ and $x' = ((e_1', e_2'), q') \in E_R \setminus X$ with $e_2, e_2' \in E_2$, then $\exists \tilde{x} = ((\tilde{e}_1, \tilde{e}_2), \tilde{q}) \in X$ with $e_2' \#_2 \tilde{e}_2 \prec_2 e_2$.

Proof follows directly from proposition A.

Lemma A-3

a) Let $Y \in \mathit{Conf}(\mathcal{E}_S)$, $y = ((e_1, *), q) \in Y$ with $e_1 \in E_1^a, q \in E_Q$. Then there exists a uniquely determined $e_2 \in E_2^a$ with $y_1 = ((e_1, e_2), q_1) \in Y$. It will be denoted by $h(y, Y)$.

Proof:
Existence of e_2:
Let $y' = ((e_1, *), q_1) \notin Y$, since y' self-conflicting, but $y' \prec_S y$. Thus $\exists y'' \in Y : y' \#_S y'' \prec_S y'$, and therefore $\neg(y'' \#_S y)$. Thus $y'' \#_S ((e_1, *), q_1), \neg(y'' \#_S ((e_1, *), q))$. Then the only possible reason for conflict is (KSc): $e_1'' = e_1$ & $q'' = q_1$ & $e_2'' \neq *$. Thus $\exists e_2 = e_2''$ with $y'' = ((e_1, e_2), q_1) \in Y$ $(y'' \prec_S y)$.

Uniqueness of e_2:
Suppose there exists $e_2' \neq e_2$ such that $((e_1, e_2'), q_1) \in Y$. With (KSd) $((e_1, e_2'), q_1)$ conflicts $((e_1, e_2), q_1)$ – thus they cannot be both contained in Y.

■

b) If $l(e_1) = l(e_2) = a$ is not auto-concurrent in $P_1 \parallel_A P_2$ for all $Y \in \mathit{Conf}(\mathcal{E}_S)$: $y_2 = ((e_1, e_2), q_2) \in Y \Rightarrow y_1 = ((e_1, e_2), q_1) \in Y$.

Proof:

One easily shows that a is auto-concurrent in $P_1 \|_A P_2$ iff a_1 is auto-concurrent in P_S. So we show: If there exists a configuration Y with $y_2 \in Y$ and $y_1 \notin Y$ then a_1 is auto-concurrent in P_S.

Let $y_2 \in Y$ and $y_1 \notin Y$. Consider $((e_1, *), q_1) \prec_S y_2$ and $((*, e_2), q_1) \prec_S y_2$, both being self-conflicting and therefore not contained in Y. So there must be $y_1', y_1'' \in Y$ with $((e_1, *), q_1) \#_S y_1' \prec_S y_2$ and $((e_2, *), q_1) \#_S y_1'' \prec_S y_2$. The only possibilities for this are $y_1' = ((e_1, e_2'), q_1), y_1'' = ((e_1'', e_2), q_1)$ with $e_2' \neq e_2, e_1'' \neq e_1$.

Define $Z = \{x \in E_S \mid x \prec_S y_1' \vee x \prec_s y_1''\} \cap Y \setminus \{y_1', y_1''\}$. Then $Z \in \mathit{Conf}(\mathcal{E}_S)$. Moreover $Z' = Z \cup \{y_1'\} \in \mathit{Conf}(\mathcal{E}_S)$, $Z'' = Z \cup \{y_1''\} \in \mathit{Conf}(\mathcal{E}_S)$ and
$Z' \cup Z'' = Z \cup \{y_1,' y_1''\} \in \mathit{Conf}(\mathcal{E}_S)$. Thus
$Z' \neq Z'', Z \rightarrow^{a_1} Z'$ & $Z \rightarrow^{a_1} Z''$ & $Z' \cup Z'' \in \mathit{Conf}(\mathcal{E})$. Therefore a_1 is auto-concurrent in P_S. ■

Lemma A-4 If $l(e_1) = l(e_2) = a$ is not auto-concurrent in $P_1 \|_A P_2$ and $y = ((e_1, e_2), q), y' = ((e_1', e_2'), q') \in Y \in \mathit{Conf}(\mathcal{E}_S)$, $y \prec_S y'$, then $g_Y(y) \prec_R g_Y(y')$.

Proof

If $y \prec_S y'$ holds by FSa), FSb), then $g_Y(y) \prec_R y'$ holds by FRa), FRb).
If $y \prec_S y'$ holds by FSc), i.e. $e_1' = e_1 \neq *, q' \prec_{Q'} q$, then
$g_Y(y) = ((e_1, e_{2x}), q), g_Y(y') = ((e_1', e_{2x}'), q')$ with
$e_{2x} = e_2 \vee e_{2x} = h(y, Y), e_{2x}' = e_2' \vee e_{2x}' = h(y', Y)$. If $e_{2x} \neq e_{2x}'$ then
$((e_1, e_{2x}), q_1) \#_S ((e_1', e_{2x}'), q_1)$ and with lemma A-3 $((e_1, e_2), q_1), ((e_1', e_2'), q_1) \in Y$, Contradiction. Thus (FRc) can be applied.
If $y \prec_S y'$ holds by FSd), i.e. $e_2' = e_2 \neq *, q' = q_1$ & $q = q_2$. Then with lemma A-3 $((e_1', e_2'), q_1) \in Y$ and therefore $e_1' = e_1$, so again (FRc) can be applied.

■

References

[AH91] L. Aceto, M. Hennessy, "Adding Action Refinement to a Finite Process Algebra", in J. Leach Albert, B. Monien, M. R. Artalejo (eds.): Automata, Languages and Programming, LNCS 510, pp. 506-519, Springer-Verlag, 1991.

[AH93] L. Aceto, M. Hennessy, "Towards action-refinement in process algebras", Information and Computation, vol. 103, pp. 204 -269, 1993.

[Bo90] G. Boudol, "Flow Event Structures and Flow Nets", in I. Guessarian (ed.): Semantics of Systems of Concurrent Processes, LNCS 469, pp. 62-95, Springer-Verlag, 1990.

[BC91] G. Boudol, I. Castellani, "Flow models of distributed computations: Three equivalent semantics for CCS", Information and Computation, vol. 114, pp. 247-314, 1994.

[BMC94] C. Baier, M.E. Majster-Cederbaum, "The connection between an event structure semantics and an operational semantics for TCSP", Acta Informatica, 31, 1994.

[Co95] R. Costantini, "Abstraktion in ereignisbasierten Modellen verteilter Systeme", Ph.D Thesis, University of Hildesheim, 1995.

[CZ89] I. Castellani, G.Q. Zhang, "Parallel product of event structures", Rapports de Recherche 1078, INRIA, 1989.

[DG95] P. Degano, R. Gorrieri, "A Causal Operational Semantics of Refinement", Information and Computation, vol. 122, pp. 97-119, 1995.

[DGR93] P. Degano, R. Gorrieri, G. Rosolini, "Graphs and Event Refinement", Proc. Workshop on Semantics: Theory and Applications, 1993.

[GG90] R. J. van Glabbeek, U. Goltz: "Equivalences and Refinement", in I. Guessarian (ed.): Semantics of Systems of Concurrent Processes, LNCS 469, pp. 309-333, Springer-Verlag, 1990.

[GGR92] U. Goltz, R. Gorrieri, A. Rensink, "On Syntactic and Semantic Action Refinement", Hildesheimer Informatik-Berichte 17/92, 1992.

[LG91] R. Loogen, U. Goltz, "Modelling nondeterministic concurrent processes with event structures", in Fundamenta Informatica, vol. XIV, pp. 39-74, 1991.

[NEL89] 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, pp. 523-548, Springer-Verlag, 1989.

[Sch91] S. Schreiber, "Fluss-Ereignisstrukturen als Modell fuer eine Sprache", Master's thesis, Rheinische Friedrich-Wilhelms-Universitaet Bonn, 1991.

[Wi89] G. Winskel, “An Introduction to Event Structures ”, 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, pp. 364–397, Springer-Verlag, 1989.