Reihe Informatik

7/92

On the definability of concurrency and communication: event structures versus pomset classes

C. Baier, M.E. Majster-Cederbaum

December 2, 1992

Contents

1 Introduction

In the context of communicating parallel process systems various paradigma for communication resp. synchronisation have been proposed. Two well-known theoretical models for communicating systems are *CCS* [15] and *TCSP* [5, 17]. A variety of semantics has been proposed for these and similar languages which can be characterized by different criteria: true versus interleaving parallelism, linear versus branching time models, operational versus denotational approaches, choice of the mathematical discipline to handle recursion and domain equations. In recent years interest has shifted more and more towards semantics that model true parallelism. The most known are petri net semantics [8, 10, 17, 18], event structure [3, 9, 21, 22] and pomset [2] semantics. The present paper investigates the question whether the two closely related approaches of event structures and pomsets are equally suitable to provide semantics for language constructs as avaible in *CCS* or *TCSP*. Given the variety of approaches to semantic description comparative studies like the present one are important as a guideline. They help us to decide which method suits which purpose. In addition, comparative studies enhance the better understanding of the language constructs, and finally comparative studies of semantics that yield consistency results as in [3, 6, 7, 14, 20] strengthen our confidence in the correctness of each of the semantics involved.

The paper is divided into seven sections. Section 2 introduces *CCS* and *TCSP* and discusses their communication mechanisms. Section 3 introduces event structures. Section 4 defines pomset classes. Section 5 models the communication mechanisms of *CCS* and *TCSP* using event structures and section 6 discusses the problems that arise when pomset classes are used. Section 7 is the conclusion. The appendix contains some formal definitions.

2 The languages *CCS* and *TCSP*

The language *CCS* [15, 16] is given by the following production system.

Definition 2.1
Let Δ be a set of names, $\overline{\Delta} = \{\, \bar{a} \,:\, a \in \Delta \,\}$, $\Delta \cap \overline{\Delta} = \emptyset$, $\bar{\bar{a}} = a$ for $a \in \Delta$, $\tau \notin \Delta \cup \overline{\Delta}$, $\mathcal{L} = \Delta \cup \overline{\Delta}$ and $Act = \mathcal{L} \cup \{\tau\}$.

A function $f : Act \rightarrow Act$ is called a `relabelling function` *for Act if $f(\tau) = \tau$ and $\overline{f(c)} = f(\bar{c})$ for $c \in \mathcal{L}$. Then CCS is given by the production system*

$$s \;::=\; nil \mid a.s \mid s_1 + s_2 \mid s_1|s_2 \mid x \mid fix(x = s) \mid s \setminus L \mid s[f]$$

where $a \in Act$, $L \subseteq \mathcal{L}$, f is a relabelling function and where x is an identifier.

nil models the inactive agent. The process $a.s$ first performs a and then acts as s. $s_1 + s_2$ describes the nondeterministic choice, $fix(x = s)$ is a recursion, $s \setminus L$ models restriction and $s[f]$ relabelling. Our interest lies in the parallel operator '|'. Milner [15] gave an interleaving operational semantics for *CCS* using a transition system. The corresponding rules for '|' are

$$\frac{s_1 \xrightarrow{a} s_1'}{s_1|s_2 \xrightarrow{a} s_1'|s_2} \qquad \frac{s_2 \xrightarrow{a} s_2'}{s_1|s_2 \xrightarrow{a} s_1|s_2'}$$

where $a \in Act$ and

$$\frac{s_1 \xrightarrow{c} s_1',\ s_2 \xrightarrow{\bar{c}} s_2'}{s_1|s_2 \xrightarrow{\tau} s_1'|s_2'}$$

where $c \in \mathcal{L}$.

The language *TCSP* [17] is given by the following production system.

Definition 2.2
Let $Act = Comm \cup \{\tau\}$ where Comm is a set of communications, $\tau \notin Comm$. TCSP is given by the following production system:

$$t \quad ::= \quad stop \mid a.t \mid t_1 \; or \; t_2 \mid t_1 \Box t_2 \mid t_1 \parallel_A t_2 \mid x \mid fix\text{-}x.t \mid t \setminus c \mid$$

where $a \in Act$, $c \in Comm$, $A \subseteq Comm$ and x is an identifier.

stop models inactiveness. $a.t$ stands for the process which first performs a and then acts as t. *or* resp. $\Box$ stands for internal resp. external nondeterminism. *fix* $x.t$ is a recursion, $t \setminus c$ models hiding, i.e. relabelling the action c by the internal action τ. In the parallel composition $t_1 \parallel_A t_2$ actions of A may only be executed as joint events by both partners together.

Olderog [17] gave an interleaving operational semantics for *TCSP* using a transition system. The rules for $\parallel_A$ are

$$\frac{t_1 \xrightarrow{a} t_1'}{t_1 \parallel_A t_2 \xrightarrow{a} t_1' \parallel_A t_2} \qquad \frac{t_2 \xrightarrow{a} t_2'}{t_1 \parallel_A t_2 \xrightarrow{a} t_1 \parallel_A t_2'}$$

where $a \notin A$ and

$$\frac{t_1 \xrightarrow{c} t_1',\ t_2 \xrightarrow{c} t_2'}{t_1 \parallel_A t_2 \xrightarrow{c} t_1' \parallel_A t_2'}$$

where $c \in A$.

In *TCSP*, if a process t_1 runs in parallel with t_2 with respect to some set $A \subseteq Comm$ (i.e. we consider $t_1 \parallel_A t_2$) and the action that t_1 wants to perform next is a then either $a \notin A$ and t_1 can proceed or $a \in A$ then t_1 can only proceed jointly with t_2 who must then also want to perform a next. In contrast to this the *CCS* parallel operator does not enforce synchronisation. A process s_1 running in parallel with s_2 (i.e. we consider $s_1|s_2$) either performs an action $c \in \mathcal{L}$ independently or may cooperate with s_2 who wants to perform $\bar{c}$.

The above described semantics for *TCSP* and *CCS* are interleaving. We are here investigating the question to what extent event structures and pomsets, that are suitable for describing true parallelism, can capture these synchronisation mechanisms.

3 Event structures

Event structures have been introduced by [22]. In this (resp. the next) section we will introduce the class *Ev* of labelled and finitely approximable event structures (resp. the class *Pom** of pomset classes). Event structures and pomset classes have been used to provide a compositional semantics for *CCS*- and *TCSP*-like languages, i.e. semantic operators on event structures (resp. pomset classes) have been defined that model the syntactic language constructs [2, 9, 21, 22]. In order to be able to give a meaning to recursive constructs it has been proposed to impose a metric on *Ev* (resp. *Pom**) that turns *Ev* (resp. *Pom**) into a complete metric space [2, 9]. Making use of Banach's fixed point theorem there is then a standard way to give a meaning to guarded recursion provided that the semantic operator for prefixing is contractive and that the remaining semantic operators are **nondistance increasing**. As it will turn out there is no problem in defining nondistance increasing semantic operators for communicating parallelism using event structures that capture the intended meaning. In contrast to this using pomset classes causes considerable problems.

3.1 Definitions

Definition 3.1
$\varepsilon = (E, \leq, \#, l)$ is called a (`labelled` *and* `finitely approximable`*)* `event structure` *iff*

1. *E is a set (of* `events` *),*

2. *$\leq$ is a partial order on E*
(i.e. $\leq$ is a transitive, reflexive, antisymmetric relation on E),

3. # is an irreflexive, symmetric relation on E , called conflict relation, *with:*

$$\forall e_1, e_2, e_3 \in E: \; (e_1 \leq e_2 \textit{ and } c_1 \# e_3 \;) \implies e_2 \# e_3$$

4. $l: E \to Act$ is a function. l is called the labelling function.

where

(1) For each event $e \in E$ $depth_\varepsilon(e)$ is finite where

$$depth_\varepsilon(e) \;=\; \max \; \{ \;\; n \in I\!N_0 \;: \exists e_1, \ldots e_n \in E \quad e_1 \leq e_2 \leq \ldots \leq e_n = e, \; e_i \neq e_j, \; 1 \leq i < j \leq n \;\; \}$$

(2) For each $n \in I\!N_0$ the set $E_\varepsilon[n] \;=\; \{ \; e \in E \;:\; depth_\varepsilon(e) \;\leq\; n \; \}$ is finite.

In the following we write $depth(e)$ instead of $depth_\varepsilon(e)$ and $E[n]$ instead of $E_\varepsilon[n]$.

Two event structures $\varepsilon_i = (E_i, \leq_i, \#_i, l_i)$, $i = 1, 2$, are isomorphic if there exists a bijective mapping $f: E_1 \to E_2$ such that

1. $e_1 \;\leq_1\; e_2 \iff f(e_1) \;\leq_2\; f(e_2) \;\; \forall e_1, e_2 \in E_1$,
2. $e_1 \;\#_1\; e_2 \iff f(e_1) \;\#_2\; f(e_2) \;\; \forall e_1, e_2 \in E_1$ and
3. $l_2(f(e)) \;=\; l_1(e) \;\; \forall e \in E_1$.

In the following we abstract from the names of the events, i.e. we will not distinguish between isomorphic event structures.

Definition 3.2
Ev denotes the set of all event structures (modulo isomorphism).

The empty event structure $(\emptyset, \emptyset, \emptyset, \emptyset)$ is denoted by $\emptyset$.

Event structures can be depicted graphically by representing events as boxes (inscribed with the event label) and connecting them with their direct predecessors and successors.
A conflict between two events is a direct conflict if no predecessors of the events are in conflict. Direct conflicts are depicted graphically by a broken line. Example: The event structure $\varepsilon = (E, \leq, \#, l)$ with

1. $E \;=\; \{e_1, e_2, e_3\}$
2. $e_1 \leq e_2, \;\; e_1 \# e_3, \;\; e_2 \# e_3$ and
3. $l(e_1) = a, \; l(e_2) = b, \; l(e_3) = c$

is shown as

$$\begin{tikzcd} \boxed{a} \arrow[r] \arrow[d, dash, dashed] & \boxed{b} \\ \boxed{c} & \end{tikzcd}$$

Event structures are used to model language constructs as follows: sequencing is modelled by the partial order. Parallel execution without communication is modelled by incomparability with respect to the partial order and nondeterministic choice is modelled by the conflict relation.

3.2 The metric space (Ev, d)

Definition 3.3
Let $\varepsilon = (E, \leq, \#, l)$ be an event structure, $A \subseteq E$. A is called leftclosed *iff each predecessor of an event $e \in A$ belongs to A, i.e.*

$$e \in A, \; e' \in E, \; e' \;\leq\; e \implies e' \in A$$

If A is a leftclosed subset of E, then the event structure $\varepsilon\lceil A$ is defined by

$$\varepsilon\lceil A \;=\; (\,A,\; \leq\,\cap\, A\times A,\; \#\,\cap\, A\times A,\; l|A\,)$$

It is clear that $E[n]$ is leftclosed. We define

$$\varepsilon[n] \;=\; \varepsilon\lceil\, E[n]$$

$\varepsilon[n]$ is called the n-cut of ε.

Definition 3.4
Let $d\;:\;Ev\times Ev \longrightarrow [0,1]$ be defined by

$$d(\varepsilon_1,\varepsilon_2) \;=\; \inf\,\{\frac{1}{2^n}\;:\;\varepsilon_1[n] \;=\; \varepsilon_2[n]\,\}$$

It has been shown in [9] that (Ev,d) is a complete ultrametric space. Each event structure ε can be approximated by its n-cuts $\varepsilon[n]$. We have:

$$d(\,\varepsilon[n],\;\varepsilon\,) \;\leq\; \frac{1}{2^n} \quad \forall\, n\geq 0$$

4 Pomset classes

Pomset classes are nonempty and closed sets of conflictfree event structures. The metric d on Ev induces a metric on $Pom_\emptyset^*$ - the class of all pomset classes - which turns $Pom_\emptyset^*$ into a complete ultrametric space.

Pomset classes are used in the same way as event structures to model language constructs except for nondeterministic choice. The alternatives of a nondeterministic choice are collected into a set of pomsets instead of connecting them via a conflict relation.

4.1 Pomsets

Definition 4.1
An event structure $\varepsilon \;=\; (E,\leq,\#,l)$ is called `conflictfree` *iff E does not contain conflicting events, i.e. iff $\# \;=\; \emptyset$.*
Conflictfree event structures are also called `pomsets`.

Let Pom denote the class of all pomsets.

$$Pom \;=\; \{\,p\in Ev\;:\;p \text{ is conflictfree }\}$$

If $\varepsilon \;=\; (E,\leq,\#,l)$ is a pomset (i.e. $\# \;=\; \emptyset$) we write shortly $\varepsilon \;=\; (E,\leq,l)$.

Lemma 4.2
Pom is a closed subspace of Ev. In particular Pom (with the subspace metric) is a complete ultrametric space.

Proof: It is easy to see that the limit of a convergent sequence of conflictfree event structures is conflictfree.

$\square$

4.2 The metric space $(Pom_\emptyset^*, d)$

Definition 4.3
Let $Pom_\emptyset^*$ denote the class of all closed subsets of Pom. The elements of $Pom_\emptyset^*$ are called `pomset classes`.

Definition 4.4
The metric d on Pom induces the `Hausdorff-metric` *on* $Pom_\emptyset^*$ *(which is also called d)*

$$d(H_1, H_2) = \max\{ \sup_{p\in H_1} \inf_{q\in H_2} d(p,q), \sup_{p\in H_2} \inf_{q\in H_1} d(p,q) \}$$

for all H_1, $H_2 \in Pom_\emptyset^*$.

The Hausdorff-metric on the set of nonempty and closed subspaces of a complete ultrametric space is a complete ultrametric space. Hence $(Pom_\emptyset^*, d)$ is a complete ultrametric space (see [11]).

5 Parallel operators on *Ev*

5.1 The *CCS* parallel operator on *Ev*

We introduce a parallel operator | on *Ev* which models the *CCS* parallelism with communication by giving some examples. The formal definition can be found in the appendix.

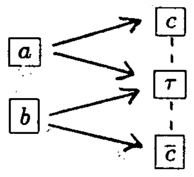
Example 5.1
Let ε_1, ε_2, ε_3 resp. ε_4 be given by

$$\begin{tikzcd} \boxed{a} \arrow[r] & \boxed{c} \\ \boxed{c} & \end{tikzcd} \qquad \begin{tikzcd} \boxed{a} \arrow[r] & \boxed{\bar{c}} \end{tikzcd} \qquad \begin{tikzcd} \boxed{b} \arrow[r] & \boxed{\bar{c}} \end{tikzcd}$$

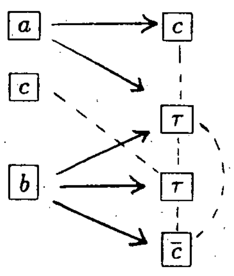
resp. $\boxed{a} \rightarrow \boxed{c}$.
Since ε_2 and ε_3 do not contain complementary actions no communication in $\varepsilon_2|\varepsilon_3$ is possible. We get

$$\varepsilon_2|\varepsilon_3 = \begin{tikzcd} \boxed{a} \arrow[r] & \boxed{\bar{c}} \\ \boxed{b} \arrow[r] & \boxed{\bar{c}} \end{tikzcd}$$

The events labelled by c resp. $\bar{c}$ in ε_3 resp. ε_4 are able to communicate. We get:

$\varepsilon_4 \mid \varepsilon_3 =$ 

Next we look for $\varepsilon_1|\varepsilon_3$. The $\bar{c}$-event in ε_3 has two possibilities to communicate. We get

$\varepsilon_1 \mid \varepsilon_3 =$ 

Lemma 5.2

$$d(\varepsilon_1|\varepsilon_2, \varepsilon_1'|\varepsilon_2') \leq \max\{ d(\varepsilon_1,\varepsilon_1'), d(\varepsilon_2,\varepsilon_2') \}$$

for all ε_1, ε_1', ε_2, $\varepsilon_2' \in Ev$.

Proof: We show

(a) $(\varepsilon_1|\varepsilon_2)[n] \;=\; (\;\varepsilon_1[n] \mid \varepsilon_2[n]\;)[n]$

(b) $d(\;\varepsilon_1|\varepsilon_2,\;\varepsilon_1'|\varepsilon_2'\;) \;\leq\; \max\{\;d(\varepsilon_1,\varepsilon_1'),\;d(\varepsilon_2,\varepsilon_2')\;\}$

for all $\varepsilon_1, \varepsilon_1', \varepsilon_2, \varepsilon_2' \in Ev$ and $n \geq 0$.

ad (a): Let $\varepsilon_i = (E_i, \leq_i, \#_i, l_i)$, $i = 1, 2$, $E_1 \cap E_2 = \emptyset$. $\varepsilon \;=\; \varepsilon_1|\varepsilon_2$, $\varepsilon' \;=\; \varepsilon_1[n]|\varepsilon_2[n]$. Let $\mathcal{C}$ resp. $\mathcal{C}'$ denote the set of 'possible' events in ε resp. ε'.

$$\mathcal{C} \;=\; E_1 \;\cup\; E_2 \;\cup\; \mathcal{C}_{Comm}(\varepsilon_1, \varepsilon_2)$$

resp.

$$\mathcal{C}' \;=\; E_1[n] \;\cup\; E_2[n] \;\cup\; \mathcal{C}_{Comm}(\varepsilon_1[n], \varepsilon_2[n])$$

Then $\mathcal{C}' \subseteq \mathcal{C}$ and for all (e_1, e_2), $(e_1', e_2') \in \mathcal{C}'$:

$$(e_1, e_2) \;\; R_{\mathcal{C}'} \;\; (e_1', e_2') \;\;\Longleftrightarrow\;\; (e_1, e_2) \;\; R_{\mathcal{C}} \;\; (e_1', e_2')$$

$$(e_1, e_2) \;\; \#_{\mathcal{C}'} \;\; (e_1', e_2') \;\;\Longleftrightarrow\;\; (e_1, e_2) \;\; \#_{\mathcal{C}} \;\; (e_1', e_2')$$

In particular: If C is a subset of $\mathcal{C}'$ then C is leftclosed (conflictfree resp. linear) as a subset of $\mathcal{C}'$ if and only if it is leftclosed as a subset of $\mathcal{C}$. Let E resp. E' denote the set of events in ε resp. ε', i.e. E resp. E' is the set of leftclosed, conflictfree and linear subsets of $\mathcal{C}$ resp. $\mathcal{C}'$. Then $E' \subseteq E$ and therefore $E'[n] \subseteq E[n]$. We show that $E[n] = E'[n]$.
If $C \in E$, $depth_\varepsilon(C) \leq n$, then $depth(e) \leq n$ for each event e in $E_1 \cup E_2$ which occurs as a component of an element in C. We conclude:

$$(e_1, e_2) \in C \;\;\Longrightarrow\;\; e_i \in E_i \cup \{\star\}$$

Therefore $C \subseteq \mathcal{C}'$ belongs to $E'[n]$.

Since the partial orders on E and E' correspond to the 'subset'-relation $\subseteq$ and since the conflict relation on E' is the restriction of the conflict relation of E on E' we get:

$$(\varepsilon_1|\varepsilon_2)[n] \;=\; (\;\varepsilon_1[n] \mid \varepsilon_2[n]\;)[n]$$

ad (b): If

$$\max\{\;d(\varepsilon_1, \varepsilon_1'),\; d(\varepsilon_2, \varepsilon_2')\;\} \;=\; r$$

then either $r = 0$ or r is of the form $\frac{1}{2^n}$ for some natural number n. If $r = 0$ then $\varepsilon_i = \varepsilon_i'$ and there is nothing to show. Now we assume that $r = \frac{1}{2^n}$ for some n. Then

$$d(\varepsilon_i, \varepsilon_i') \;\leq\; \frac{1}{2^n}, \quad i = 1, 2$$

and therefore $\varepsilon_1[n] = \varepsilon_1'[n]$, $\varepsilon_2[n] = \varepsilon_2'[n]$. We get by (a):

$$(\varepsilon_1|\varepsilon_2)[n] \;=\; (\varepsilon_1[n]|\varepsilon_2[n])[n] \;=\; (\varepsilon_1'[n]|\varepsilon_2'[n])[n] \;=\; (\varepsilon_1'|\varepsilon_2')[n]$$

and therefore

$$d(\;\varepsilon_1|\varepsilon_2,\;\varepsilon_1'|\varepsilon_2'\;) \;\leq\; \frac{1}{2^n}$$

□

5.2 The *TCSP* parallel operator on *Ev*

This operator has been already introduced by [9]. We present only examples. The formal definition of [9] can be found in the appendix.

Example 5.3
We consider

$$\varepsilon_1 = \begin{tikzcd} \boxed{b} \arrow[r] & \boxed{c} \arrow[r] & \boxed{a} \end{tikzcd}$$

and

$$\varepsilon_2 = \begin{tikzcd} \boxed{d} \arrow[r] & \boxed{c} \arrow[r] & \boxed{b} \end{tikzcd}$$

Then $\varepsilon_1 \; ||_{\{c\}} \; \varepsilon_2$ is given by

$$\begin{tikzcd} \boxed{b} \arrow[dr] & & \boxed{a} \\ & \boxed{c} \arrow[ur] \arrow[dr] & \\ \boxed{d} \arrow[ur] & & \boxed{b} \end{tikzcd}$$

and $\varepsilon_1 \; ||_{\emptyset} \; \varepsilon_2$ is given by

$$\begin{tikzcd} \boxed{b} \arrow[r] & \boxed{c} \arrow[r] & \boxed{a} \\ \boxed{d} \arrow[r] & \boxed{c} \arrow[r] & \boxed{b} \end{tikzcd}$$

Example 5.4
Let ε_1 be $\boxed{c} \longrightarrow \boxed{b}$ and ε_2 be $\boxed{d}$. Then $\varepsilon_1 \; ||_{\{c\}} \; \varepsilon_2$ is the empty event structure which represents a process where no action is possible.

This example shows that events may become impossible in the parallel composition.

Example 5.5
Let ε_1 be $\boxed{c} \longrightarrow \boxed{b}$ and ε_2 be given by

$$\begin{tikzcd} \boxed{c} \arrow[d, dash, dashed] & \\ \boxed{a} \arrow[r] & \boxed{c} \end{tikzcd}$$

then there are two possible communications c in $\varepsilon_1 \; ||_{\{c\}}\varepsilon_2$, depending on which branch is chosen in ε_2. $\varepsilon_1 \; ||_{\{c\}} \; \varepsilon_2$ is given by

$$\begin{tikzcd} \boxed{c} \arrow[r] \arrow[d, dash, dashed] & \boxed{b} & \\ \boxed{a} \arrow[r] & \boxed{c} \arrow[r] & \boxed{b} \end{tikzcd}$$

Lemma 5.6
Let A be a subset of Comm and $\varepsilon_1, \varepsilon_1', \varepsilon_2, \varepsilon_2' \in Ev$. Then

$$d(\; \varepsilon_1 ||_A \varepsilon_2, \; \varepsilon_1' ||_A \varepsilon_2' \;) \;\; \leq \;\; \max \{ \; d(\varepsilon_1, \varepsilon_1'), \; d(\varepsilon_2, \varepsilon_2') \; \}$$

Proof: see [9].

From Lemma 5.2 and 5.6 we conclude that both *CCS* and *TCSP* synchronisation/ communication can be modelled using event structures, as the respective parallel operators are shown to be nondistance increasing and as the other operators exhibit the desired properties [1, 9] to guarantee the solution of fixed point equations arising from recursion.

6 Parallel operators on $Pom^*_\emptyset$

In [2] de Bakker and Warmerdam attempt to model *CCS*-style synchronisation using pomset classes. The first attempt that models intuition best would be to associate with e.g. the *CCS*-program $s = a.nil \mid \bar{a}.nil$ the pomset class

$$\begin{tikzcd}[row sep=small] & \boxed{a} & & \\ \{ & & , \; \boxed{\tau} & \} \\ & \boxed{\bar{a}} & & \end{tikzcd}$$

which expresses that a and $\bar{a}$ can be performed independently or that the processes act jointly which results in the silent action τ. As [2] observe, this intuitive modelling results in a semantic operator for '|' that is not nondistance increasing. [2] give the following example: Let $s_1 = a.c.nil$, $s_2 = a.d.nil$ and $s_3 = \bar{c}.nil$ with associated meanings

$$Me[s_1] = \{ \boxed{a} \longrightarrow \boxed{c} \}, \; Me[s_2] = \{ \boxed{a} \longrightarrow \boxed{d} \}$$

resp. $Me[s_3] = \{ \boxed{\bar{c}} \}$.
One would like to define a semantic operator $|_{Pom^*_\emptyset}$ such that

$$\begin{tikzcd}[row sep=small] & & \boxed{a} \arrow[r] & \boxed{c} & & & \\ Me[\, s_1|s_3 \,] = Me[s_1] \;|_{Pom^*_\emptyset}\; Me[s_3] = & \{ & & & , \; \boxed{a} \arrow[r] & \boxed{\tau} & \} \\ & & \boxed{\bar{c}} & & & & \end{tikzcd}$$

and

$$\begin{tikzcd}[row sep=small] & & \boxed{a} \arrow[r] & \boxed{d} & \\ Me[\, s_2|s_3 \,] = Me[s_2] \;|_{Pom^*_\emptyset}\; Me[s_3] = & \{ & & & \} \\ & & \boxed{\bar{c}} & & \end{tikzcd}$$

but

$$d(\, Me[s_1] \;|_{Pom^*_\emptyset}\; Me[s_3],\; Me[s_2] \;|_{Pom^*_\emptyset}\; Me[s_3] \,) = 1$$

while $d(\, Me[s_1],\; Me[s_2] \,) = \frac{1}{2}$, hence such a semantic operator is not nondistance increasing.

As a remedy to this situation [2] propose to use a modified operator that includes interleaving behaviour and yields the following expression for $s_1|s_3$:

$$\begin{tikzcd}[row sep=small] & \boxed{a} \arrow[r] & \boxed{c} & & & & & \boxed{c} & \boxed{a} \arrow[dr] & & \\ \{ & & & , \; \boxed{a} \arrow[r] & \boxed{\tau} & , \; \boxed{a} \arrow[urr] \arrow[drr] & & , & & \boxed{c} & , \\ & \boxed{\bar{c}} & & & & & & \boxed{\bar{c}} & \boxed{\bar{c}} \arrow[ur] & & \end{tikzcd}$$

$$\boxed{a} \longrightarrow \boxed{c} \longrightarrow \boxed{\bar{c}} \;,\; \boxed{a} \longrightarrow \boxed{\bar{c}} \longrightarrow \boxed{c} \;,$$

$$\boxed{\bar{c}} \longrightarrow \boxed{a} \longrightarrow \boxed{c} \; \}$$

This operator is shown in [2] to have the desired mathematical properties but this solution is hardly satisfactory from a semantic point of view. In addition, this parallel operator allows the processes (represented by pomset classes) to communicate with themselves. We consider a *CCS* process with relabelling $s = (b|c)[f]$ where f is a relabelling function with $f(b) = c$. Then the meaning of s should be $Me[s] = \{p\}$ where

$$\begin{tikzcd}[row sep=small] & \boxed{c} \\ p = & \\ & \boxed{c} \end{tikzcd}$$

We consider the parallel execution of s and nil, i.e. the process $s' = s|nil$. If $|'$ denotes the parallel operator defined in [2] and if we assume that the meaning function Me is compositional we get

$$\boxed{\tau} \in \{p\} \,|'\, \emptyset \;=\; Me[s] \,|'\, Me[nil] \;=\; Me[s']$$

One might think that the difficulties in using pomset classes for modelling communication originate in the specific CCS properties. However analogous problems arise when we try to model $TCSP$ using pomset classes. Let us consider the $TCSP$ programs $t_1 = a.c.stop \,||_\emptyset\, c.stop$, $t_2 = a.d.stop \,||_\emptyset\, c.stop$ and $t_3 = c.stop$ with associated meanings

$$Me[t_1] = \left\{ \begin{tikzcd} \boxed{a} \arrow[r] & \boxed{c} \\ \boxed{c} & \end{tikzcd} \right\} , \quad Me[t_2] = \left\{ \begin{tikzcd} \boxed{a} \arrow[r] & \boxed{d} \\ \boxed{c} & \end{tikzcd} \right\}$$

resp. $Me[t_3] = \{ \boxed{c} \}$. Then we would like to define a semantic operator $||_{\{c\}}$ such that

$$Me[\, t_1||_{\{c\}}t_3 \,] = Me[t_1] \,||_{\{c\}}\, Me[t_3] = \left\{ \begin{tikzcd} \boxed{a} \\ \boxed{c} \end{tikzcd} ,\; \begin{tikzcd} \boxed{a} \arrow[r] & \boxed{c} \end{tikzcd} \right\}$$

and

$$Me[\, t_2||_{\{c\}}t_3 \,] = Me[t_2] \,||_{\{c\}}\, Me[t_3] = \left\{ \begin{tikzcd} \boxed{a} \arrow[r] & \boxed{d} \\ \boxed{c} & \end{tikzcd} \right\}$$

But then again

$$d(\, Me[t_1] \,||_{\{c\}}\, Me[t_3],\; Me[t_2] \,||_{\{c\}}\, Me[t_3] \,) = 1$$

while $d(\, Me[t_1],\, Me[t_2] \,) = \frac{1}{2}$.

Let us finally remark that even though pomsets are event structures the definition of the parallel operators on Ev does not carry over to $Pom^*_\emptyset$ as Pom is not closed with respect to these operators.

In [1] where pomsets and event structure semantics are studied in more detail, it is shown how an event structure can be decomposed into maximal components which are pomset classes. Based on this result one might think of proceeding as follows: Given two pomset classes one applies a parallel operator to Ev elementwise. Each event structure in the result is then decomposed into its maximal components. Unfortunately this definition yields again a parallel operator that is not nondistance increasing.

7 Conclusion

The question to what extent the two closely related 'true parallelism' models of event structures and pomset classes are suitable to describe communicating and synchronising parallel processes has been discussed. We found that - using a metric framework - event structures seem more suitable for modelling various types of synchronisation since it is possible to define nondistance increasing operators on Ev for modelling the CCS or $TCSP$ communication whereas this is not possible on pomset classes. It is an open problem if the use of complete partial orders will change the situation.

A Appendix

We give the formal definitions of the CCS and $TCSP$ parallel operators on Ev. They are defined in a similar to that proposed in [21].

A.1 The CCS parallel operator on Ev

Definition A.1
Let $e_i = (E_i, \leq_i, \#_i, l_i)$ be event structures, $i = 1, 2$. We assume w.l.o.g. that $E_1 \cap E_2 = \emptyset$.

$\mathcal{C}_{Comm}(\varepsilon_1, \varepsilon_2) \;=\; \mathcal{C}_{Comm}$ *denotes the set of possible communications*

$$\mathcal{C}_{Comm} \;=\; \{\, (e_1, e_2) \in E_1 \times E_2 \;:\; l_1(e_1) \;=\; \overline{l_2(e_2)} \in Comm \,\}$$

and let $\mathcal{C}$ *denote the set of all possible events*

$$\mathcal{C} \;=\; E_1 \,\cup\, E_2 \,\cup\, \mathcal{C}_{Comm}$$

Let $\star$ *be a symbol which does not belong to* E_1 *or* E_2. *We identify each event* e *in* ε_1 *resp.* ε_2 *with* $(e, \star)$ *resp.* $(\star, e)$. *We extend* $\leq_i$ *and the conflict relation on* $E_i \cup \{\star\}$ *in the following way:*

(1) $(e \leq_i \star) \vee (\star \leq_i e) \implies e = \star$

(2) $\neg(e \#_i \star) \wedge \neg(\star \#_i e)$

for all $e \in E_i \,\cup\, \{\star\}$.

Let $\leq_{\mathcal{C}}$ *be the transitive, reflexive closure of* $\rightarrow$ *where the binary relation* $\rightarrow$ *is given by*

$$(e_1, e_2) \rightarrow (e_1', e_2') \;:\Longleftrightarrow\; [(e_1 \leq_1 e_1') \wedge \neg(e_2 >_2 e_2')] \vee [(e_2 \leq_2 e_2') \wedge \neg(e_1 >_1 e_1')]$$

(Here $e >_i f$ *means* $(f \leq_i e) \wedge (f \neq e)$*.)*
The conflict relation $\#_{\mathcal{C}}$ *on* $\mathcal{C}$ *is given by*

$$\begin{aligned}(e_1, e_2) \#_{\mathcal{C}} (e_1', e_2') \;:\Longleftrightarrow\; & (e_1 \#_1 e_1') \vee (e_2 \#_2 e_2') \vee \\ & [\, (e_1 = e_1') \wedge (e_2 \neq e_2') \,] \vee \\ & [\, (e_2 = e_2') \wedge (e_1 \neq e_1') \,]\end{aligned}$$

Let C *be a subset of* $\mathcal{C}$. C *is called*

- `leftclosed` *if for each pair* $(e_1, e_2) \in C$
  If $e_1' \in E_1$, $e_1' \leq_1 e_1$ *then there exists* $e_2' \in E_2 \cup \{\star\}$ *such that* $(e_1', e_2') \in C$ *and* $(e_1', e_2') \rightarrow (e_1, e_2)$.
  If $e_2' \in E_2$, $e_2' \leq_2 e_2$ *then there exists* $e_1' \in E_1 \cup \{\star\}$ *such that* $(e_1', e_2') \in C$ *and* $(e_1', e_2') \rightarrow (e_1, e_2)$.
- `conflictfree` *if* $\neg(\xi \#_{\mathcal{C}} \xi')$ *for all* $\xi, \xi' \in C$.
- `linear` *if* $\leq_C = \leq_{\mathcal{C}} \cap\, C \times C$ *is a partial order on* C *(i.e.* $\leq_C$ *is antisymmetric) and if there exists a unique maximal element (with respect to* $\leq_C$*) in* C. *The maximal element of* C *is denoted by* $\max(C)$.

Let E *denote the set of all leftclosed conflictfree and linear subsets of* $\mathcal{C}$. *We define a conflict relation* $\#$ *on* E *as follows:*

$$C_1 \# C_2 \;:\Longleftrightarrow\; \exists \xi_1 \in C_1 \; \exists \xi_2 \in C_2 \;\; \xi_1 \#_{\mathcal{C}} \xi_2$$

Then

$$\varepsilon_1 | \varepsilon_2 \;=\; (E, \subseteq, \#, l)$$

is an event structure where the labelling function $l : E \rightarrow Act$ *is given by*

$$l(C) \;=\; \begin{cases} l_i(\,\max(C)\,) & : \text{if } \max(C) \in E_i,\; i = 1, 2 \\ \tau & : \text{if } \max(C) \in \mathcal{C}_{Comm} \end{cases}$$

It is easy to see that $\varepsilon_1 | \varepsilon_2$ is indeed an event structure.

A.2 The $TCSP$ parallel operator on Ev

We give the formal definition of the $TCSP$ parallel operator as it was introduced in [9]. We define the syntactic communication of two event structures $\varepsilon_1, \varepsilon_2$ with respect to some set of actions A, denoted by $\mathcal{D}_A(\varepsilon_1, \varepsilon_2)$. This set contains all pairs (e_1, e_2) of events e_1 in ε_1, e_2 in ε_2 with the same label $c \in A$ (these pairs describe potential communications). In addition, $\mathcal{D}_A(\varepsilon_1, \varepsilon_2)$ contains a representation of all events in ε_1 or in ε_2 that are not related to A.
The events of the parallel composition are certain conflictfree subsets of $\mathcal{D}_A(\varepsilon_1, \varepsilon_2)$, where conflictfreeness of a set $C \subset \mathcal{D}_A(\varepsilon_1, \varepsilon_2)$ means that C does not contain conflicting 'communications', i.e. two elements (e_1, e_2), (e'_1, e'_2) such that either they contain conflicting events or one event communicates with two distinct events.

Definition A.2
Let $\varepsilon_i = (E_i, \leq_i, \#_i, l_i) \in Ev$, $i = 1, 2$, w.l.o.g. $E_1 \cap E_2 = \emptyset$ and $A \subseteq Comm$.

The `syntactic communication` *of ε_1 and ε_2 on A is defined by*

$$\begin{array}{rcll} \mathcal{D}_A(\varepsilon_1, \varepsilon_2) & = & \mathcal{D}_A & = \{ (e, \star) : e \in E_1,\ l_1(e) \notin A \} \ \cup \\ & & & \{ (\star, e) : e \in E_2,\ l_2(e) \notin A \} \ \cup \\ & & & \{ (e_1, e_2) \in E_1 \times E_2 : l_1(e_1) = l_2(e_2) \in A \} \end{array}$$

Here, $\star$ is an auxiliary symbol, $\star \notin E_1 \cup E_2$. We extend the relations $\leq_i$ and $\#_i$ on $E_i \cup \{\star\}$ by defining

(1) $(\star \leq_i e) \lor (e \leq_i \star) \iff e = \star$

(2) $\neg(\star \#_i e \ \lor \ e \#_i \star) \ \forall e \in E_i \cup \{\star\}$

Two communications (e_1, e_2), $(e'_1, e'_2) \in \mathcal{D}_A$ are in `conflict` *iff they contain conflicting events, i.e. $e_1 \#_1 e'_1$ or $e_2 \#_2 e'_2$, or one event communicates with two distinct events, i.e. $(e_1 = e'_1 \neq \star \land e_2 \neq e'_2)$ or $(e_2 = e'_2 \neq \star \land e_1 \neq e'_1)$.*

The relation $\rightarrow_A$ on $\mathcal{D}_A$ is defined by

$$(e_1, e_2) \rightarrow_A (f_1, f_2) \iff ((e_1 \leq_1 f_1) \land \neg(e_2 >_2 f_2)) \lor ((e_2 \leq_2 f_2) \land \neg(e_1 >_1 f_1))$$

The transitive closure of $\rightarrow_A$ is denoted by $\prec$. If C is a subset of $\mathcal{D}_A$ then $\prec_C$ denotes the restriction of $\prec$ on C.

A subset C of $\mathcal{D}_A$ is called

- `conflictfree` *iff no two communications in C are in conflict.*
- `leftclosed` *iff*
  $\forall (e_1, e_2) \in C, \forall f_1 \in E_1$ with $f_1 \leq_1 e_1$ there exists $(f_1, f_2) \in C$ with

  $$(f_1, f_2) \rightarrow_A (e_1, e_2)$$

  and symmetrically
  $\forall (e_1, e_2) \in C, \forall f_2 \in E_2$ with $f_2 \leq_2 e_2$ there exists $(f_1, f_2) \in C$ with

  $$(f_1, f_2) \rightarrow_A (e_1, e_2).$$

- `linear` *iff $\prec_C$ is antisymmetric and there exists exactly one maximal element in C (with respect to $\prec_C$). This is denoted by* $\max(C)$.

The `parallel composition` *of ε_1 and ε_2 with communication on $A \subseteq Comm$ is given by*

$$\varepsilon_1 \parallel_A \varepsilon_2 := (E, \subseteq, \#, l)$$

where E denotes the set of conflictfree, leftclosed and linear subsets of $\mathcal{D}_A$. The conflict relation # on E is given by

$$C_1 \# C_2 \quad :\Longleftrightarrow \quad \exists \eta_1 \in C_1,\ \eta_2 \in C_2 \text{ such that } \eta_1 \text{ and } \eta_2 \text{ are in conflict}$$

The labelling function $l : E \rightarrow Act$ is given by

$$l(C) = \begin{cases} l_1(e) & : \textit{if} \max(C) = (e, \star) \\ l_2(e) & : \textit{if} \max(C) = (\star, e) \\ c & : \textit{if} \max(C) = (e_1, e_2) \in E_1 \times E_2,\ l_1(e_1) = l_2(e_2) = c \in A \end{cases}$$

References

[1] C. Baier, M.E. Majster-Cederbaum:
Algebraic metric spaces for modelling concurrency and communication, to appear.

[2] J.W. de Bakker, J.H.A. Warmerdam:
Metric pomset semantics for a concurrent language with recursion,
Report CS-R9033, Centre for Mathematics and Computer Science, Amsterdam, July 1990.

[3] G. Boudol, I. Castellani:
Three Equivalent Semantics for *CCS*,
Lecture Notes in Computer Science 469, Springer-Verlag, 1990.

[4] S.D. Brookes:
A Model for Communicating Sequential Processes,
Report CMU-CS 83-149, Carnegie-Mellon University, January 1983.

[5] S.D. Brookes, C.A.R. Hoare, A.W. Roscoe:
A Theory of Communicating Sequential Processes, Journal of the ACM, Vol. 31, No. 3, July 1984.

[6] S.D. Brookes, A.W. Roscoe:
An Improved Failures Model for Communicating Processes,
Seminar on Concurrency, Lecture Notes in Computer Science 197, Springer-Verlag, 1985.

[7] P. Degano, R. De Nicola, U. Montanari:
On the Consistency of 'Truly Concurrent' Operational and Denotational Semantics, Proc. Symposium on Logic in Computer Science, Edinburgh, pp 133-141, 1988.

[8] U. Goltz:
On Representing *CCS* Programs as Finite Petri Nets,
Proc. MFCS 88, Lecture Notes in Computer Science 324, Springer-Verlag, pp 339-350, 1988.

[9] U. Goltz, R. Loogan:
Modelling Nondeterministic Concurrent Processes with Event Structures,
Fundamenta Informaticae, Vol. 14, No. 1, pp 39-73, 1991.

[10] U. Goltz, A. Mycroft:
On the Relationship of *CCS* and Petri Nets, Proc. ICALP 84, Lecture Notes in Computer Science 172, Springer-Verlag, 1984.

[11] H. Hahn:
Reelle Funktionen, Chelsea, New York, 1948.

[12] C.A.R. Hoare:
Communicating Sequential Processes, Prentice Hall, 1985.

[13] J.N. Kok, J.J.M.M. Rutten:
Contractions in Comparing Concurrency Semantics,
Report CS-R8755, Centre for Mathematics and Computer Science, Amsterdam, November 1987.

[14] M. Majster-Cederbaum, F. Zetzsche:
The comparison of a cpo-based with a cms-based semantics for *CSP*, Manuskripte der Fakultät für Mathematik und Informatik, Mannheim, Reihe Informatik 2/91, to appear in Theoretical Computer Science.

[15] R. Milner:
A Calculus of Communicating Systems,
Lecture Notes in Computer Science 92, Springer-Verlag, 1980.

[16] R. Milner:
Lectures on a Calculus of Communicating Systems, Seminar on Concurrency, Lecture Notes in Computer Science 197, Springer-Verlag, 1985.

[17] E.R. Olderog:
TCSP : Theory of Communicating Sequential Processes, Advances in Petri-Nets 1986, Lecture Notes in Computer Science 255, Springer-Verlag, pp 441-465, 1987.

[18] E.R. Olderog:
Operational Petri-Net Semantics for *CCSP*, Advances in Petri-Nets 1987, Lecture Notes in Computer Science 266, Springer-Verlag, pp 196-223, 1987.

[19] V. Pratt:
Modelling Concurrency with Partial Orders,
International Journal of Parallel Programming 15, pp 33-71, 1986.

[20] W. Reisig:
Partial Order Semantics versus Interleaving Semantics for *CSP*-like Languages and its Impact on Fairness,
Proc. ICALP 84, Lecture Notes in Computer Science 172, Springer-Verlag, pp 403-413, 1984.

[21] F.W. Vaandragger:
A Simple Definition for Parallel Composition of Prime Event Structures,
Techn. Report CS-R 8903, Centre for Mathematics and Computer Science, Amsterdam, 1989.

[22] G. Winskel:
Event Structure Semantics for *CCS* and Related Languages,
Proc. ICALP 82, Lecture Notes in Computer Science 140, Springer-Verlag, pp 561-576, 1982.

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