On generating synchronous interworkings from PSF process traces

S.F.M. van Vlijmen
Programming Research Group, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

A. van Waveren
Programming Research Group, University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands

Synchronous interworkings are a specification formalism on their own right, but might also be of help when analysing processes and process traces. In this paper the processes are specified in PSF and traces are PSF process traces. The main part of this paper is concerned with the definition of the notions needed to perform the generation of interworkings from traces and the conditions under which the generation is feasible. Furthermore we give a pseudo-code algorithm for the interworking generating function, give examples based on a trace of the Alternating Bit Protocol and discuss informally the role synchronous interworkings could have in the specification process.

1990 Mathematics Subject Classification: 68Q10, 68Q45, 68U20.
1987 CR Categories: D.1.3 Concurrent programming, D.2.1 Methodologies, F.1.2 Parallelism, D.2.5 Tracing.
Key words & Phrases: synchronous interworkings, message sequence charts, PSF, process algebra, process traces.

Introduction

Interworkings are descriptions of communication behaviour of so called entities, and they are used for requirements analysis within PKI Nürnberg (Franz Wild, Philips Kommunikations Industrie). By an entity one can think of a process, a system or a part of a system. Interworkings can be represented graphically and textually. We will use the graphical representation in the following to illustrate our examples. In a graphical interworking an entity is represented by a vertical line, communication is represented by a horizontal arrow, marked with a message, running from one entity to another. One can also have actions which are

placed upon the 'entity lines' (see figure 1 of section 3 for examples). Precise syntax and semantics of the synchronous interworkings used within Philips can be found in [MWW92]. Besides the graphical notation we will also use the semantics from [MWW92] to denote synchronous interworkings.

A technique similar to interworkings are the so called Message Sequence Charts, see the work of Rudolph [R92]. Message Sequence Charts are used in combination with SDL (Specification and Description Language). The CCITT (Comité Consultatif International Télégrafique et Téléphonique) is currently working on a standardization of these techniques. In the following we will use the word 'interworking' for short to denote synchronous interworkings.

In the PSF language, see [MV89], [MV90] and [M91], one can specify parallel processes that communicate synchronously. It is our aim to translate traces of a PSF process, that have been produced with the PSF simulator [V91], to interworkings. This could be useful when interworkings are used for a first specification of a system and PSF for the second (more detailed) specification. Interworkings generated from traces of this second specification can then be matched with the interworkings from the initial specification. In this way, a consistency check between both specifications can be performed.

Intuitively, it seems feasible to associate PSF processes with entities and PSF communication actions with communications, but some choices have to be made. For instance, it is not a priori clear how to associate processes in a PSF text with entities. The text is not always as nicely formatted as we will require, see definition 2.4 and 2.7 below. Furthermore, direction in a communication is not a PSF notion. Additional information about the meaning of a specification is needed here. Although PSF plays a central role in this paper, the formal notions — definitions and propositions — are really about ACP^{τ} [BW90, ch.5]. This is because full PSF with all its mechanisms for modularisation, parametrisations etc. is much to complex as a formal starting point.

In section 1 we will give an introduction to the work of Mauw, van Wijk and Winter, especially BPA_{iw} [MWW92]. In section 2 we will give definitions of all important notions we need, culminating in the definition of the function *iw* that generates an interworking from a trace. In section 3 we will prove some propositions about the definitions in section 2. In section 4 we will draw some conclusions about the feasibility of the notions discussed in this paper. Section 4 is followed by three appendices. In appendix A we will give an algorithm in pseudo-code for the function *iw*. In appendix B we will give an example of the generation of interworkings out of traces. The specification used, OBP', is a slight adaptation of the One Bit Sliding Window Protocol (OBP) in [W92]. OBP' is given in Appendix C.

Section 1 Preliminaries

A considerable part of the text in this section can literally be found in [MWW92].

Extension of *BPA*[1]

First an extension of *BPA* is defined.

$x + \delta = x$		$\neg$isdelta(a)	if $a \not\equiv \delta$
$\delta x = \delta$		isdelta(δ)	
		isdelta(ax) = isdelta(a)	
		isdelta($x + y$) = isdelta(x) $\wedge$ isdelta(y)	
$\partial_H(a) = a$	if $a \notin$ H	$\neg$df(δ)	
$\partial_H(a) = \delta$	if $a \in$ H	df(a)	if $a \not\equiv \delta$
$\partial_H(x + y) = \partial_H(x) + \partial_H(y)$		df(ax) = df(x)	if $a \not\equiv \delta$
$\partial_H(xy) = \partial_H(x) \cdot \partial_H(y)$		df($x + y$) = df(x) $\wedge$ df(y)	if $\neg$isdelta(x) and $\neg$isdelta(y)

BPA extended with the above stated operators, predicates and axioms is called $BPA_{\delta,\partial_H,df}$.

Actions

In the setting of interworkings, [MWW92] only considers communication actions of a fixed format and primitive actions of a fixed format, so they specialize the set of atomic actions. Let *EID* and *MID* be finite sets, the elements of which are called entity identifiers and message identifiers respectively. [MWW92] defines the following collection of communication actions.

$A_c = \{c(p, q, m) \mid p, q \in EID, m \in MID\}$

Action c(p, q, m) means that entity p sends a message m to entity q. Furthermore [MWW92] defines a set of primitive actions. Let *AN*, *TID* and *DID* be finite sets representing primitive action names, timer identifiers and duration identifiers, respectively. Then the set of primitive actions is defined as follows.

$$\begin{aligned} A_p = \quad & \{act(p, an) \mid p \in EID, an \in AN\} \cup \\ & \{lost(p, q, m) \mid p, q \in EID, m \in MID\} \cup \\ & \{timerset(p, t, d) \mid p \in EID, t \in TID, d \in DID\} \cup \\ & \{timeout(p, t) \mid p \in EID, t \in TID\} \end{aligned}$$

Primitive action act(p, an) is interpreted as the fact that entity p executes the non-communication action an. If this collection of atomic actions is extended with δ we get A_δ.

$A = A_c \cup A_p$ and $A_\delta = A \cup \{\delta\}$

[1] Basic Process Algebra [BW90, ch.2].

Unless stated differently, variables a, b range over A_δ in this section. Furthermore, an auxiliary function E is needed, which determines the entities involved in an action.

$$E(\delta) = \varnothing$$
$$E(c(p, q, m)) = \{p, q\}$$
$$E(act(p, an)) = \{p\}$$
$$E(lost(p, q, m)) = \{p\}$$
$$E(timerset(p, t, d)) = \{p\}$$
$$E(timeout(p, t)) = \{p\}$$
$$E(ax) = E(a) \cup E(x) \qquad \text{if } a \not\equiv \delta$$
$$E(x + y) = E(x) \cup E(y)$$

Interworking Sequencing

The interworking sequencing of two processes (notation $\circ_{iw}$) is their interleaved composition with the restriction that actions involving common entities are performed first by the left process and then by the right process. [MWW92] defines the set of actions generated by the entities of a given process with: $\alpha_E(x) = \{\ a \in A \mid E(a) \cap E(x) \neq \varnothing\ \}$. The definition of the interworking sequencing resembles the definition of the communication free merge from *PA* [BW90]. [MWW92] uses the auxiliary operators left sequencing ($\mathbf{L}\circ_{iw}$) and right sequencing ($\mathbf{R}\circ_{iw}$) which have the following intuitive meaning. The left sequencing of two processes means that the left operand is forced to do the first step. The right sequencing of two processes means that the right operand has to do the first step, but it may only execute this step if it is not blocked by some action from the left operand which involves the same entity. The encapsulation operator (∂_H) is used for blocking unwanted actions.

$$x \circ_{iw} y = x\ \mathbf{L}\circ_{iw} y + x\ \mathbf{R}\circ_{iw} y$$
$$a\ \mathbf{L}\circ_{iw} x = ax$$
$$ax\ \mathbf{L}\circ_{iw} y = a(x \circ_{iw} y)$$
$$(x + y)\ \mathbf{L}\circ_{iw} z = x\ \mathbf{L}\circ_{iw} z + y\ \mathbf{L}\circ_{iw} z$$
$$x\ \mathbf{R}\circ_{iw} a = \partial_{\alpha_E(x)}(a) \cdot x$$
$$x\ \mathbf{R}\circ_{iw} ay = \partial_{\alpha_E(x)}(a) \cdot (x \circ_{iw} y)$$
$$x\ \mathbf{R}\circ_{iw} (y + z) = x\ \mathbf{R}\circ_{iw} y + x\ \mathbf{R}\circ_{iw} z$$

Interworking Merge

The interworking merge of two processes (notation $\|_{iw}$) is their interleaved composition, except that the processes are forced to synchronize on a set of atomic actions. This set consists of the actions concerning every pair of entities which the processes have in common. First the S-interworking merge (notation $\|^S_{iw}$) for a subset S of A is defined. As in the definition of the parallel composition in *ACP* [MWW92] needs two auxiliary operators: the left interworking merge ($\lfloor\!\lfloor^S_{iw}$) and the synchronization interworking merge ($|^S_{iw}$). The left interworking merge forces that the first step is

taken from the left argument, provided that this action is not an element of the set S. The synchronization interworking merge can only execute an action if both operands can perform this same action and this action is an element of the set S. The function γ_S determines whether two actions have to synchronize.

$$x \parallel_{iw}^S y = x \lfloor\!\lfloor_{iw}^S y + y \lfloor\!\lfloor_{iw}^S x + x \mid_{iw}^S y$$
$$a \lfloor\!\lfloor_{iw}^S x = \partial_S(ax)$$
$$ax \lfloor\!\lfloor_{iw}^S y = \partial_S(a) \cdot (x \parallel_{iw}^S y)$$
$$(x + y) \lfloor\!\lfloor_{iw}^S z = x \lfloor\!\lfloor_{iw}^S z + y \lfloor\!\lfloor_{iw}^S z$$
$$a \mid_{iw}^S b = \gamma_S(a, b)$$
$$ax \mid_{iw}^S b = \gamma_S(a, b) \cdot \partial_S(x)$$
$$a \mid_{iw}^S bx = \gamma_S(a, b) \cdot \partial_S(x)$$
$$ax \mid_{iw}^S by = \gamma_S(a, b) \cdot (x \parallel_{iw}^S y)$$
$$(x + y) \mid_{iw}^S z = x \mid_{iw}^S z + y \mid_{iw}^S z$$
$$x \mid_{iw}^S (y + z) = x \mid_{iw}^S y + x \mid_{iw}^S z$$
$$\gamma_S(a, b) = \begin{cases} a \text{ if } a = b \wedge a \in S \\ \delta \text{ otherwise} \end{cases}$$

Define the function α_{CE}, which determines the actions performed by common entities as follows.

$$\alpha_{\mathrm{CE}}(x, y) = \{c(p, q, m) \in A \mid p, q \in E(x) \cap E(y), m \in MID\}$$

Then we define the interworking merge as follows.

$$x \parallel_{iw} y = x \parallel_{iw}^{\alpha_{\mathrm{CE}}(x,y)} y$$

Interworkings

The theory $BPA_{\delta,\partial_H,df}$ with the two operators for interworking merge and interworking sequencing will be called BPA_{iw}. An *interworking* is a process which can be constructed from atomic actions in A_δ only and applications of the interworking sequencing and interworking merge operator. The set of interworkings is denoted by IW. A process which can be constructed of atomic actions in A_δ only and the interworking sequencing operator is called a locally deterministic interworking. The set of locally deterministic interworkings is called IW^{ld}.

1.1 Proposition

$E(x) \cap E(y) = \emptyset \Rightarrow x \circ_{iw} y = y \circ_{iw} x$

A process x is *deterministic* if it is of the following form.

$$\sum_{i \in I} a_i \cdot x_i + \sum_{j \in J} b_j$$

where it is required that all a_k are different for $k \in I \cup J$ and all x_i are deterministic.

1.2 Proposition

Every interworking is deterministic

1.3 Definition (tr, the function that gives the traces of a process in BPA_{iw})

$\mathrm{tr}(a) = \{a\}$

$\mathrm{tr}(ax) = \{at \mid t \in \mathrm{tr}(x)\}$ $\quad a \in \mathrm{A}$, x a process

$\mathrm{tr}(x + y) = \mathrm{tr}(x) \cup \mathrm{tr}(y)$ $\quad x \neq \delta$, $y \neq \delta$

T_{iw} is the set of traces of all interworking processes. A trace in T_{iw} is an element of $\mathrm{A}^+ \cup \mathrm{A}^+\delta \cup \{\delta\}$.

Section 2 Definitions

In this section we denote with A the set of all atomic actions, τ and $\delta \notin \mathrm{A}$. Here A is not specialized as in section 1.

2.1 Definition (trijective communication function γ [V90, definition 3.1.1])

A communication function γ is trijective if three pairwise disjoint subsets $R, S, C \subseteq \mathrm{A}$ can be given, and bijections $^{-}: R \to S$ and $^{\circ}: R \to C$ such that for every $a, b, c \in \mathrm{A}$:

$$\gamma(a, b) = c \;\Rightarrow\; (a \in R \wedge b = \bar{a} \wedge c = a^{\circ}) \vee (b \in R \wedge a = \bar{b} \wedge c = b^{\circ}).$$

2.1.1 Remark ([V90, Remark 3.1.2])

Observe that a trijective communication function γ satisfies the following three properties, and that each γ satisfying these properties is trijective ($a, b, c, d \in \mathrm{A}$):

1. $\gamma(a, a) = \delta$;
2. if $\gamma(a, b) \neq \delta$ and $\gamma(a, c) \neq \delta$ then $b = c$ (γ is 'monogamous');
3. if $\gamma(a, b) = \gamma(c, d) \neq \delta$ then $a = c$ or $a = d$ (γ is 'injective').

Observe further that a trijective γ satisfies $\gamma(\gamma(a, b), c) = \delta$ ('handshaking').

If γ is trijective and $\gamma(a, b) = c$, we say $\gamma^{-1}(c) = \{a, b\}$.

2.2 Definition (communication action)

Let $\mathrm{Arg}(\gamma) = \{a \in \mathrm{A} \mid \exists b \in \mathrm{A}\ (\gamma(a, b) \neq \delta)\}$.

Let $\mathrm{Im}(\gamma) = \{c \in \mathrm{A} \mid \exists\, a, b \in \mathrm{Arg}(\gamma)\ (\gamma(a, b) = c)\}$. (Note that $\delta \notin \mathrm{A}$.)

A communication action is an element of $\mathrm{Im}(\gamma)$.

2.3 Definition ($\alpha(p)$, alphabet of a process)
In the context of a ACP$^\tau$ specification a process p is identified with the process that is the solution for the process variable with name p in the specification. The alphabet of a process can be determined as defined in [BW90, ch.6].

2.4 Definition (type 1 restricted specification)
A type 1 restricted specification is a ACP$^\tau$ specification s with a so called 'top-process' of the form $P = \partial_H(p_1 \parallel .. \parallel p_n)$ $(n \geq 1)$, communication function γ and encapsulation set H. Furthermore s has to obey the following restrictions:

1. the p_i $(1 \leq i \leq n)$ are definable processes[1];
2. γ is trijective;
3. $\forall\, c \in \mathrm{Im}(\gamma)\ (c \notin \alpha(p_i)\ (1 \leq i \leq n))$.

2.4.1 Remarks

1. If p is given as the solution of a finite linear specification then p is a regular process [BW90]. The alphabet of a regular process is computable from its specification (this holds in the absence of data, in a PSF setting this will be more complicated).
2. Restriction 2 and 3 make it straightforward to determine which subsets of processes of $\{p_i \mid 1 \leq i \leq n\}$ participate in a communication found in a trace.
3. In definition 2.7 we will allow limited abstraction when we introduce type 2 restricted specifications.

2.5 Definition (*P*-entity and *P*-partition)
Given a type 1 restricted specification s with top-process $P = \partial_H(p_1 \parallel .. \parallel p_n)$. A *P*-partition E is a partition of $\{p_i \mid 1 \leq i \leq n\}$, which obeys the following constraint.

$$\forall e_1, e_2 \in E\ (e_1 \neq e_2 \Rightarrow \bigcup_{p \in e_1} \alpha(p) \cap \bigcup_{p \in e_2} \alpha(p) = \varnothing).$$

A *P*-entity e is an element of a *P*-partition.

2.6 Definition (*entity*, the function that assigns *P*-entity names to atomic actions)
Given a type 1 restricted specification s with top-process P and given a *P*-partition E for s. Let furthermore be given a set of names N, $N \subseteq EID$, and a bijection φ: $E \to N$.

The function $entity_{E,\varphi}$: $\bigcup_{e \in E} (\bigcup_{p \in e} \alpha(p)) \to \varphi(E)$ is defined by:

$\forall a \in \bigcup_{e \in E} (\bigcup_{p \in e} \alpha(p))\ (\forall n \in \varphi(E)\ (entity_{E,\varphi}(a) = n \Leftrightarrow a \in \bigcup_{p \in \varphi^{-1}(n)} \alpha(p)))$.

(The constraint in definition 2.5 above and the bijectivity of φ ensure that $entity_{E,\varphi}$ is a function.)

---

[1] A process is definable if it is recursively definable over + and · using guarded recursion [BW90, definition 2.4.6].

2.7 Definition (type 2 restricted specification)
Given a type 1 restricted specification s with top process P. Let furthermore be given a P-partition E, a set of names N, a bijection φ: E $\rightarrow$ N, and a set of communication actions $\mathrm{Im}_{E,\varphi}(\gamma) = \{c \in A \mid \exists\, a, b \in \mathrm{Arg}(\gamma)\ (\gamma(a, b) = c \ \wedge\ entity_{E,\varphi}(a) \neq entity_{E,\varphi}(b))\}$. With $\mathrm{Im}_{E,\varphi}(\gamma)$ we catch all the communication actions between different P-entities. Abstraction from these actions can lead via traces to interworkings that allow an interleaving of actions not found in any trace of P (this is examplified in example 3.10 below).

Suppose I is a subset of the set of all atomic actions A. Let s' be the specification we get from s by taking $\tau_I(P)$ as the top-process. We say s' is type 2 restricted with respect to E and φ if $I \cap \mathrm{Im}_{E,\varphi}(\gamma) = \varnothing$.

2.7.1 Remark

1. Given type 1 restricted specification s, and given P-partition E and bijection φ: E $\rightarrow$ N for s. If in s no abstraction is used, this comes down to taking the abstraction set of the abstraction operator τ equal to the empty set. So, s can be said to be type 2 restricted with respect to E and φ.
2. In the following we will drop the 'with respect to E ...' part when introducing a type 2 restricted specification; E and φ are clear from the context or irrelevant for the discussion at hand. We will even go one step further and use 'restricted specification' to denote type 1 and type 2 restricted specifications.

2.8 Definition (tr, the function that gives the traces [BW90, §6.3] of a process)
As usual λ denotes the empty trace. For process p, tr(p) is given by:

$\mathrm{tr}(\delta) = \mathrm{tr}(\tau) = \{\lambda\}$	
$\mathrm{tr}(a) = \{\lambda, a\}$	$a \in A$
$\mathrm{tr}(\tau x) = \mathrm{tr}(x)$	x a process
$\mathrm{tr}(ax) = \{\lambda\} \cup \{at \mid t \in \mathrm{tr}(x)\}$	$a \in A$, x a process
$\mathrm{tr}(x + y) = \mathrm{tr}(x) \cup \mathrm{tr}(y)$	

With tr(s) we denote the set of traces of the process P specified by restricted specification s; tr(s) = tr(P). T_{PSF} is the set of all process traces of processes specified by restricted specifications. A trace in T_{PSF} is an element of A^*.

2.9 Definition (D, the direction function for a restricted specification s)
Direction of communication is not a notion of PSF. Although it is customary to call actions *read* and *send* in specification texts this has no special semantics, but an intuitive one. In interworkings however an action c(p, q, m) has a direction. Given a communication action c in a trace t of a restricted specification s. If we translate t to an interworking w we have to decide what direction this c should get in the interworking w. D is the function for s that

determines the direction a communication action will get in interworking w. The function D is used by the function iw defined below in definition 2.11. The input to D is a set of two atomic actions and the output is an ordered pair of these atomic actions. More precisely:
$D(\{a, b\}) = (a, b)$ or (b, a), dependent upon the direction one chooses. If $\gamma(a, b) = \delta$, then $D(\{a, b\})$ is not defined.

2.10 Definition (SENφD)
SENφD is the set of quintets (s, E , N, φ, D). Here s is a type 1 restricted specification, E a P-partition for s, $N \subseteq EID$, $\varphi: E \to N$ a bijection. Or, s is a type 2 restricted specification with respect to E and $\varphi: E \to N$. D is a direction function for s.

2.11 Definition (iw, the interworking generating function)
Let $r = (s, E , N, \varphi, D)$ be a quintet in SENφD, iw_r is the function iw connected to r. The function $iw_r: tr(s) - \{\lambda\} \to IW^{ld}$ produces an interworking from a trace, and is defined as the composition of two other functions, namely $CA_r: tr(s) - \{\lambda\} \to T_{iw} - \{\lambda\}$ and $\sim: T_{iw} - \{\lambda\} \to IW^{ld}$. The function $\sim$ is straigthforward: it places the operators $\circ_{iw}$ in traces. The function CA_r translates actions in traces into the 'interworking actions': $c(p, q, m)$ and $act(p, m)$ (see section 1). The function CA_r is defined with functions we already know (D, γ^{-1}, *entity*) and a new one, a projection function we will now define.
Let π_i ($i \in \{1, 2\}$) be the projection function giving the i-th element of a pair:
$\pi_1((a, b)) = a$; $\pi_2((a, b)) = b$.
Below follow definitions for CA_r, $\sim$ and iw_r. Here t is a non empty suffix of a trace in $tr(s) - \{\lambda\}$, a an action occurring in a trace of s, x and y are element of $T_{iw} - \{\lambda\}$, b is element of the specialised set of action from section 1.

$CA_r(a) = act(entity_{E,\varphi}(a), a)$ $\qquad a \notin Im(\gamma_s)$
$CA_r(a) = c(entity_{E,\varphi}(\pi_1(D(\gamma^{-1}(a)))), entity_{E,\varphi}(\pi_2(D(\gamma^{-1}(a)))), a)$ $\qquad a \in Im(\gamma_s)$
$CA_r(at) = CA_r(a)CA_r(t)$

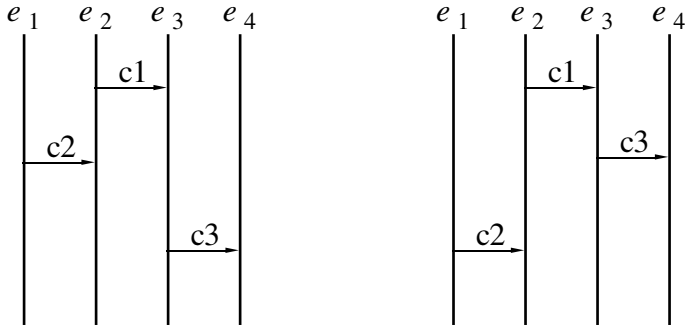
$\sim(b) = b$
$\sim(xy) = \sim(x) \circ_{iw} \sim(y)$

$iw_r(t) = \sim(CA_r(t))$

2.11.1 Remark
The function iw only operates on non-empty traces. This is because there fails to be an interworking that can naturally serve as image for trace λ. An image for the $iw(\lambda)$ would preferably be the empty process ε, however ε is not present in BPA_{iw}.

Section 3 Propositions

If we generate interworkings we do not want these interworkings to give rise to misinterpretation of the meaning of the PSF processes that led to these interworkings. This misinterpretation can happen because the set of communications in an interworking is partially ordered. In figure 1 the interworkings *i1* and *i2* have the same meaning. Action c1 precedes action c2 and action c3. Although the graphical representation might suggest that c2 and c3 are ordered they are not, because they have no entities in common: action c2 is a communication between entities e_1 and e_2, and action c3 is a communication between entities e_3 and e_4.



i1 *i2*

figure 1

Semantically *i1* and *i2* are equivalent. It follows with proposition 1.1 above, that c1 $\circ_{iw}$ c2 $\circ_{iw}$ c3 = c1 $\circ_{iw}$ c3 $\circ_{iw}$ c2. If we formed interworking *i1* from a trace c1c2c3 of a PSF process *P*, *P* should also be able to generate trace c1c3c2. Definition 3.1 states this requirement precisely.

3.1 Definition (IW-interleaving condition)
Let $r = (s, E, N, \varphi, D)$ be a quintet in SENφD, and iw_r the *iw* function connected to *r*.
Let $CA_r(\mathrm{tr}(s) - \{\lambda\}) = \{CA_r(t) \mid t \in \mathrm{tr}(s) - \{\lambda\}\}$.
We say that iw_r satisfies the IW-interleaving condition if

$$\bigcup_{t \in \mathrm{tr}(s) - \{\lambda\}} \mathrm{tr}(iw_r(t)) \;=\; CA_r(\mathrm{tr}(s) - \{\lambda\}).$$

We will now give an example in order to illustrate the notions introduced in section 2 and to help in understanding the propositions below.

3.2 Example
The example is about the quintet $r = (s, E, N, \varphi, D)$ in SENφD.

Specification *s*:

$P = \partial_H(p_1 \parallel p_2 \parallel p_3 \parallel p_4 \parallel p_5)$, $p_1 = s2$, $p_2 = s1 \cdot r2 \cdot a1$, $p_3 = r1 \cdot s3$, $p_4 = a2$, $p_5 = r3$

$H = \{r1, s1, r2, s2, r3, s3\}$

$\gamma(s1, r1) = c1$, $\gamma(s2, r2) = c2$, $\gamma(s3, r3) = c3$

P-partition E:

$E = \{\{p_1\}, \{p_2\}, \{p_3\}, \{p_4, p_5\}\}$

Set of names N:

$N = \{e_1, e_2, e_3, e_4\}$

Bijection φ that assigns names to *P*-entities:

$\varphi(\{p_1\}) = e_1$, $\varphi(\{p_2\}) = e_2$, $\varphi(\{p_3\}) = e_3$, $\varphi(\{p_4, p_5\}) = e_4$

Direction function D:

$D(\{s1, r1\}) = (s1, r1)$, $D(\{s2, r2\}) = (s2, r2)$, $D(\{s3, r3\}) = (s3, r3)$

Let T be the set of traces of maximum length in tr(*s*).

T = {a2c1c2a1c3, a2c1c2c3a1, a2c1c3c2a1,
c1c2a1c3a2, c1c2a1a2c3,
c1c2a2a1c3, c1c2a2c3a1,
c1c2c3a1a2, c1c2c3a2a1,
c1c3c2a1a2, c1c3c2a2a1, c1c3a2c2a1,
c1a2c2a1c3, c1a2c2c3a1, c1a2c3c2a1}

One can easily see that tr(*s*) – {λ} is the set of all non-empty prefixes of elements from T.

Take one of the traces in tr(*s*) – {λ}. Suppose we take *t* = c1c2a1c3, this is a prefix of trace c1c2a1c3a2. Then we have:

$\sim(CA_r(t)) = c(e_2, e_3, c1) \circ_{iw} c(e_1, e_2, c2) \circ_{iw} act(e_2, a1) \circ_{iw} c(e_3, e_4, c3)$.

In figure 2 below, interworking $\sim(CA_r(t))$ is depicted.

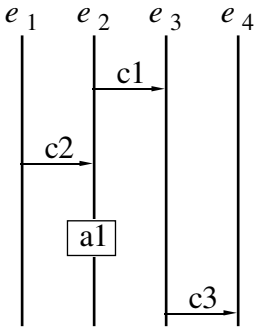


figure 2

With the axioms of BPA_{iw} we derive:

$\sim(CA_r(t)) = \ c(e_2, e_3, c1) \cdot$
$\quad (\ c(e_1, e_2, c2) \cdot$
$\quad\quad (\ act(e_2, a1) \cdot c(e_3, e_4, c3) +$
$\quad\quad\quad c(e_3, e_4, c3) \cdot act(e_2, a1)$
$\quad\quad) +$
$\quad\quad c(e_3, e_4, c3) \cdot c(e_1, e_2, c2) \cdot act(e_2, a1)$
$\quad)$

With the definition of tr in section 1 we now have:

$tr(\sim(CA_r(t))) \quad = \quad \{ \ c(e_2, e_3, c1)c(e_1, e_2, c2)act(e_2, a1)c(e_3, e_4, c3),$
$\quad\quad\quad\quad\quad\quad c(e_2, e_3, c1)c(e_1, e_2, c2)c(e_3, e_4, c3)act(e_2, a1),$
$\quad\quad\quad\quad\quad\quad c(e_2, e_3, c1)c(e_3, e_4, c3)c(e_1, e_2, c2)act(e_2, a1) \ \}$

One can easily verify that $tr(\sim(CA_r(t))) \subseteq CA_r(tr(s) - \{\lambda\})$.

3.3 Proposition

$E(x \circ_{iw} y) = E(x) \cup E(y)$ for x and $y \in IW$

Proof

We proof the proposition with induction over the length of closed terms. BPA_{iw} is a conservative extension of BPA, and so x and y have normalforms.

For $n > 0$ we take $x = x_1 + .. + x_n$, where $x_i = a_i x_i'$ if $1 \leq i \leq k$ and $x_i = a_i$ if $k < i \leq n$, $0 \leq k \leq n$. For $n = 0$ we take $x = \delta$.

For $m > 0$ we take $y = y_1 + .. + y_m$, where $y_j = b_j y_j'$ if $1 \leq j \leq k$ and $y_j = b_j$ if $k < j \leq m$, $0 \leq k \leq m$. For $m = 0$ we take $y = \delta$.

1° $(n = 0)$ $\quad x \circ_{iw} y = \delta \ \mathbf{L}\circ_{iw} y + \delta \ \mathbf{R}\circ_{iw} y = \delta y + \delta \ \mathbf{R}\circ_{iw} y = \delta \ \mathbf{R}\circ_{iw} y$

1.1 $(m = 0)$ $\quad \delta \ \mathbf{R}\circ_{iw} y = \delta \ \mathbf{R}\circ_{iw} \delta = \partial_{\alpha_E(\delta)}(\delta) \cdot \delta = \delta;$

thus $E(x \circ_{iw} y) = E(\delta) = \varnothing = \varnothing \cup \varnothing = E(\delta) \cup E(\delta) = E(x) \cup E(y)$.

1.2 $(m > 0)$ $\quad \delta \ \mathbf{R}\circ_{iw} y = \delta \ \mathbf{R}\circ_{iw} (y_1 + .. + y_m) = \delta \ \mathbf{R}\circ_{iw} y_1 + .. + \delta \ \mathbf{R}\circ_{iw} y_m$

$\delta \ \mathbf{R}\circ_{iw} b_j y_j' = \partial_{\alpha_E(\delta)}(b_j) \cdot (\delta \circ_{iw} y_j') = b_j \cdot (\delta \circ_{iw} y_j')$

$\delta \ \mathbf{R}\circ_{iw} b_j = \partial_{\alpha_E(\delta)}(b_j) \cdot \delta = b_j \cdot \delta$

Thus $E(\delta \ \mathbf{R}\circ_{iw} y_j) = E(y_j)$ $(1 \leq j \leq m)$.

Therefore $E(x \circ_{iw} y) = E(\delta \ \mathbf{R}\circ_{iw} y_1 + .. + \delta \ \mathbf{R}\circ_{iw} y_m) =$
$\varnothing \cup E(y_1) \cup .. \cup E(y_m) = E(x) \cup E(y_1 + .. + y_m) = E(x) \cup E(y)$.

2° $(n > 0)$ $\quad x \circ_{iw} y = x \ \mathbf{L}\circ_{iw} y + x \ \mathbf{R}\circ_{iw} y$

$x \ \mathbf{L}\circ_{iw} y = x_1 \ \mathbf{L}\circ_{iw} y + .. + x_n \ \mathbf{L}\circ_{iw} y$

$a_i x_i' \ \mathbf{L}\circ_{iw} y = a_i \cdot (x_i' \circ_{iw} y)$ and $a_i \ \mathbf{L}\circ_{iw} y = a_i \cdot y$

Thus $E(x_i \ \mathbf{L}\circ_{iw} y) = E(x_i) \cup E(y)$, and so $E(x \ \mathbf{L}\circ_{iw} y) = E(x) \cup E(y)$.

With the axioms in section 1 of BPA_{iw} it is evident that

$E(x \ \mathbf{R}\circ_{iw} y) \subseteq E(x) \cup E(y)$; thus $E(x \circ_{iw} y) = E(x) \cup E(y)$. ■

3.4 Notation

We introduce some notation to enhance the readibility of the propositions below. Given interworking $w = a_1 \circ_{iw} .. \circ_{iw} a_n$ in IW^{ld} ($n \geq 1$). With $(a_1 \circ_{iw} .. \circ_{iw} a_n)^{(i)}$ ($1 \leq i \leq n$) we denote the interworking we get from w by forgetting action a_i. With $(a_1 \circ_{iw} .. \circ_{iw} a_n)^{(i,j)}$ ($1 \leq i,j \leq n$) we denote the interworking we get from w by forgetting a_i and a_j. Not defined is $(a_1)^{(1)}$, but $x \cdot (a_1)^{(1)} = x$. Examples: $(a_1 \circ_{iw} a_2 \circ_{iw} a_3)^{(1)} = a_2 \circ_{iw} a_3$, $(a_1 \circ_{iw} a_2 \circ_{iw} a_3)^{(1,3)} = a_2$.
Note that $(a_1 \circ_{iw} .. \circ_{iw} a_n)^{(i,j)} = (a_1 \circ_{iw} .. \circ_{iw} a_n)^{(j,i)}$, and $(a_1 \circ_{iw} .. \circ_{iw} a_n)^{(i,i)} = (a_1 \circ_{iw} .. \circ_{iw} a_n)^{(i)}$.

3.5 Proposition

$$a_1 \circ_{iw} .. \circ_{iw} a_n = \sum_{\substack{1 \leq i \leq n \\ E(a_i) \cap E(1,i-1) = \emptyset}} a_i \cdot (a_1 \circ_{iw} .. \circ_{iw} a_n)^{(i)}, \text{ for } a_1 \circ_{iw} .. \circ_{iw} a_n \in IW^{ld} \ (n \geq 1).$$

$E(i,j) = E(a_i \circ_{iw} .. \circ_{iw} a_j)$ if $i \leq j$, otherwise $E(i,j) = \emptyset$.

Proof

We proof the proposition with induction on the length of interworking $w = a_1 \circ_{iw} .. \circ_{iw} a_n$.

1° ($n = 1$)

$$a_1 = \sum_{\substack{1 \leq i \leq 1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} a_i \cdot w^{(i)}.$$

2° ($n = m + 1$, $m \geq 1$)

$a_1 \circ_{iw} .. \circ_{iw} a_{m+1}$ { with proposition 1.1 and proposition 3.3 }

$$= \sum_{\substack{1 \leq i \leq m+1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} a_i \circ_{iw} w^{(i)}$$

$$= \sum_{\substack{1 \leq i \leq m+1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} [a_i \mathbf{L} \circ_{iw} w^{(i)} + a_i \mathbf{R} \circ_{iw} w^{(i)}]$$

$$= \sum_{\substack{1 \leq i \leq m+1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} [a_i \cdot w^{(i)} + a_i \mathbf{R} \circ_{iw} w^{(i)}] \quad \{ \text{ induction hypothesis } \}$$

$$= \sum_{\substack{1 \leq i \leq m+1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} [a_i \cdot w^{(i)} + a_i \mathbf{R} \circ_{iw} \sum_{\substack{1 \leq j \leq i-1 \\ E(a_j) \cap E(1,j-1) = \emptyset}} a_j \cdot w^{(i,j)} + a_i \mathbf{R} \circ_{iw} \sum_{\substack{i+1 \leq j \leq m+1 \\ E(a_j) \cap (E(1,i-1) \cup E(i+1,j-1)) = \emptyset}} a_j \cdot w^{(i,j)}]$$

$$= \sum_{\substack{1 \leq i \leq m+1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} [a_i \cdot w^{(i)} + \sum_{\substack{1 \leq j \leq i-1 \\ E(a_j) \cap E(1,j-1) = \emptyset}} a_i \mathbf{R} \circ_{iw} a_j \cdot w^{(i,j)} + \sum_{\substack{i+1 \leq j \leq m+1 \\ E(a_j) \cap (E(1,i-1) \cup E(i+1,j-1)) = \emptyset}} a_i \mathbf{R} \circ_{iw} a_j \cdot w^{(i,j)}]$$

$$= \sum_{\substack{1 \leq i \leq m+1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} [a_i \cdot w^{(i)} + \sum_{\substack{1 \leq j \leq i-1 \\ E(a_j) \cap E(1,j-1) = \emptyset}} \partial_{\alpha_E(a_i)}(a_j) \cdot (a_i \circ_{iw} w^{(i,j)}) + \sum_{\substack{i+1 \leq j \leq m+1 \\ E(a_j) \cap (E(1,i-1) \cup E(i+1,j-1)) = \emptyset}} \partial_{\alpha_E(a_i)}(a_j) \cdot (a_i \circ_{iw} w^{(i,j)})]$$

$$= \sum_{\substack{1 \leq i \leq m+1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} [a_i \cdot w^{(i)} + \sum_{\substack{1 \leq j \leq m+1 \\ E(a_j) \cap E(1,j-1) = \emptyset}} \partial_{\alpha_E(a_i)}(a_j) \cdot w^{(j)}]$$

{ $\partial_{\alpha_E(a_i)}(a_j)$ is a_j or δ. If it is a_j, then the term $\partial_{\alpha_E(a_i)}(a_j) \cdot w^{(j)}$ is already present in the first summation. If it is δ, then it is absorbed. }

$$= \sum_{\substack{1 \le i \le m+1 \\ E(a_i) \cap E(1,i-1) = \emptyset}} a_i \cdot w^{(i)}$$

■

3.6 Proposition

Given $r = (s, E, N, \varphi, D)$ in SENφD with s type 1 restricted and with top-process P.

If $a_1 \,..\, a_n \in \mathrm{tr}(P)$, then there exists a $r' = (s', E', N', \varphi', D')$ in SENφD with s' type 1 restricted and with top-process P' such that $P \xrightarrow{a_1} P'$, $a_2 \,..\, a_n \in \mathrm{tr}(P')$ and $\forall t \in \mathrm{tr}(P') - \{\lambda\}$ $(CA_r(t) = CA_{r'}(t))$

Proof

Suppose $P = \partial_H(p_1 \parallel .. \parallel p_m)$ $(m \ge 1)$.

$\mathrm{tr}(P) = \mathrm{tr}(\partial_H(p_1 \parallel .. \parallel p_m)) =$ {expansion theorem [BW90, theorem 4.3.5]}

$$\mathrm{tr}(\partial_H(\sum_{1 \le i \le m} p_i \between (\parallel_{\substack{1 \le k \le m \\ k \ne i}} p_k) + \sum_{1 \le i < j \le m} (p_i \mid p_j) \between (\parallel_{\substack{1 \le k \le m \\ k \ne i,\ k \ne j}} p_k))) =$$

$$\bigcup_{1 \le i \le m} \mathrm{tr}(\partial_H(p_i \between (\parallel_{\substack{1 \le k \le m \\ k \ne i}} p_k))) \cup \bigcup_{1 \le i < j \le m} \mathrm{tr}(\partial_H((p_i \mid p_j) \between (\parallel_{\substack{1 \le k \le m \\ k \ne i,\ k \ne j}} p_k)))$$

It is clear that there is a trace set $\mathrm{tr}(\partial_H(X))$ in the union above such that $a_1 \,..\, a_n \in \mathrm{tr}(\partial_H(X))$.

1° $X = p_i \between (\parallel_{\substack{1 \le k \le m \\ k \ne i}} p_k)$ $(1 \le i \le m)$.

1.1 if $p_i \xrightarrow{a_1} \surd$, then $X \xrightarrow{a_1} \parallel_{\substack{1 \le k \le m \\ k \ne i}} p_k$.

1.2 if $p_i \xrightarrow{a_1} p_i'$, then $X \xrightarrow{a_1} p_i' \parallel (\parallel_{\substack{1 \le k \le m \\ k \ne i}} p_k)$.

2° $X = (p_i \mid p_j) \between (\parallel_{\substack{1 \le k \le m \\ k \ne i,\ k \ne j}} p_k)$ $(1 \le i < j \le m)$ and suppose $a_1 = \gamma(a_1', a_1'')$.

2.1 if $p_i \xrightarrow{a_1'} \surd$ and $p_j \xrightarrow{a_1''} \surd$, then $X \xrightarrow{a_1} \parallel_{\substack{1 \le k \le m \\ k \ne i,\ k \ne j}} p_k$.

2.2 if $p_i \xrightarrow{a_1'} p_i'$ and $p_j \xrightarrow{a_1''} \surd$, then $X \xrightarrow{a_1} p_i' \parallel (\parallel_{\substack{1 \le k \le m \\ k \ne i,\ k \ne j}} p_k)$.

2.3 if $p_i \xrightarrow{a_1'} \surd$ and $p_j \xrightarrow{a_1''} p_j'$, then $X \xrightarrow{a_1} p_j' \parallel (\parallel_{\substack{1 \le k \le m \\ k \ne i,\ k \ne j}} p_k)$.

2.4 if $p_i \xrightarrow{a_1'} p_i'$ and $p_j \xrightarrow{a_1''} p_j'$, then $X \xrightarrow{a_1} p_i' \parallel p_j' \parallel (\parallel_{\substack{1 \le k \le m \\ k \ne i,\ k \ne j}} p_k)$.

Take $P' = \partial_H(Q)$, where Q is the process that remains after P has performed a_1 as fully analysed in the situations 1.1, 1.2, 2.1 to 2.4 above. Now we will construct E', N', φ' and D'. Take D' = D. It is obvious we that we can construct a P'-partition E' such that

$\forall e' \in E'\ (\exists!\ e \in E\ (\bigcup_{p \in e'} \alpha(p) \subseteq \bigcup_{p \in e} \alpha(p)))$. (*)

If |E'| = |E|, then we take N = N' and define φ' by

$\forall e' \in E'\ (\forall n' \in N'\ (\varphi'(e') = n' \Leftrightarrow \exists e \in E\ (\varphi(e) = n'$ and $\bigcup_{p \in e'} \alpha(p) \subseteq \bigcup_{p \in e} \alpha(p))))$. Note that $|E'| \not> |E|$. We however can have $|E'| < |E|$. If we have in situation 1.1 above that $\{p_i\} \in E$, then $|E'| = |E| - 1$. If we have in situation 2.1 above that $\{p_i\} \in E$ and $\{p_j\} \in E$, then $|E'| = |E| - 2$. Note that $|E'| \not< |E| - 2$. If $|E'| < |E|$ we construct N' from N by leaving out the names $\varphi^{-1}(e)$ of the *P*-entities *e* for which there is not an *e'* in E' with $\bigcup_{p \in e'} \alpha(p) \subseteq \bigcup_{p \in e} \alpha(p)$. Construct φ' now as above for the case |E'| = |E|.

Now we have to proof: $\forall t \in tr(P') - \{\lambda\}\ (CA_r(t) = CA_{r'}(t))$. Suppose $d_1 .. d_k \in tr(P')$ ($k \geq 1$). The proof comes down to showing: $CA_r(d_i) = CA_{r'}(d_i)$ ($1 \leq i \leq k$). Suppose $d_i \notin Im(\gamma)$, $d_i \in \bigcup_{p \in e} \alpha(p)$ and $d_i \in \bigcup_{p \in e'} \alpha(p)$ for $e \in E$ and $e' \in E'$. It is clear there are such *e* and *e'* otherwise E and E' are not partitions. With (*) and because E is a *P*-partition we have $\bigcup_{p \in e'} \alpha(p) \subseteq \bigcup_{p \in e} \alpha(p)$. Now it follows with the construction of φ' and the definition of the function *entity* (def. 2.6) that $entity_{E',\varphi'}(d_i) = entity_{E,\varphi}(d_i)$.

In the situation where $d_i \in Im(\gamma)$ and $d_i = \gamma(d_i', d_i'')$ the proof is analogously for d_i' and d_i''.

■

3.7 Example

As an introduction to propostion 3.8, take trace *t* = c1c2a1c3 from example 3.2. Here $E(iw_r(c3)) \cap E(iw_r(c2a1)) = E(c(e_3, e_4, c3)) \cap E(c(e_1, e_2, c2) \circ_{iw} act(e_2, a1)) = \varnothing$. With proposition 3.8 below, c1c3c2a1 should also be a trace in tr(*s*) – {λ}. One can easily verify that this is the case in example 3.2.

3.8 Proposition

Given quintet *r* = (*s*, E, N, φ, D) in SENφD with *s* type 1 restricted and *s* has top-process $P = \partial_H(p_1 \parallel .. \parallel p_m)$ ($m \geq 1$).

If $a_1 .. a_n \in tr(P)$ and $E(iw_r(a_i)) \cap E(iw_r(a_h .. a_{i-1})) = \varnothing$ ($1 \leq h \leq i \leq n$),

then $a_1 .. a_{h-1} a_i a_h .. a_{i-1} a_{i+1} .. a_n \in tr(P)$. (We take $E(iw_r(a_j .. a_{j-1})) = \varnothing$.)

Proof

We proof the proposition with induction on the length n of the trace. If n = 1 or if $n \geq 2$ and h = i the proposition can easily be seen correct.

Suppose $n \geq 2$ and $h < i$. Now we have to consider two situations: $1 < h$ and h = 1.

1° $(1 < h)$

With proposition 3.6 it follows there exists a quintet $(s', E', N', \varphi', D')$ in SENφD such that s' is type 1 restricted and with top-process P' such that $P \xrightarrow{a_1} P'$, $a_2 \,..\, a_n \in \mathrm{tr}(P')$ and $\forall t \in \mathrm{tr}(P') - \{\lambda\}$ $(CA_r(t) = CA_{r'}(t))$. Thus also $E(iw_{r'}(a_i)) \cap E(iw_{r'}(a_h \,..\, a_{i-1})) = \varnothing$. With the inductionhypothesis we have $a_2 \,..\, a_{h-1}a_i a_h \,..\, a_{i-1}a_{i+1} \,..\, a_n \in \mathrm{tr}(P')$ and thus $a_1 a_2 \,..\, a_{h-1}a_i a_h \,..\, a_{i-1}a_{i+1} \,..\, a_n \in \mathrm{tr}(P)$.

2° $(h = 1)$

From the fact that $E(iw_r(a_i)) \cap E(iw_r(a_{h=1} \,..\, a_{i-1})) = \varnothing$ we have that also $E(iw_r(a_i)) \cap E(iw_r(a_2 \,..\, a_{i-1})) = \varnothing$. With 1° it follows that $a_1 a_i a_2 \,..\, a_{i-1}a_{i+1} \,..\, a_n \in \mathrm{tr}(P)$. Now we only have to proof that we can also 'exchange' a_1 and a_i.

$\mathrm{tr}(P) = \mathrm{tr}(\partial_H(p_1 \parallel .. \parallel p_m)) =$ {expansion theorem [BW90, theorem 4.3.5]}

$$\mathrm{tr}\Big(\partial_H\Big(\sum_{1\le i\le m} p_i \between \Big(\parallel_{\substack{1\le k\le m\\ k\ne i}} p_k\Big) + \sum_{1\le i<j\le m} (p_i \mid p_j) \between \Big(\parallel_{\substack{1\le k\le m\\ k\ne i,\ k\ne j}} p_k\Big)\Big)\Big) =$$

$$\bigcup_{1\le i\le m} \mathrm{tr}\Big(\partial_H\Big(p_i \between \Big(\parallel_{\substack{1\le k\le m\\ k\ne i}} p_k\Big)\Big)\Big) \;\cup\; \bigcup_{1\le i<j\le m} \mathrm{tr}\Big(\partial_H\Big((p_i \mid p_j) \between \Big(\parallel_{\substack{1\le k\le m\\ k\ne i,\ k\ne j}} p_k\Big)\Big)\Big)$$

We only consider the most complex situation where a_1 and a_i both are communication actions. For readability, put $a_1 = a$ and $a_i = b$. Suppose $a = \gamma(a', a'')$, $b = \gamma(b', b'')$, $a' \in \alpha(p_i)$ $(1 \le i \le m)$, $a'' \in \alpha(p_j)$ $(1 \le j \le m,\ i \ne j)$, $b' \in \alpha(p_k)$ $(1 \le k \le m)$ and $b'' \in \alpha(p_l)$ $(1 \le l \le m,\ k \ne l)$. Because $E(iw_r(a)) \cap E(iw_r(b)) = \varnothing$ we have $\{i, j\} \cap \{k, l\} = \varnothing$. Because b is found in a trace we have $\partial_H(p_k \mid p_l) \ne \delta$. Now it follows that in the union above there must be a set $\mathrm{tr}\Big(\partial_H\Big((p_k \mid p_l) \between (p_i \parallel p_j \parallel \Big(\parallel_{\substack{1\le h\le m\\ h\notin\{i,j,k,l\}}} p_h\Big)\Big)\Big)\Big)$.

It is obvious that $baa_2 \,..\, a_{i-1}a_{i+1} \,..\, a_n = a_i a_1 a_2 \,..\, a_{i-1}a_{i+1} \,..\, a_n$ is an element of this set.

■

3.9 Proposition

Given $r = (s, E, N, \varphi, D)$ in SENφD with s type 1 restricted, then iw_r satisfies

$$\bigcup_{t\in \mathrm{tr}(s)-\{\lambda\}} \mathrm{tr}(iw_r(t)) \;\subseteq\; CA_r(\mathrm{tr}(s)-\{\lambda\}).$$

Proof

Suppose we have a trace $t \in \mathrm{tr}(s)-\{\lambda\}$. We proof with induction on the length of t that $\mathrm{tr}(iw_r(t)) \subseteq CA_r(\mathrm{tr}(s)-\{\lambda\})$. If $|t| = 1$, then this is obvious. Suppose $t = a_1 \,..\, a_n$ $(n \ge 2)$.

$\mathrm{tr}(iw_r(a_1 \,..\, a_n)) = \mathrm{tr}(\sim(CA_r(a_1) \,..\, CA_r(a_n))) = \mathrm{tr}(CA_r(a_1) \circ_{iw} .. \circ_{iw} CA_r(a_n)) =$ {prop. 3.5}

$$\mathrm{tr}\Big(\sum_{\substack{1\le i\le n\\ E(iw_r(a_i))\cap E(iw_r(a_1 \,..\, a_{i-1}))=\varnothing}} CA_r(a_i) \cdot (CA_r(a_1) \circ_{iw} .. \circ_{iw} CA_r(a_n))^{(i)}\Big) =$$ {def. tr}

$$\bigcup_{\substack{1\le i\le n\\ E(iw_r(a_i))\cap E(iw_r(a_1 \,..\, a_{i-1}))=\varnothing}} \{CA_r(a_i)u \mid u \in \mathrm{tr}((CA_r(a_1) \circ_{iw} .. \circ_{iw} CA_r(a_n))^{(i)})\} \qquad (*)$$

Suppose for j $(1 \le j \le n)$ we have $E(iw_r(a_j)) \cap E(iw_r(a_1 \,..\, a_{j-1})) = \varnothing$ (there is always such an index j: one can always take j = 1). With proposition 3.8 we have that $a_j a_1 \,..\, a_{j-1}a_{j+1} \,..$

$a_n \in \mathrm{tr}(P)$. With proposition 3.6 follows that there is a $r' = (s', E', N', \varphi', D')$ in SENφD such that s' is type 1 restricted and with top-process P' such that $P \xrightarrow{a_j} P'$ and $a_1 .. a_{j-1} a_{j+1} .. a_n \in \mathrm{tr}(P')$ and $\forall t \in \mathrm{tr}(P') - \{\lambda\}$ $(CA_r(t) = CA_{r'}(t))$. (**)
With the inductionhypothesis we have $\mathrm{tr}(iw_{r'}(a_1 .. a_{j-1} a_{j+1} .. a_n)) \subseteq CA_{r'}(\mathrm{tr}(P')-\{\lambda\})$. With (**) this gives $\mathrm{tr}(iw_r(a_1 .. a_{j-1} a_{j+1} .. a_n)) \subseteq CA_r(\mathrm{tr}(P')-\{\lambda\}))$. Or, more conform the notation in (*): $\mathrm{tr}((CA_r(a_1) \circ_{iw} .. \circ_{iw} CA_r(a_n))^{(j)}) \subseteq CA_r(\mathrm{tr}(P')-\{\lambda\}))$. It is clear we have such a P' for every i ($1 \le i \le n$) with $E(iw_r(a_i)) \cap E(iw_r(a_1 .. a_{i-1})) = \varnothing$, call these processes respectively P'_i.

Thus (*) $\subseteq \bigcup_{\substack{1 \le i \le n \\ E(iw_r(a_i)) \cap E(iw_r(a_1 .. a_{i-1}))=\varnothing}} \{CA_r(a_i)u \mid u \in CA_r(\mathrm{tr}(P'_i)-\{\lambda\})\}$

$\subseteq CA_r(\mathrm{tr}(P)-\{\lambda\}) = CA_r(\mathrm{tr}(s)-\{\lambda\})$.

■

3.10 Example

Let $r = (s, E, N, \varphi, D)$ be a triple in SENφD with s type 1 restricted. Suppose t = c1a1c2c3c4 $\in \mathrm{tr}(s) - \{\lambda\}$. Then we have $\sim(CA_r(t)) = c(e_1, e_2, c1) \circ_{iw} act(e_2, a1) \circ_{iw} c(e_3, e_2, c2) \circ_{iw} c(e_3, e_3, c3) \circ_{iw} c(e_4, e_3, c4)$. In figure 3 interworking $\sim(CA_r(t))$ is depicted. The actions a1 and c3 are not in $\mathrm{Im}_{E,\varphi}(\gamma)$. This is because $\mathrm{Im}_{E,\varphi}(\gamma)$ contains only the communication actions between different *P*-entities. Suppose we abstract from a1 and c3 in t with $\lambda_{\{a1, c3\}}(t)$. Here, $\lambda_I(\sigma)$ is the word obtained from σ by leaving out all elements from I [BW90, lemma 6.3.8]. The interworking we get from $\lambda_{\{a1, c3\}}(t)$ is $\sim(CA_r(\lambda_{\{a1, c3\}}(t))) = c(e_1, e_2, c1) \circ_{iw} c(e_3, e_2, c2) \circ_{iw} c(e_4, e_3, c4)$. Interworking $\sim(CA_r(\lambda_{\{a1, c3\}}(t)))$ is also depicted in figure 3.

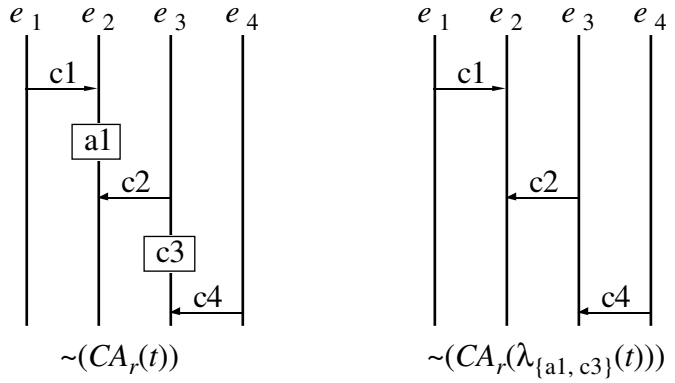


figure 3

In this example one can easily see that interworking $\sim(CA_r(\lambda_{\{a1, c3\}}(t)))$ does not have the ability to interleave c1, c2 and c4 differently than $\sim(CA_r(t))$. If we take as abstraction set {a1, c3, c2}, then $c(e_4, e_3, c4)c(e_1, e_2, c1) \in \mathrm{tr}(iw_r(c1c4))$. However, we need not have $c4c1 \in \mathrm{tr}(\lambda_{\{a1, c3, c2\}}(s)) - \{\lambda\}$!

3.11 Notation

With uppercase symbols in a trace we denote elements t from A^* such that $\lambda_I(t) = \lambda$. For instance, given $r = (s, E, N, \varphi, D)$ in SENφD with s is type 2 restricted and s has top-process $P = \tau_I(\partial_H(p_1 \parallel .. \parallel p_m))$ $(m \geq 1)$. If $A_1a_1A_2a_2 .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$, then we assume $a_1 .. a_n \in tr(\tau_I(\partial_H(p_1 \parallel .. \parallel p_m)))$.

3.12 Proposition

Given $r = (s, E, N, \varphi, D)$ in SENφD with s type 2 restricted and with top-process P.

If $a_1 .. a_n \in tr(P)$, then there exists a $r' = (s', E', N', \varphi', D')$ in SENφD with s' type 2 restricted and with top-process P' such that $P \xrightarrow{a_1} P'$, $a_2 .. a_n \in tr(P')$ and

$\forall t \in tr(P') - \{\lambda\}$ $(CA_r(t) = CA_{r'}(t))$

Proof

Suppose $P = \tau_I(\partial_H(p_1 \parallel .. \parallel p_m))$ $(m \geq 1)$. From the fact that $a_1 .. a_n \in tr(P)$ we have with definition of the trace set for constructible processes [BW90, 6.3.9] that $a_1 .. a_n \in \{\lambda_I(u) \mid u \in tr(\partial_H(p_1 \parallel .. \parallel p_m))\}$. So there must exist a trace t in $tr(\partial_H(p_1 \parallel .. \parallel p_m))$ such that $\lambda_I(t) = a_1 .. a_n$. Suppose $t = A_1a_1A_2a_2 .. A_na_nA_{n+1}$ and $|A_1a_1A_2| = k$ (thus $k \geq 1$). Take the type 1 specification we get from r by replacing P with $\partial_H(p_1 \parallel .. \parallel p_m)$, and apply proposition 3.6 k-times. Suppose this gives the quintet $r'' = (s'', E'', N'', \varphi'', D'')$ in SENφD such that s'' is type 1 restricted and s'' has top-process P'' such that $\partial_H(p_1 \parallel .. \parallel p_m) \xrightarrow{A_1a_1A_2} P''$, $a_2 .. A_{n+1} \in tr(P'')$ and $\forall t \in tr(P'') - \{\lambda\}$ $(CA_r(t) = CA_{r''}(t))$.

Now we take the specification r', that we were looking for, equal to $(s', E'', N'', \varphi'', D'')$ where s' is constructed from s'' by taking $P' = \tau_I(P'')$. One can easily see that

$P \xrightarrow{a_1} P'$, $a_2 .. a_n \in tr(P')$ and $\forall t \in tr(P') - \{\lambda\}$ $(CA_r(t) = CA_{r'}(t))$. (*)

Remains to proof that s' constructed in this way is indeed type 2 restricted. This amounts to showing that the restriction on I is still true, i.e. $I \cap Im_{E,\varphi}(\gamma) = \emptyset$. Suppose we have a communication action c, $c \in I$ and $c = \gamma(a, b)$. For s is type 2 we have $entity_{E,\varphi}(a) = entity_{E,\varphi}(b)$. With (*) we see that also $entity_{E'',\varphi''}(a) = entity_{E'',\varphi''}(b)$; thus $c \notin Im_{E,\varphi}(\gamma)$.

■

3.13 Notation

Given $r = (s, E, N, \varphi, D)$ in SENφD with s is type 2 restricted and s has top-process $P = \tau_I(\partial_H(p_1 \parallel .. \parallel p_m))$ $(m \geq 1)$. Furthermore $A_1a_1A_2a_2 .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$. Take index j $(1 \leq j \leq n)$. With $A'_{i,j}$ we denote the sequence of actions that appear in that order in A_i such that $E(iw_r(a_j)) \cap E(iw_r(A'_{i,j})) \neq \emptyset$ $(1 \leq i \leq n+1)$. With $A''_{i,j}$ we denote the sequence of actions that appear in that order in A_i such that $E(iw_r(a_j)) \cap E(iw_r(A''_{i,j})) = \emptyset$ $(1 \leq i \leq n+1)$. Note that all actions appearing in A_i must either appear in $A'_{i,j}$ or in $A''_{i,j}$.

3.14 Proposition

Given $r = (s, E, N, \varphi, D)$ in SENφD with s type 2 restricted.

If $a_1 .. a_n \in tr(s)$ and $E(iw_r(a_j)) \cap E(iw_r(a_1 .. a_{j-1})) = \varnothing$,

then $a_j .. a_{j-1}a_{j+1} .. a_n \in tr(s)$ $(1 \leq j \leq n)$.

Proof

Suppose s has top-process $P = \tau_I(\partial_H(p_1 \parallel .. \parallel p_m))$ $(m \geq 1)$ and $A_1a_1A_2a_2 .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$ such that $\lambda_I(A_1a_1A_2a_2 .. A_na_nA_{n+1}) = a_1 .. a_n$ (notation 3.13).

We proof:

$A_1A'_{2,j}A'_{3,j} .. A'_{j,j}a_ja_1A''_{2,j}a_2A''_{3,j}a_3 .. A''_{j,j}A_{j+1}a_{j+1} .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$.

If $j = 1$, then $E(iw_r(a_j)) \cap E(iw_r(a_1 .. a_{j-1})) = \varnothing$ by definition and we know already $A_1a_1A_2a_2 .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$. (Note that with the subtrace $a_1 .. a_{j-1}$ we denote a prefix of $a_1 .. a_n \in tr(s)$; thus $\lambda_I(a_1 .. a_{j-1}) = a_1 .. a_{j-1}$.) Suppose $2 \leq j \leq n$. Define $\alpha(t)$ as the set of actions occurring in trace t.

Claim: $\forall a \in \bigcup_{2\leq i\leq j}\alpha(A_i)\ (E(iw_r(a)) \cap E(iw_r(a_j)) \neq \varnothing \Rightarrow E(iw_r(a)) \cap E(iw_r(a_1 .. a_{j-1})) = \varnothing)$

Proof: Suppose we have an a in $\bigcup_{2\leq i\leq j}\alpha(A_i)$ such that $E(iw_r(a)) \cap E(iw_r(a_j)) \neq \varnothing$.

We know $a \in I$. From the fact that $I \cap Im_{E,\varphi}(\gamma) = \varnothing$ we have $E(iw_r(a)) \subseteq E(iw_r(a_j))$.

From the fact that $E(iw_r(a_j)) \cap E(iw_r(a_1 .. a_{j-1})) = \varnothing$ we now have that also

$E(iw_r(a)) \cap E(iw_r(a_1 .. a_{j-1})) = \varnothing$.

Corollary: $E(iw_r(A'_{i,j})) \cap E(iw_r(A''_{k,j})) = \varnothing$ $(2 \leq i, k \leq j)$.

With the corollary we can apply proposition 3.8 repeatedly in order to get:

$A_1a_1A'_{2,j}A''_{2,j}a_2A'_{3,j}A''_{3,j}a_3 .. A'_{j,j}A''_{j,j}a_ja_{j+1}a_{j+1} .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$.

With the claim and again with multiple applications of proposition 3.8 we can see we have the following traces:

$A_1A'_{2,j}a_1A''_{2,j}a_2A'_{3,j}A''_{3,j}a_3 .. A'_{j,j}A''_{j,j}a_ja_{j+1}a_{j+1} .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$,

$A_1A'_{2,j}A'_{3,j}a_1A''_{2,j}a_2A''_{3,j}a_3 .. A'_{j,j}A''_{j,j}a_ja_{j+1}a_{j+1} .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$,

.

.

$A_1A'_{2,j}A'_{3,j} .. A'_{j,j}a_1A''_{2,j}a_2A''_{3,j}a_3 .. A''_{j,j}a_jA_{j+1}a_{j+1} .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$.

And finally we have with proposition 3.8 that

$A_1A'_{2,j}A'_{3,j} .. A'_{j,j}a_ja_1A''_{2,j}a_2A''_{3,j}a_3 .. A''_{j,j}A_{j+1}a_{j+1} .. A_na_nA_{n+1} \in tr(\partial_H(p_1 \parallel .. \parallel p_m))$, and thus $a_j .. a_{j-1}a_{j+1} .. a_n \in tr(s)$.

■

3.15 Proposition

Given $r = (s, E, N, \varphi, D)$ in SENφD with s type 2 restricted, then iw_r satisfies

$\bigcup_{t\in tr(s)-\{\lambda\}} tr(iw_r(t)) \subseteq CA_r(tr(s) - \{\lambda\})$.

Proof

Suppose we have a trace $t \in \mathrm{tr}(s)-\{\lambda\}$. We proof with induction on the length of t that $\mathrm{tr}(iw_r(t)) \subseteq CA_r(\mathrm{tr}(s)-\{\lambda\})$. If $|t| = 1$, then this is obvious. Suppose $t = a_1 .. a_n \in \mathrm{tr}(s)-\{\lambda\}$ with $n \geq 2$. Futhermore suppose s has top-process $P = \tau_I(\partial_H(p_1 \parallel .. \parallel p_m))$ $(m \geq 1)$. From the fact that $a_1 .. a_n \in \mathrm{tr}(P)$ we have with the definition of the trace set for constructible processes [BW90, 6.3.9] that $a_1 .. a_n \in \{\lambda_I(u) \mid u \in \mathrm{tr}(\partial_H(p_1 \parallel .. \parallel p_m))\}$. So there must exist a trace t in $\mathrm{tr}(\partial_H(p_1 \parallel .. \parallel p_m))$ such that $\lambda_I(t) = a_1 .. a_n$. Suppose $t = A_1 a_1 A_2 a_2 .. A_n a_n A_{n+1}$.

$\mathrm{tr}(iw_r(a_1 .. a_n)) = \mathrm{tr}(\sim(CA_r(a_1) .. CA_r(a_n))) = \mathrm{tr}(CA_r(a_1) \circ_{iw} .. \circ_{iw} CA_r(a_n)) = \{$ prop. 3.5$\}$

$$\mathrm{tr}\Big(\sum_{\substack{1\leq i\leq n \\ E(iw_r(a_i))\cap E(iw_r(a_1 .. a_{i-1}))=\emptyset}} CA_r(a_i) \cdot (CA_r(a_1) \circ_{iw} .. \circ_{iw} CA_r(a_n))^{(i)} \Big) = \{\text{def. tr}\}$$

$$\bigcup_{\substack{1\leq i\leq n \\ E(iw_r(a_i))\cap E(iw_r(a_1 .. a_{i-1}))=\emptyset}} \{CA_r(a_i)u \mid u \in \mathrm{tr}((CA_r(a_1) \circ_{iw} .. \circ_{iw} CA_r(a_n))^{(i)})\} \qquad (*)$$

Suppose for index j $(1 \leq j \leq n)$ we have $E(iw_r(a_j)) \cap E(iw_r(a_1 .. a_{j-1})) = \emptyset$ (there is always such an index j: one can always take j = 1). With proposition 3.14 we have $a_j a_1 .. a_{j-1} a_{j+1} .. a_n \in \mathrm{tr}(P)$. With proposition 3.12 follows that there is a quintet $r' = (s', E', N', \varphi', D')$ in SENφD such that s' is type 2 restricted and s' has top-process P' such that

$P \xrightarrow{a_j} P'$ and $a_1 .. a_{j-1} a_{j+1} .. a_n \in \mathrm{tr}(P')$ and $\forall t \in \mathrm{tr}(P') - \{\lambda\}$ $(CA_r(t) = CA_{r'}(t))$. $\qquad (**)$

With the induction hypothesis we have $\mathrm{tr}(iw_{r'}(a_1 .. a_{j-1} a_{j+1} .. a_n)) \subseteq CA_{r'}(\mathrm{tr}(P')-\{\lambda\})$. With (**) this gives $\mathrm{tr}(iw_r(a_1 .. a_{j-1} a_{j+1} .. a_n)) \subseteq CA_r(\mathrm{tr}(P')-\{\lambda\}))$. Or, more conform the notation in (*): $\mathrm{tr}((CA_r(a_1) \circ_{iw} .. \circ_{iw} CA_r(a_n))^{(j)}) \subseteq CA_r(\mathrm{tr}(P')-\{\lambda\}))$. It is clear we have such a P' for every i $(1 \leq i \leq n)$ with $E(iw_r(a_i)) \cap E(iw_r(a_1 .. a_{i-1})) = \emptyset$, call these processes respectively P'_i.

$$\text{Thus } (*) \subseteq \bigcup_{\substack{1\leq i\leq n \\ E(iw_r(a_i))\cap E(iw_r(a_1 .. a_{i-1}))=\emptyset}} \{CA_r(a_i)u \mid u \in CA_r(\mathrm{tr}(P'_i)-\{\lambda\})\}$$

$$\subseteq CA_r(\mathrm{tr}(P)-\{\lambda\}) = CA_r(\mathrm{tr}(s)-\{\lambda\}).$$

■

3.16 Proposition

Let $r = (s, E, N, \varphi, D)$ be a triple in SENφD with s type 2 restricted, then iw_r satisfies

$$CA_r(\mathrm{tr}(s) - \{\lambda\}) \subseteq \bigcup_{t \in \mathrm{tr}(s)-\{\lambda\}} \mathrm{tr}(iw_r(t)).$$

Proof

Given trace $b_1 .. b_n \in CA_r(\mathrm{tr}(s) - \{\lambda\})$. We proof that $b_1 .. b_n \in \mathrm{tr}(\sim(b_1 .. b_n))$ $(n \geq 1)$.

For n = 1 this is evident. Suppose $n \geq 2$. With proposition 3.5 we have

$$\mathrm{tr}(b_1 \circ_{iw} .. \circ_{iw} b_n) = \mathrm{tr}\Big(\sum_{\substack{1\leq i\leq n \\ E(b_i)\cap E(1,i-1)=\emptyset}} b_i \cdot (b_1 \circ_{iw} .. \circ_{iw} b_n)^{(i)} \Big) \supseteq \{b_1 u \mid u \in \mathrm{tr}(b_2 \circ_{iw} .. \circ_{iw} b_n)\}.$$

With the induction hypothesis we have $b_2 .. b_n \in \mathrm{tr}(\sim(b_2 .. b_n))$.

Thus $b_1 .. b_n \in \mathrm{tr}(\sim(b_1 .. b_n))$.■

3.17 Proposition

Let r be a quintet in SENφD with s is type 2 restricted, then iw_r satisfies the IW-interleaving condition. (see definition 3.1.)

Proof

Directly with proposition 3.15 and proposition 3.16.

■

3.18 Corollary

Let r be a quintet in SENφD with s is type 1 restricted, then iw_r satisfies the IW-interleaving condition.

Proof

Directly with proposition 3.17, for the type 1 specifications are contained in the type 2 specifications.

■

Section 4 Conclusions

In this section we will draw some conclusions and make some final remarks on the notions introduced in this paper.

We have the impression that the restrictions posed on a specification in definition 2.4 and 2.7 do not render the idea of interworking generation from traces absolete. Most specifications in PSF are already conform with the restrictions 1 to 4 in definition 2.7 for type 2 restricted specifications. There may be a problem with abstraction at first glance. However in a real situation one will 'play around' with abstraction and P-partitioning to focus on different interesting parts of a trace. I.e. one will probably alter the abstraction set anyway. Such a newly choosen abstraction set has to obey the restriction in definition 2.7. But we think that will not be a real handicap in practice.

Implementation of a tool that generates interworkings from traces is feasable. The pseudo-code in appendix A is hopefully clear. Such a tool could become part of the already existing set of tools developed by Mauw, van Wijk and Winter at Eindhoven University of Technology and Philips Research Laboratories Eindhoven. Figure 2 in appendix B is originally generated by one of these tools.

Now we turn to the idea proposed in the introduction. This was about matching interworkings made before hand (a first specification) and interworkings generated from traces (a PSF specification). In other words, given an interworking and a process, decide if the traces of this interworking are (sub)traces of traces of the process. The complexity of this matching can be determined for regular processes (this holds in the absence of data, in a PSF setting this will be again more complicated). This leads us to the first of three suggestions for

further research: What is the complexitity in the setting of regular processes and what heuristics would be helpfull in a PSF setting?

The generation of interworkings from traces can be seen as a separate process that one can perform if one feels like it. However, we feel that interworkings generated from traces do enhance the comprehension of a PSF specification. We would therefore like to give interworkings a more profound role in the specification process. Especially when simulating a specification. Here the work of Korver [K92] is of interest, he investigates the notion of simulation and simulation tools from a formal point of view. This leads us to the second suggestion for further research: can interworkings form a basic notion in the formal setting of simulations? And subsequently, should interworkings be preferred to traces when using a simulation tool?

In this paper, time is not considered. However there is literature available about process algebras and time [BB92]. This leads us to the third suggestion for further research: how can interworkings be extended with time?

We will finish now with a few brief remarks.

The 'skip<x>' that may occur in a real PSF trace is not fully treated. This is more or less an implementation issue, and the treatment is also dependent on further development of the PSF toolkit. Only in the pseudo-code in appendix A, skips are explicitly removed.

Actions of the form c(p, p, m) are allowed in [MWW92]. In 'handwritten' interworkings these action will probably seldom occur. This is in contrast with generated interworkings. One could argue that actions of the form c(p, p, m) be identified with actions of the form 'act(..)' in one way or another.

We do not pose restrictions on the use of encapsulation. That is, we do not pose restrictions on the argument H of the encapsulation-function ∂. Given restricted specification s and three actions *read*, *send* and *comm* in A, such that γ(*send*, *read*) = *comm* and *send* $\notin$ H $\cup$ I. We could have traces now, with occurences of the action *send*. Suppose t is such a trace. If we translate t to an interworking *send* will be translated to act(e, *send*), for e a P-entity of s. One could argue that Arg(γ) be a subset of H, or *send* be translated to lost(e, f, *send*), where f is the entity that was not able to receive.

If we adhere to the convention of using 'read' and 'send' as names for communicating actions, the direction function D can be generated from the specification. We can also 'code' direction by placing a restriction on the use of encapsulation. Take as encapsulation set the set of 'read' actions R; H = R. Direction can then easily be decided by checking membership in H. Actions from S ocurring in a trace should then be turned into actions of the form lost(..) by the function CA.

Acknowledgements

We wish to thank Jan Bergstra for introducing us at Philips Research Laboratories Eindhoven (PRLE) and for suggesting us to work on the generation of interworkings from traces. At PRLE, Thijs Winter introduced us to the project on interworkings in which he was cooperating with Sjouke Mauw and Mark van Wijk (both from the Eindhoven University of Technology). We owe a lot to Thijs Winter and Sjouke Mauw, they read our work repeatedly and supplied us with useful comments and suggestions. Piet Rodenburg was of great help in structuring the proofs.

References

[MV89] S. Mauw, G.J. Veltink, An introduction to PSFd, in: Proc. International Joint Conference on Theory and Practice of Software Development, TAPSOFT '89, (J. Diaz, F. Orejas, eds.) LNCS 352, pp. 272-285, Springer Verlag, 1989.

[BW90] J.C.M. Baeten, W.P. Weijland, Process Algebra, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[MV90] S. Mauw, G.J. Veltink, A process specification formalism, Fundamenta Informaticae XIII, pp. 85-139, 1990.

[V90] F.W. Vaandrager, Some observations on redundancy in a context; in J.C.M. Baeten ed., Applications of Process Algebra, Cambridge Tracts in Theoretical Computer Science 17, Cambridge University Press, 1990.

[M91] S. Mauw, PSF a process specification formalism, PhD Thesis, University of Amsterdam, 1991.

[V91] G.J. Veltink, The PSF Toolkit, Report P9107, University of Amsterdam, 1991.

[BB92] J.C.M. Baeten, J.A. Bergstra, Discrete Time Process Algebra, Report P9208b, University of Amsterdam, 1992.

[K92] H.Korver, A Theory for Simulator Tools in Process Algebra, Department of Software Technology CWI, 1992. To appear.

[MWW92] S. Mauw, M. van Wijk, T. Winter, Syntax and Semantics of Synchronous Interworkings, Philips Research RWB508RE92436, 1992.

[R92] E. Rudolph, Syntax and Semantics of Basic Sequence Charts, Contribution to the Study Group X, WP X/3, Q8, CCITT meeting, Geneva 1992.

[W92] J.J. van Wamel, A study of a One Bit Sliding Window Protocol in ACP, Report P9212, University of Amsterdam, 1992.