Refinement in Interworkings

S. Mauw and M.A. Reniers

Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, NL–5600 MB Eindhoven, The Netherlands.
sjouke@win.tue.nl , michelr@win.tue.nl

Abstract. Interworkings is a graphical language for displaying the interaction between system components. In this paper we give a formal semantics for Interworkings based on process algebra. A notion of refinement on Interworkings will be defined.

1 Introduction

The Interworking language (IW) is a graphical formalism for displaying the communication behaviour of system components. It was developed in order to support the informal diagrams used at Philips Kommunikations Industrie (Nürnberg) which were used for requirements specification and design.

One of the reasons for developing an explicit language was that it showed very hard to maintain a large collection of diagrams by hand. Several problems were encountered. First of all, manually drawing and updating large diagrams is an expensive activity. Secondly, diagrams that are linked to each other must be updated consistently. Therefore, consistency checks are needed. Thirdly, the relation between the diagrams in a collection is only implicit. Some diagrams describe successive behaviour of one part of the system, other diagrams define the concurrent behaviour of different parts of the system, while still others describe the same behaviour of the same part of the system, but from a different level of abstraction. Thus, diagrams may be a refinement of other diagrams. Finally, there existed different interpretations of the meaning of even simple Interworkings.

In order to solve above mentioned problems, a tool set was developed [10] and a formal semantics was proposed [9]. The semantics are given via a translation into process algebra [4, 3, 2].

The proposed semantics does not consider the notion of refinement between Interworkings. Furthermore, it has some minor shortcomings. The purpose of this paper is to extend and improve upon the semantics treated in [9] such that refinement can be defined. Thereto, we will extend the process algebra used and the bisimulation model. We will prove soundness and completeness of our theory and derive some useful properties.

The Interworking language is a member of a large class of similar graphical notations, most of which are only informally defined, such as Signal Sequence Charts, Use Cases, Information Flow Diagrams, Message Flow and Arrow Diagrams. Interworkings are similar to Message Sequence Charts [5], which are

standardized by the International Telecommunication Union (ITU). The main difference is that Interworkings describe synchronous communication, whereas Message Sequence Charts describe asynchronous communication. The semantics of MSC as described in [7, 6] is also very similar to the semantics of IW.

This paper is organized as follows. In Sect. 2 we give a short introduction to the Interworking language and the Interworking operators. Section 3 contains a formal definition of the Interworking operators and several properties. Complete proofs can be found in [8]. The Interworking refinement is defined in Sect. 4.

Acknowledgements. We would like to thank Thijs Winter and Mark van Wijk for cooperating on a preliminary, although never published, version of Interworkings with refinement. Furthermore, we thank Jos Baeten, Twan Basten, Loe Feijs, Hans Mulder, Jan Gerben Wijnstra and the anonymous referees for their criticism.

2 Interworkings

In this section, we will explain the graphical Interworking language and two ways of composing Interworkings, namely the Interworking-sequencing and the Interworking-merge. Although Interworkings can also be expressed in a textual notation, we will not discuss this.

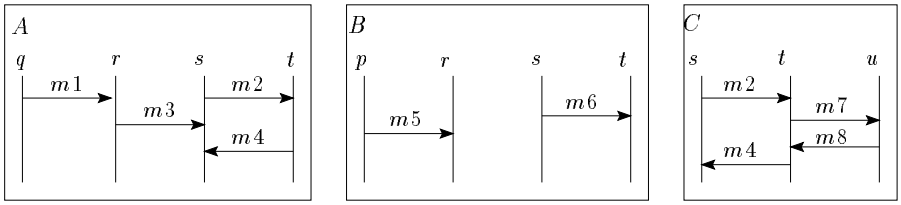

Fig. 1. Interworkings A, B and C

Figure 1 shows a collection of three Interworkings, named A, B and C. Interworking C, for instance, describes the communication between three entities, called s, t and u. Every entity is represented by a vertical axis. Along one axis, time runs from top to bottom, but there is no global time ordering assumed. Messages exchanged between entities are represented by arrows. The interpretation of Interworking C is simple: first s sends message $m2$ to t, then t sends $m7$ to u, next, u sends $m8$ to t, and finally t sends $m4$ to s. Due to the time ordering per entity axis, the messages in Interworking C are totally ordered.

Interworking B shows two unrelated messages $m5$ and $m6$. Although $m6$ is drawn above $m5$, they may occur in any order. In Interworking A messages $m1$

and $m2$ are not related, but they have to occur before $m3$, which in turn occurs before $m4$.

In practice, Interworking diagrams may become very large. Therefore, composition and decomposition techniques are introduced that help to keep the size of an Interworking manageable. For vertical composition of Interworkings we introduce the Interworking-sequencing operator ($\circ_{\mathrm{w}}$). Applying this operator means that the operands are simply concatenated below each other, taking care that common entities are linked in the right way. Interworking D from Fig. 2 is simply the vertical composition of Interworkings A and B ($A \circ_{\mathrm{w}} B$).

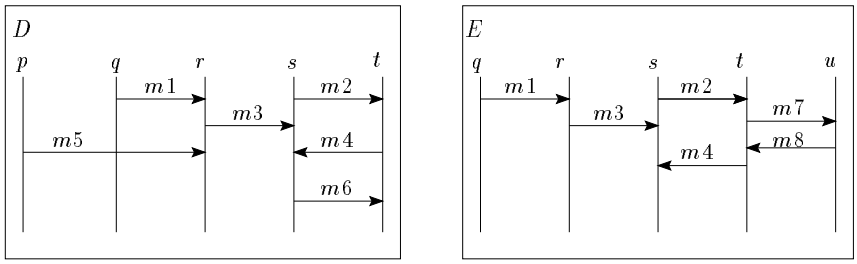

Fig. 2. Interworkings D ($= A \circ_{\mathrm{w}} B$) and E ($= A \parallel_{\mathrm{iw}} C$)

Please notice that every non-empty Interworking is equal to the sequential composition of a number of Interworkings that consist of one message each. So, given the semantics of one message, the Interworking-sequencing operator defines the semantics of a given Interworking diagram.

Next, we will explain the Interworking-merge operator ($\parallel_{\mathrm{iw}}$), which is used for horizontal composition of Interworkings. The easiest case is that the operands have no entities in common. In this case, the Interworkings are simply put next to each other. In the case that the merged Interworkings have entities in common we will interpret this as an overlap between the Interworkings that should not be duplicated. The common entities of the operands are identified. Likewise, the messages between the common instances of the operands are identified.

Interworking E from Fig. 2 is the horizontal composition of the Interworkings A and C ($A \parallel_{\mathrm{iw}} C$). The common entities are s and t. Notice that the communication behaviour between s and t from A is exactly equal to the communication behaviour between s and t from C.

In case that the communication behaviour between the common entities is not equal for both operands, this is considered as an inconsistency. It indicates that in the description of a part of the system assumptions are made about another part of the system, which are not met.

Among other things, the tool set described in [10] can be used to compose Interworkings both horizontally and vertically. It will report all inconsistencies

with respect to horizontal composition. The next section contains a formal treatment of Interworking-sequencing and Interworking-merge operators. The tools also support the refinement of Interworkings. This notion will be explained in Sect. 4.

3 Process Algebra for Interworkings

3.1 Basic Process Algebra with Deadlock and Empty Process

We will give a brief introduction to the process algebra $BPA_{\delta\varepsilon}(A)$ [2, 3]. This process algebra will be our starting point towards the more complex algebras which are introduced in the following sections. The parameter A of the process algebra represents the set of atomic actions. Besides the atomic actions from the set A, the process algebra has the additional constants δ and ε, which represent *deadlock* and the *empty process*, respectively. The process deadlock is incapable of executing any actions and can moreover not terminate successfully. The empty process can also execute no actions, but it terminates successfully. The set of the atomic actions and the deadlock constant is denoted by A_δ.

From these constants more complex processes can be built by using the operators $+$ and $\cdot$. The $+$ is called *alternative composition* and $\cdot$ is called *sequential composition*. The process $x + y$ can execute either process x or process y, but not both. The process $x \cdot y$ starts executing process x, and upon termination thereof starts the execution of process y. These operators are axiomatized by the axioms from Table 1. In these axioms the variables x, y and z denote arbitrary processes. In order to reduce the number of brackets in processes we have the following priorities on operators: $\cdot$ binds stronger than all other operators and $+$ binds weaker than all other operators.

Table 1. Axioms of $BPA_{\delta\varepsilon}(A)$

$x + y$	$= y + x$	$\delta + x$	$= x$
$(x + y) + z$	$= x + (y + z)$	$\delta \cdot x$	$= \delta$
$x + x$	$= x$	$x \cdot \varepsilon$	$= x$
$(x + y) \cdot z$	$= x \cdot z + y \cdot z$	$\varepsilon \cdot x$	$= x$
$(x \cdot y) \cdot z$	$= x \cdot (y \cdot z)$		

To the process algebra $BPA_{\delta\varepsilon}(A)$ we associate a structured operational semantics in the form of the term deduction system $T(BPA_{\delta\varepsilon}(A))$ in Table 2. For the deduction rules in this table we require that $a \in A$ and that x, y, and z are arbitrary processes. A deduction rule is of the form $\frac{H}{C}$ where H is a set of *hypotheses* and C is the *conclusion*. The formula $x \xrightarrow{a} x'$ expresses that the process x can perform an action a and thereby evolves into the process x'. The

Table 2. Structured operational semantics of $BPA_{\delta\varepsilon}(A)$ ($a \in A$)

$\dfrac{}{\varepsilon \downarrow}$	$\dfrac{x \downarrow}{x + y \downarrow}$	$\dfrac{y \downarrow}{x + y \downarrow}$	$\dfrac{x \downarrow, y \downarrow}{x \cdot y \downarrow}$	
$\dfrac{}{a \xrightarrow{a} \varepsilon}$	$\dfrac{x \xrightarrow{a} x'}{x + y \xrightarrow{a} x'}$	$\dfrac{y \xrightarrow{a} y'}{x + y \xrightarrow{a} y'}$	$\dfrac{x \xrightarrow{a} x'}{x \cdot y \xrightarrow{a} x' \cdot y}$	$\dfrac{x \downarrow, y \xrightarrow{a} y'}{x \cdot y \xrightarrow{a} y'}$

formula $x \downarrow$ expresses that process x has an option to terminate immediately and successfully. For a formal definition of term deduction systems we refer to [2].

Definition 1 (Bisimulation). A symmetric relation R on closed $BPA_{\delta\varepsilon}(A)$ terms is a *bisimulation relation*, if and only if, for every pair $(p, q) \in R$ and $a \in A$, the following conditions hold:

1. if $p \xrightarrow{a} p'$, then there exists a closed $BPA_{\delta\varepsilon}(A)$ term q' such that $q \xrightarrow{a} q'$ and $(p', q') \in R$,
2. if $p \downarrow$, then $q \downarrow$.

The closed $BPA_{\delta\varepsilon}(A)$ terms x and y are *bisimilar*, notation $x \leftrightarroweq y$, if and only if there exists a bisimulation relation R relating them.

Finally, we would like to mention the following well-known result from literature, e.g. [3]. The process algebra $BPA_{\delta\varepsilon}(A)$ is a sound and complete axiomatization of bisimulation equivalence on closed $BPA_{\delta\varepsilon}(A)$ terms. This result will be used in the following sections when relating the extended process algebras to $BPA_{\delta\varepsilon}(A)$.

3.2 Interworking-sequencing and Interworking-merge

In this section, we will extend the process algebra $BPA_{\delta\varepsilon}(A)$ from the previous section. First, we will instantiate the parameter A of $BPA_{\delta\varepsilon}(A)$ by the actual atomic actions occurring in Interworkings. Next, we define the Interworking-sequencing operator $\fatsemi_{\mathrm{iw}}$, and the E-Interworking merge operator $\|^E_{\mathrm{iw}}$. We will extend the process descriptions with sets of entities and call them entity-labeled processes. Finally, we define the Interworking-merge on these extended processes.

The Entity Function and the Interworking-sequencing. In an Interworking diagram there are two types of objects: entities and messages. Entities come from a set *EID* and messages from a set *MID*. We use $c(i, j, m)$ to denote the sending of message m from entity i to entity j. Thus, parameter A from $BPA_{\delta\varepsilon}(A)$ will be instantiated by $\{c(i, j, m) \mid i, j \in EID, m \in MID\}$.

We want to determine for each process description which entities are actively involved in it. Thereto, we define the entity function E (see Table 3).

Table 3. Active entities of an Interworking ($i, j \in EID$, $m \in MID$, $a \in A$)

$E(\varepsilon)$	$= \emptyset$	$E(a \cdot x)$	$= E(a) \cup E(x)$
$E(\delta)$	$= \emptyset$	$E(x + y)$	$= E(x) \cup E(y)$
$E(c(i,j,m))$	$= \{i, j\}$		

The Interworking-sequencing of two processes x and y ($x \between_{\mathrm{iw}} y$) is their parallel execution with the restriction that the right-hand process may execute an action only if the entities of that action are disjoint from the entities of the left-hand process. The Interworking-sequencing operator is similar to the *weak sequential composition* operator from [11].

The axiomatization of $\between_{\mathrm{iw}}$ is basically the one presented in [9], but extended with axioms for the empty process (see Table 4). We use the two auxiliary operators $\mathbf{L}\between_{\mathrm{iw}}$ and $\mathbf{R}\between_{\mathrm{iw}}$. The process $x \,\mathbf{L}\between_{\mathrm{iw}}\, y$ behaves like the process $x \between_{\mathrm{iw}} y$ with the restriction that the first action to be executed must originate from process x. The process $x \,\mathbf{R}\between_{\mathrm{iw}}\, y$ also behaves like the process $x \between_{\mathrm{iw}} y$ but this time with the restriction that the first action to be executed must be from process y. In this case, the first action from y can only be executed if it is not blocked by any of the actions from x. The operator $\surd$ is used to obtain a proper treatment of the empty process (i.e., we want $\varepsilon \between_{\mathrm{iw}} \varepsilon = \varepsilon$).

Table 4. Axioms of Interworking-sequencing ($a \in A_\delta$)

$x \between_{\mathrm{iw}} y$	$= x \,\mathbf{L}\between_{\mathrm{iw}}\, y + x \,\mathbf{R}\between_{\mathrm{iw}}\, y + \surd(x) \cdot \surd(y)$	
$\varepsilon \,\mathbf{L}\between_{\mathrm{iw}}\, x$	$= \delta$	
$a \cdot x \,\mathbf{L}\between_{\mathrm{iw}}\, y$	$= a \cdot (x \between_{\mathrm{iw}} y)$	
$(x + y) \,\mathbf{L}\between_{\mathrm{iw}}\, z$	$= x \,\mathbf{L}\between_{\mathrm{iw}}\, z + y \,\mathbf{L}\between_{\mathrm{iw}}\, z$	
$x \,\mathbf{R}\between_{\mathrm{iw}}\, \varepsilon$	$= \delta$	
$x \,\mathbf{R}\between_{\mathrm{iw}}\, a \cdot y$	$= a \cdot (x \between_{\mathrm{iw}} y)$	if $E(a) \cap E(x) = \emptyset$
$x \,\mathbf{R}\between_{\mathrm{iw}}\, a \cdot y$	$= \delta$	if $E(a) \cap E(x) \neq \emptyset$
$x \,\mathbf{R}\between_{\mathrm{iw}}\, (y + z)$	$= x \,\mathbf{R}\between_{\mathrm{iw}}\, y + x \,\mathbf{R}\between_{\mathrm{iw}}\, z$	
$\surd(\varepsilon)$	$= \varepsilon$	
$\surd(a \cdot x)$	$= \delta$	
$\surd(x + y)$	$= \surd(x) + \surd(y)$	

The structured operational semantics of the Interworking-sequencing and the auxiliary operators is given in Table 5.

The *E*-Interworking Merge. The axiomatization of the S-Interworking merge as presented in [9] uses the auxiliary operators left S-Interworking merge $\lfloor\!\lfloor^S_{\mathrm{iw}}$ and synchronization Interworking-merge $|^S_{\mathrm{iw}}$ with S a set of atomic actions. We

Table 5. Structured operational semantics of Interworking-sequencing ($a \in A$)

$$\frac{x\downarrow, y\downarrow}{x \triangleleft_{\mathrm{iw}} y\downarrow} \qquad \frac{x \xrightarrow{a} x'}{x \triangleleft_{\mathrm{iw}} y \xrightarrow{a} x' \triangleleft_{\mathrm{iw}} y} \qquad \frac{y \xrightarrow{a} y', E(a)\cap E(x) = \emptyset}{x \triangleleft_{\mathrm{iw}} y \xrightarrow{a} x \triangleleft_{\mathrm{iw}} y'}$$

$$\frac{x\downarrow}{\surd(x)\downarrow} \qquad \frac{x \xrightarrow{a} x'}{x \,\mathbf{L}\!\triangleleft_{\mathrm{iw}}\, y \xrightarrow{a} x' \triangleleft_{\mathrm{iw}} y} \qquad \frac{y \xrightarrow{a} y', E(a)\cap E(x) = \emptyset}{x \,\mathbf{R}\!\triangleleft_{\mathrm{iw}}\, y \xrightarrow{a} x \triangleleft_{\mathrm{iw}} y'}$$

will use similar auxiliary operators only now labeled with a set of entities instead of a set of atomic actions. This set represents the entities on which communication actions must synchronize. The process $x \|^E_{\mathrm{iw}} y$ is the parallel execution of the processes x and y with the restriction that the processes must synchronize on all atomic actions which are defined on entities from the set E. The process $x \lfloor\!\lfloor^E_{\mathrm{iw}} y$ behaves like the process $x \|^E_{\mathrm{iw}} y$ with the restriction that the first action must come from process x and that action cannot synchronize with an action from y. The process $x \mid^E_{\mathrm{iw}} y$ behaves as the process $x \|^E_{\mathrm{iw}} y$ with the restriction that the first action to be executed must be a synchronization. Again we will use the termination operator $\surd$ to make sure that the E-Interworking merge behaves correctly for empty processes (Interworkings), i.e., $\varepsilon \|^E_{\mathrm{iw}} \varepsilon = \varepsilon$ for all $E \subseteq EID$. The definition of the E-Interworking merge operator is given in Table 6. Recall that the axioms for the termination operator are given in Table 4.

Table 6. Axioms of E-Interworking merge ($a, b \in A_\delta$)

$x \|^E_{\mathrm{iw}} y$	$= x \lfloor\!\lfloor^E_{\mathrm{iw}} y + y \lfloor\!\lfloor^E_{\mathrm{iw}} x + x \mid^E_{\mathrm{iw}} y + \surd(x) \cdot \surd(y)$	
$\varepsilon \lfloor\!\lfloor^E_{\mathrm{iw}} x$	$= \delta$	
$a \cdot x \lfloor\!\lfloor^E_{\mathrm{iw}} y$	$= a \cdot (x \|^E_{\mathrm{iw}} y)$	if $E(a) \not\subseteq E$
$a \cdot x \lfloor\!\lfloor^E_{\mathrm{iw}} y$	$= \delta$	if $E(a) \subseteq E$
$(x + y) \lfloor\!\lfloor^E_{\mathrm{iw}} z$	$= x \lfloor\!\lfloor^E_{\mathrm{iw}} z \; + \; y \lfloor\!\lfloor^E_{\mathrm{iw}} z$	
$\varepsilon \mid^E_{\mathrm{iw}} x$	$= \delta$	
$x \mid^E_{\mathrm{iw}} \varepsilon$	$= \delta$	
$a \cdot x \mid^E_{\mathrm{iw}} b \cdot y$	$= a \cdot (x \|^E_{\mathrm{iw}} y)$	if $a \equiv b \wedge E(a) \subseteq E$
$a \cdot x \mid^E_{\mathrm{iw}} b \cdot y$	$= \delta$	if $a \not\equiv b \vee E(a) \not\subseteq E$
$(x + y) \mid^E_{\mathrm{iw}} z$	$= x \mid^E_{\mathrm{iw}} z + y \mid^E_{\mathrm{iw}} z$	
$x \mid^E_{\mathrm{iw}} (y + z)$	$= x \mid^E_{\mathrm{iw}} y + x \mid^E_{\mathrm{iw}} z$	

Table 7 presents the structured operational semantics of the E-Interworking merge and the auxiliary operators introduced. The process algebra consisting of all operators and axioms introduced so far is called IW_ε. The term deduction

system $T(IW_\varepsilon)$ consists of the deduction rules of Tables 2, 5 and 7.

Table 7. Structured operational semantics of E-Interworking merge ($a \in A$)

$$\frac{x\downarrow, y\downarrow}{x \parallel^E_{\mathrm{iw}} y\downarrow} \qquad \frac{x \xrightarrow{a} x', y \xrightarrow{a} y', E(a) \subseteq E}{x \parallel^E_{\mathrm{iw}} y \xrightarrow{a} x' \parallel^E_{\mathrm{iw}} y'} \qquad \frac{x \xrightarrow{a} x', E(a) \not\subseteq E}{x \parallel^E_{\mathrm{iw}} y \xrightarrow{a} x' \parallel^E_{\mathrm{iw}} y}$$

$$\frac{y \xrightarrow{a} y', E(a) \not\subseteq E}{x \parallel^E_{\mathrm{iw}} y \xrightarrow{a} x \parallel^E_{\mathrm{iw}} y'} \qquad \frac{x \xrightarrow{a} x', E(a) \not\subseteq E}{x \between^E_{\mathrm{iw}} y \xrightarrow{a} x' \parallel^E_{\mathrm{iw}} y} \qquad \frac{x \xrightarrow{a} x', y \xrightarrow{a} y', E(a) \subseteq E}{x \mid^E_{\mathrm{iw}} y \xrightarrow{a} x' \parallel^E_{\mathrm{iw}} y'}$$

It turns out that bisimulation equivalence is a congruence for the function symbols in the signature of IW_ε. Furthermore, IW_ε is a sound and complete axiomatization of bisimulation equivalence on closed IW_ε terms. In [8] these results are proven in more detail. These proofs are based on the meta-theory presented in [2, 12].

The Interworking-merge. Now that we have given the axioms and structured operational semantics of the E-Interworking merge we will define the Interworking-merge operator. The Interworking-merge of two processes is their parallel execution with the restriction that the processes must synchronize on all atomic actions which are defined on the common entities of the processes. For the Interworking-merge operator it is necessary to determine the common entities of the operands. The entities of an operand cannot be obtained from the process term representing it (as was done in [9]), since empty entities are not represented in the process term. Therefore, we label every process term by a set of entity names over EID. For an Interworking x, this set represents the entities of the Interworking (including the empty entities). An Interworking with a dynamical behaviour denoted by x over the entities from E is denoted by $\langle x, E\rangle$. Such a tuple $\langle x, E\rangle$ will be called an entity-labeled process.

On entity-labeled processes we define the operators Interworking-sequencing and Interworking-merge. The set of all entity-labeled processes is called LP. The definition of the Interworking-sequencing on entity-labeled processes is straightforward. As was done in [9] the Interworking-merge is expressed in terms of the E-Interworking merge operator and the common entities of the operands. Technically speaking, we can axiomatize the Interworking-merge without using the E-Interworking merge. But, to stay as close as possible to the existing axiomatization of the Interworking-merge, we use the E-Interworking merge. The axioms for entity-labeled processes are given in Table 8 for $E, F \subseteq EID$. The extension of IW_ε with entity-labeled processes is denoted by IWE_ε.

Next, we define a structured operational semantics of entity-labeled processes. In order to make a clear distinction between the transition relation and termination predicate on non-labeled processes and on entity-labeled processes,

Table 8. Axioms of entity-labeled processes

$\langle x, E\rangle \circ_{\mathrm{iw}} \langle y, F\rangle = \langle x \circ_{\mathrm{iw}} y, E \cup F\rangle$	$\langle x, E\rangle \,\|\|_{\mathrm{iw}}\, \langle y, F\rangle = \langle x \,\|\|_{\mathrm{iw}}^{E \cap F}\, y, E \cup F\rangle$

we denote the latter by $\stackrel{a}{\Rightarrow}$ and $\Downarrow$, respectively. The structured operational semantics of entity-labeled processes is related directly to the structured operational semantics of non-labeled processes as expressed in Table 9 (s, t represent entity-labeled processes). Thereto, two auxiliary functions π_{p} and π_{e} are introduced for entity-labeled processes. Intuitively, $\pi_{\mathrm{p}}(s)$ denotes the process-part of s, and $\pi_{\mathrm{e}}(s)$ denotes the entity-part of s.

Table 9. Structured operational semantics of entity-labeled processes ($a \in A$)

$\pi_{\mathrm{p}}(\langle x, E\rangle) = x$	$\pi_{\mathrm{e}}(\langle x, E\rangle) = E$
$\pi_{\mathrm{p}}(s \circ_{\mathrm{iw}} t) = \pi_{\mathrm{p}}(s) \circ_{\mathrm{iw}} \pi_{\mathrm{p}}(t)$	$\pi_{\mathrm{e}}(s \circ_{\mathrm{iw}} t) = \pi_{\mathrm{e}}(s) \cup \pi_{\mathrm{e}}(t)$
$\pi_{\mathrm{p}}(s \,\|\|_{\mathrm{iw}}\, t) = \pi_{\mathrm{p}}(s) \,\|\|_{\mathrm{iw}}^{\pi_{\mathrm{e}}(s) \cap \pi_{\mathrm{e}}(t)}\, \pi_{\mathrm{p}}(t)$	$\pi_{\mathrm{e}}(s \,\|\|_{\mathrm{iw}}\, t) = \pi_{\mathrm{e}}(s) \cup \pi_{\mathrm{e}}(t)$

$$\frac{\pi_{\mathrm{p}}(s) \stackrel{a}{\rightarrow} y}{s \stackrel{a}{\Rightarrow} \langle y, \pi_{\mathrm{e}}(s)\rangle} \qquad \frac{\pi_{\mathrm{p}}(s) \downarrow}{s \Downarrow}$$

Definition 2 (Entity bisimulation). A symmetric relation R on closed LP terms is an *entity bisimulation relation*, if and only if, for every pair $(s, t) \in R$ and $a \in A$, the following conditions hold:

1. if $s \stackrel{a}{\Rightarrow} s'$, then there is a closed LP term t' such that $t \stackrel{a}{\Rightarrow} t'$ and $(s', t') \in R$,
2. if $s \Downarrow$, then $t \Downarrow$,
3. $\pi_{\mathrm{e}}(s) = \pi_{\mathrm{e}}(t)$.

The closed LP terms s and t are *entity bisimilar*, notation $s \underline{\leftrightsquigarrow} t$, if and only if there exists an entity bisimulation relation R relating them.

It is also possible to define entity bisimulation in terms of bisimulation of the process-parts and set equality of the entity-parts.

Lemma 3. *For closed LP terms s and t we have*

$$s \underline{\leftrightsquigarrow} t \quad \text{iff} \quad \pi_{\mathrm{p}}(s) \underline{\leftrightarrow} \pi_{\mathrm{p}}(t) \text{ and } \pi_{\mathrm{e}}(s) = \pi_{\mathrm{e}}(t) \tag{1}$$

Proof. Suppose that $s \underline{\leftrightsquigarrow} t$. Then there exists an entity bisimulation relation R on closed LP terms that relates s and t. Then the relation $R' = \{(\pi_{\mathrm{p}}(p), \pi_{\mathrm{p}}(q)) \mid (p, q) \in R\}$ is a bisimulation relating $\pi_{\mathrm{p}}(s)$ and $\pi_{\mathrm{p}}(t)$. From the definition of entity bisimulation we also obtain $\pi_{\mathrm{e}}(s) = \pi_{\mathrm{e}}(t)$.

Next, suppose that $\pi_{\rm p}(s) \leftrightarrows \pi_{\rm p}(t)$ and $\pi_{\rm e}(s) = \pi_{\rm e}(t)$. Then there exists a bisimulation relation R that relates $\pi_{\rm p}(s)$ and $\pi_{\rm p}(t)$. Then the relation $R' = \{(\langle p, E\rangle, \langle q, E\rangle) \mid (p,q) \in R, E \subseteq EID\}$ is an entity bisimulation relating s and t. Hence, $s \leftrightsquigarrow t$. □

Theorem 4 (Congruence). *Entity bisimulation equivalence is a congruence for the function symbols in the signature of IWE_ε which are defined on LP terms.*

Proof. Suppose $s_1 \leftrightsquigarrow s_2$ and $t_1 \leftrightsquigarrow t_2$. By Lemma 3 we have (1) $\pi_{\rm p}(s_1) \leftrightarrows \pi_{\rm p}(s_2)$, (2) $\pi_{\rm e}(s_1) = \pi_{\rm e}(s_2)$, (3) $\pi_{\rm p}(t_1) \leftrightarrows \pi_{\rm p}(t_2)$, and (4) $\pi_{\rm e}(t_1) = \pi_{\rm e}(t_2)$.

From (1) and (3) and the fact that $\leftrightarrows$ is a congruence for $\triangleleft_{\rm w}$ on closed IW_ε terms, it follows that $\pi_{\rm p}(s_1 \triangleleft_{\rm w} t_1) = \pi_{\rm p}(s_1) \triangleleft_{\rm w} \pi_{\rm p}(t_1) \leftrightarrows \pi_{\rm p}(s_2) \triangleleft_{\rm w} \pi_{\rm p}(t_2) = \pi_{\rm p}(s_2 \triangleleft_{\rm w} t_2)$. From (2) and (4) we obtain $\pi_{\rm e}(s_1 \triangleleft_{\rm w} t_1) = \pi_{\rm e}(s_1) \cup \pi_{\rm e}(t_1) = \pi_{\rm e}(s_2) \cup \pi_{\rm e}(t_2) = \pi_{\rm e}(s_2 \triangleleft_{\rm w} t_2)$. Hence, by Lemma 3, $s_1 \triangleleft_{\rm w} t_1 \leftrightsquigarrow s_2 \triangleleft_{\rm w} t_2$.

From (2) and (4) we have that $\pi_{\rm e}(s_1) \cap \pi_{\rm e}(t_1) = \pi_{\rm e}(s_2) \cap \pi_{\rm e}(t_2)$. From (1) and (3) and the fact that $\leftrightarrows$ is a congruence for $\|^E_{\rm iw}$ on closed IW_ε terms, it follows that $\pi_{\rm p}(s_1 \|_{\rm iw} t_1) = \pi_{\rm p}(s_1) \|^{\pi_{\rm e}(s_1) \cap \pi_{\rm e}(t_1)}_{\rm iw} \pi_{\rm p}(t_1) \leftrightarrows \pi_{\rm p}(s_2) \|^{\pi_{\rm e}(s_2) \cap \pi_{\rm e}(t_2)}_{\rm iw} \pi_{\rm p}(t_2) = \pi_{\rm p}(s_2 \|_{\rm iw} t_2)$. From (2) and (4) we obtain that $\pi_{\rm e}(s_1 \|_{\rm iw} t_1) = \pi_{\rm e}(s_1) \cup \pi_{\rm e}(t_1) = \pi_{\rm e}(s_2) \cup \pi_{\rm e}(t_2) = \pi_{\rm e}(s_2 \|_{\rm iw} t_2)$. Hence, by Lemma 3, $s_1 \|_{\rm iw} t_1 \leftrightsquigarrow s_2 \|_{\rm iw} t_2$. □

Theorem 5 (Soundness). *The process algebra IWE_ε is a sound axiomatization of bisimulation equivalence on closed IW_ε terms. The process algebra IWE_ε is a sound axiomatization of entity bisimulation on closed LP terms.*

Proof. For the first proposition observe that we did not add any axioms relating closed IW_ε terms. We will prove the second proposition. Since entity bisimulation is a congruence for the closed terms of LP (Theorem 4) we only have to show that the axioms from Table 8 are sound. Thereto, we provide a bisimulation relation for each axiom. For both axioms relate the left-hand side to the right-hand side and additionally relate each term to itself. □

Theorem 6 (Conservativity). *The process algebra IWE_ε is a conservative extension of the process algebra IW_ε.*

Proof. The proof of this theorem uses the approach of [12]. The conservativity follows from the following observations:

1. bisimulation is definable in terms of predicate and relation symbols only,
2. IW_ε is a complete axiomatization of bisimulation on closed IW_ε terms,
3. IWE_ε is a sound axiomatization of bisimulation on closed IW_ε terms (see Theorem 5),
4. $T(IW_\varepsilon)$ is pure, well-founded and in path format, and
5. $T(IWE_\varepsilon)$ is in path format. □

Definition 7 (Basic terms). *Basic LP terms* are defined inductively by:

1. if x is a closed IW_ε term and $E \subseteq EID$, then $\langle x, E\rangle$ is a basic LP term
2. no other closed LP terms are basic LP terms

Theorem 8 (Elimination). *For every closed LP term s there exists a basic LP term t such that $IWE_\varepsilon \vdash s = t$.*

Proof. This theorem is proven with induction on the structure of a closed LP term. First, consider the case $s \equiv \langle x, E\rangle$ (x a closed IW_ε term and $E \subseteq EID$). Then s is a basic LP term. Next, consider the case $s \equiv s_1 \between_{\mathrm{w}} s_2$ (s_1, s_2 closed LP terms). Then we have by induction that there exist basic LP terms t_1, t_2 such that $s_1 = t_1$ and $s_2 = t_2$. From the definition of basic LP terms we then also have the existence of closed IW_ε terms x_1, x_2 and $E_1, E_2 \subseteq EID$ such that $t_1 \equiv \langle x_1, E_1\rangle$ and $t_2 \equiv \langle x_2, E_2\rangle$. Then we derive $s \equiv s_1 \between_{\mathrm{w}} s_2 = t_1 \between_{\mathrm{w}} t_2 \equiv \langle x_1, E_1\rangle \between_{\mathrm{w}} \langle x_2, E_2\rangle = \langle x_1 \between_{\mathrm{w}} x_2, E_1 \cup E_2\rangle$ which is a basic LP term. Finally, consider the case $s \equiv s_1 \parallel_{\mathrm{iw}} s_2$ (s_2, s_2 closed LP terms). Again by induction and the definition of basic LP terms, we have the existence of closed IW_ε terms x_1, x_2 and $E_1, E_2 \subseteq EID$ such that $s_1 = \langle x_1, E_1\rangle$ and $s_2 = \langle x_2, E_2\rangle$. Then we derive $s \equiv s_1 \parallel_{\mathrm{iw}} s_2 = \langle x_1, E_1\rangle \parallel_{\mathrm{iw}} \langle x_2, E_2\rangle = \langle x_1 \parallel_{\mathrm{iw}}^{E_1 \cap E_2} x_2, E_1 \cup E_2\rangle$ which is a basic LP term. □

Theorem 9 (Completeness). *The process algebra IWE_ε is a complete axiomatization of entity bisimulation on closed LP terms.*

Proof. By the elimination theorem (Theorem 8) we only have to prove this theorem for basic LP terms. Let $\langle x, E_1\rangle$ and $\langle y, E_2\rangle$ be basic LP terms such that $\langle x, E_1\rangle \leftrightsquigarrow \langle y, E_2\rangle$. By Lemma 3 we have $x \leftrightarrows y$ and $E_1 = E_2$. Since IW_ε is a complete axiomatization of bisimulation equivalence on closed IW_ε terms, we have $x = y$, and hence $\langle x, E_1\rangle = \langle y, E_2\rangle$. □

The proofs of the following properties can be found in [8]. The Interworking-sequencing is commutative under the assumption that the active entities of the operands are disjoint. Furthermore, it is associative. The Interworking-merge is both commutative and associative. The Interworking-merge as defined in [9] did not have the associativity property. This difference is a direct consequence of our decision to maintain the entities of an Interworking statically.

Proposition 10 (Commutativity of $\between_{\mathrm{w}}$ and $\parallel_{\mathrm{iw}}$). *For closed IW_ε terms x, y, closed LP terms s, t and a set of entities E we have*

$$x \between_{\mathrm{w}} y = y \between_{\mathrm{w}} x \quad \text{if } E(x) \cap E(y) = \emptyset \tag{2}$$

$$s \between_{\mathrm{w}} t = t \between_{\mathrm{w}} s \quad \text{if } E(\pi_{\mathrm{p}}(s)) \cap E(\pi_{\mathrm{p}}(t)) = \emptyset \tag{3}$$

$$x \parallel_{\mathrm{iw}}^{E} y = y \parallel_{\mathrm{iw}}^{E} x \tag{4}$$

$$s \parallel_{\mathrm{iw}} t = t \parallel_{\mathrm{iw}} s \tag{5}$$

Proposition 11 (Associativity of $\between_{\mathrm{w}}$ and $\parallel_{\mathrm{iw}}$). *For closed IW_ε terms x, y, z, closed LP terms s, t, u and sets of entities E_1, E_2, E_3 we have*

$$(x \between_{\mathrm{w}} y) \between_{\mathrm{w}} z = x \between_{\mathrm{w}} (y \between_{\mathrm{w}} z) \tag{6}$$

$$(s \between_{\mathrm{w}} t) \between_{\mathrm{w}} u = s \between_{\mathrm{w}} (t \between_{\mathrm{w}} u) \tag{7}$$

$$(x \parallel_{\mathrm{iw}}^{E_1 \cap E_2} y) \parallel_{\mathrm{iw}}^{(E_1 \cup E_2) \cap E_3} z = x \parallel_{\mathrm{iw}}^{E_1 \cap (E_2 \cup E_3)} (y \parallel_{\mathrm{iw}}^{E_2 \cap E_3} z) \tag{8}$$

$$(s \parallel_{\mathrm{iw}} t) \parallel_{\mathrm{iw}} u = s \parallel_{\mathrm{iw}} (t \parallel_{\mathrm{iw}} u) \tag{9}$$

4 Algebraic Definition of Interworking Refinement

Interworking refinement is the replacement of one entity by a number of entities such that the behaviour of the refining Interworking is identical, in a sense to be made precise shortly, to the original Interworking. It is used for enabling a top-down design strategy. In this way, a system can be viewed from the right level of abstraction. Figure 3 shows an example of such a refinement. Interworking A is a refinement of Interworking B, because A contains entities $q1$ and $q2$ that refine the bahaviour of entity q from B.

The relation between the entities in both Interworkings is given by a partial mapping $f : \mathit{EID} \hookrightarrow \mathit{EID}$ from entities to entities. An entity e from $\mathit{rng}(f)$ is refined by the set of all entities e' satisfying $f(e') = e$. If Interworking s is an f-refinement of Interworking t, i.e., s refines t with respect to the entity mapping f, we denote this by $s \sqsubseteq_f t$. The mapping f is partial in order to distinguish between an entity p which is not refined at all ($p \notin \mathit{rng}(f)$) and an entity p which is refined by (amongst others) an entity p ($f(p) = p$).

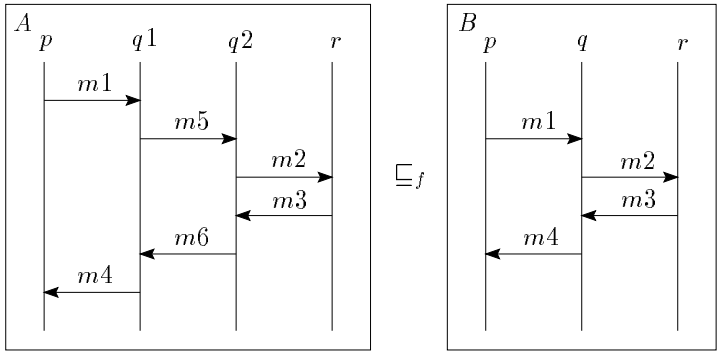

Fig. 3. Interworking refinement ($f(q1) = f(q2) = q$)

The intuition is that the external behaviour of a single entity within the Interworking t can be refined into, or implemented by, the collective behaviour of a number of entities within the Interworking s. Besides the singular refinement discussed above, it is also allowed to consider a number of refinements at the same time. An example of such a multiple refinement is given in Fig. 4. The entity p is refined by the entities $p1$ and $p2$, and the entity q is refined by the entities $q1$ and $q2$.

In the following, we will give the formal definitions involved with Interworking refinement. The operational view basic to the definition of $A \sqsubseteq_f B$ is as follows (see Fig. 3). First, we rename all refining entities of A (i.e. $q1$ and $q2$) that occur in messages of A into the refined entity q. For this purpose, we will define the

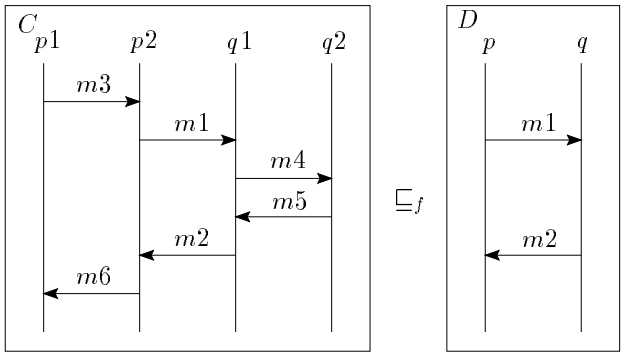

Fig. 4. Multiple refinement ($f(p1) = f(p2) = p$, $f(q1) = f(q2) = q$)

renaming function ρ_f . After this step, all messages between entities $q1$ and $q2$ will become message that are internal to entity q. Now, we remove all these internal messages from the renamed Interworking A and from Interworking B. This is done by applying the operator ε_I that renames atomic actions into the empty process ε. We say that A refines B if the resulting Interworkings are equal,

First, we define the renaming function ρ_f . This operator renames all occurrences of $e \in EID$ into $f(e)$. For the axiomatization of this operator it is easier to have a total function instead of a partial one. Thereto, we extend the partial function f to the total function f^* by asserting that $f^*(x) = x$ for all x for which f is not defined.

Let $f : EID \hookrightarrow EID$ be a partial function, then the renaming operator ρ_f related to f is defined by the axioms in Table 10. This renaming operator resembles the renaming operator ρ_f from [1]. Note that also the entity component of an entity-labeled process is renamed with respect to the mapping f.

Table 10. Entity renaming function on processes ($a \in A$, $i, j \in EID$, $m \in MID$)

$\rho_f(\varepsilon)$	$= \varepsilon$	$\rho_f(a \cdot x)$	$= \rho_f(a) \cdot \rho_f(x)$
$\rho_f(\delta)$	$= \delta$	$\rho_f(x + y)$	$= \rho_f(x) + \rho_f(y)$
$\rho_f(c(i,j,m))$	$= c(f^*(i), f^*(j), m)$	$\rho_f(\langle x, E\rangle)$	$= \langle \rho_f(x), \{f^*(e) \mid e \in E\}\rangle$

Let E be a set of entities. Then the set of all internal actions with respect to the entities from E, notation $Int(E)$, is defined as follows:

$$Int(E) = \{c(i, i, m) \mid i \in EID, m \in MID\} \tag{10}$$

Let I be a set of atomic actions, then we can define the operator ε_I (see [13]) that renames atomic actions from I into ε as in Table 11.

Table 11. Renaming atomic actions into ε ($I \subseteq A$, $a \in A_\delta$)

$\varepsilon_I(\varepsilon) = \varepsilon$		$\varepsilon_I(x+y) = \varepsilon_I(x) + \varepsilon_I(y)$	
$\varepsilon_I(a \cdot x) = \varepsilon_I(x)$	if $a \in I$	$\varepsilon_I(\langle x, E\rangle) = \langle \varepsilon_I(x), E\rangle$	
$\varepsilon_I(a \cdot x) = a \cdot \varepsilon_I(x)$	if $a \notin I$		

Let $f : EID \hookrightarrow EID$ be a refinement mapping, the f-refinement relation on entity-labeled processes is then defined as follows:

$$s \sqsubseteq_f t \quad \text{iff} \quad \varepsilon_{Int(rng(f))}(\rho_f(s)) = \varepsilon_{Int(rng(f))}(t) \tag{11}$$

Next, we extend this notion of refinement with a fixed mapping to a notion of refinement which abstracts from this mapping. This is called *entity refinement*. Interworking s is an entity refinement of Interworking t, notation $s \sqsubseteq t$ if and only if there exists a refinement mapping f such that $s \sqsubseteq_f t$. This is formally expressed as follows:

$$s \sqsubseteq t \quad \text{iff} \quad \exists_{f: EID \hookrightarrow EID} \; s \sqsubseteq_f t \tag{12}$$

Example 1. As an illustration of this algebraic definition of refinement, the refinement relation between the Interworkings in Fig. 4 is computed. Semantically the Interworkings are represented by

$$\begin{aligned} C = \langle & c(p1,p2,m3) \between_{\mathrm{w}} c(p2,q1,m1) \between_{\mathrm{w}} c(q1,q2,m4) \between_{\mathrm{w}} c(q2,q1,m5) \\ & \between_{\mathrm{w}} c(q1,p2,m2) \between_{\mathrm{w}} c(p2,p1,m6), \{p_1,p_2,q_1,q_2\}\rangle \end{aligned} \tag{13}$$

$$D = \langle c(p,q,m1) \between_{\mathrm{w}} c(q,p,m2), \{p,q\}\rangle \tag{14}$$

Elimination of the $\between_{\mathrm{w}}$ yields the following equations

$$\begin{aligned} C = \langle & c(p1,p2,m3) \cdot c(p2,q1,m1) \cdot c(q1,q2,m4) \cdot c(q2,q1,m5) \\ & \cdot c(q1,p2,m2) \cdot c(p2,p1,m6), \{p1,p2,q1,q2\}\rangle \end{aligned} \tag{15}$$

$$D = \langle c(p,q,m1) \cdot c(q,p,m2), \{p,q\}\rangle \tag{16}$$

The refinement mapping f is given by $f(p1) = f(p2) = p$ and $f(q1) = f(q2) = q$. First, we rename the entities of Interworking C according to f.

$$\begin{aligned} \rho_f(C) = \langle & c(p,p,m3) \cdot c(p,q,m1) \cdot c(q,q,m4) \cdot c(q,q,m5) \cdot c(q,p,m2) \\ & \cdot c(p,p,m6), \{p,q\}\rangle \end{aligned} \tag{17}$$

The set of actions which should be removed is given by

$$Int(rng(f)) = \{c(p,p,m), c(q,q,m) \mid m \in MID\} \tag{18}$$

Removing these actions from the Interworkings $\rho_f(C)$ and D results in the following equations

$$\varepsilon_{Int(rng(f))}(\rho_f(C)) = \langle c(p,q,m1) \cdot c(q,p,m2), \{p,q\}\rangle \tag{19}$$

$$\varepsilon_{Int(rng(f))}(D) = \langle c(p,q,m1) \cdot c(q,p,m2), \{p,q\}\rangle \tag{20}$$

We can conclude that Interworking C is an f-refinement of Interworking D.

For the entity refinement relation we have the following properties.

Proposition12 (Reflexivity). *For all closed entity-labeled processes s we have*

$$s \sqsubseteq s \tag{21}$$

Proof. We have to show that there exists a partial mapping f such that $s \sqsubseteq_f s$. Take the mapping f with empty domain. Then s is an f-refinement of s. □

Proposition13 (Transitivity). *For all closed entity-labeled processes s, t, u we have*

$$s \sqsubseteq t \text{ and } t \sqsubseteq u \quad \text{implies} \quad s \sqsubseteq u \tag{22}$$

Proof. Let F be some set and let $f : F \hookrightarrow F$ be a partial function. For all $G \subseteq F$ the extension of f with respect to G, notation f^G, is, for all $x \in F$, defined by

$$f^G(x) = \begin{cases} f(x) & \text{if } x \in dom(f) \\ x & \text{if } x \notin dom(f) \wedge x \in G \\ \text{undefined} & \text{if } x \notin dom(f) \wedge x \notin G \end{cases} \tag{23}$$

Suppose that there exist $f, g : EID \hookrightarrow EID$ such that $s \sqsubseteq_f t$ and $t \sqsubseteq_g u$. Then define $h = g^{rng(f)} \circ f^{dom(g)}$. It is our claim that $s \sqsubseteq_h u$. The proof of this claim is omitted. □

We do not have that the relation $\sqsubseteq$ is anti-symmetrical. This is due to the treatment of internal actions. Consider, for example, the Interworkings $s = \langle c(p,p,m), \{p\}\rangle$ and $t = \langle c(p,p,n), \{p\}\rangle$. Then we have $s \sqsubseteq t$ and $t \sqsubseteq s$, but we do not have $s = t$. For Interworkings without internal communications we do have antisymmetry of entity refinement. So, for Interworkings without internal communication the entity refinement relation is a partial ordering. For the more general class of Interworkings entity refinement is a pre-order.

5 Conclusions

We have given a semantics of Interworkings in which we solved some problems encountered in a former semantics and which allows a definition of entity refinement. The reformulation of the semantics has the following benefits. First, it is now possible to express an empty Interworking. Its semantics is simply the empty process. Next, by extending the processes to entity-labeled processes, the Interworking-merge became an associative operator. Further, we have solved an anomaly described in [8]. We will explain this in short and refer to [8] for an example.

Consider Interworkings A and B, where B is an exact copy of A with the difference that B has one extra entity e without any behaviour. In the old semantics there was no distinction between A and B. However, there is a good reason to make a distinction between these. Suppose there is an Interworking C

that contains entity e such that there is a message to e. Now consider placing A, respectively B in parallel with C. Intuitively, A and C can be merged consistently with respect to entity e, whereas B and C cannot. This is because the communication behaviour of e in B is different from the behaviour of e in C.

A refinement check was already implemented in the Interworking ToolSet based upon an informal explanation by means of examples. The formal definitions provided in this paper seem to correspond well to the existing implementation.

References

1. J.C.M. Baeten and J.A. Bergstra. Global Renaming Operators in Concrete Process Algebra. *Information and Computation*, 78(3):205–245, 1988.
2. J.C.M. Baeten and C. Verhoef. Concrete Process Algebra. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science, volume IV: Semantic Modelling*, pages 149–268. Oxford University Press, 1995.
3. J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, Cambridge, 1990.
4. J.A. Bergstra and J.W. Klop. Process Algebra for Synchronous Communication. *Information & Control*, 60(1/3):109–137, 1984.
5. ITU-TS. *ITU-TS Recommendation Z.120: Message Sequence Chart (MSC)*. ITU-TS, Geneva, 1994.
6. ITU-TS. *ITU-TS Recommendation Z.120 Annex B: Algebraic semantics of Message Sequence Charts*. ITU-TS, Geneva, 1995.
7. S. Mauw and M.A. Reniers. An Algebraic Semantics of Basic Message Sequence Charts. *The Computer Journal*, 37(4):269–277, 1994.
8. S. Mauw and M.A. Reniers. Empty Interworkings and Refinement - Semantics of Interworkings Revised. Computing science report 95-12, Department of Computing Science, Eindhoven University of Technology, 1995.
9. S. Mauw, M. van Wijk, and T. Winter. A Formal Semantics of Synchronous Interworkings. In O. Færgemand and A. Sarma, editors, *SDL'93 Using Objects*, Proc. Sixth SDL Forum, pages 167–178, Darmstadt, 1993. Elsevier, Amsterdam.
10. S. Mauw and T. Winter. A Prototype Toolset for Interworkings. *Philips Telecommunication Review*, 51(3):41–45, 1993.
11. A. Rensink and H. Wehrheim. Weak Sequential Composition in Process Algebras. In B. Jonsson and J. Parrow, editors, *CONCUR'94: Concurrency Theory*, volume 836 of *Lecture Notes in Computer Science*, pages 226–241. Springer-Verlag, 1994.
12. C. Verhoef. A General Conservative Extension Theorem in Process Algebra. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi (PROCOMET '94)*, volume 56 of *IFIP Transactions A*, pages 149–168. North-Holland, 1994.
13. J.L.M. Vrancken. *Studies in Process Algebra, Algebraic Specifications and Parallelism*. PhD thesis, University of Amsterdam, 1991.

This article was processed using the LaTeX macro package with LLNCS style