

Empty Interworkings and Refinement

Semantics of Interworkings Revised

S. Mauw, 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

The semantics for Interworkings from [MvWW93] does not give a proper meaning to empty entities and empty Interworkings. Furthermore, the process algebra considered has to be extended in order to define refinement of Interworkings. For these purposes we give a revision and extension of the semantics.

1 Introduction

Interworkings are used for the graphical presentation of system traces. An Interworking describes the communication behaviour of system components. Interworkings are similar to Message Sequence Charts ([IT94]), 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.

A first proposal for the syntax and semantics of Interworkings is given in [MvWW93] and [MW93] contains a description of a tool set for Interworkings. The semantics are given via a translation into process algebra ([BK84a, BK84b, BV95, BW90]). Communications are translated into atomic actions and two composition operators are defined: the interworking sequencing (\circ_{iw}) for vertical composition and the interworking merge (\parallel_{iw}) for horizontal composition. In [vdBG95] Van den Brink and Griffioen describe an extension of Interworkings with discrete absolute time features by labelling actions with a time stamp and by labelling actions with a discrete time interval.

An Interworking consists of entities, represented by vertical axes, and messages, represented by horizontal arrows. The intuition behind the semantics is as follows. If two messages share an entity, the highest drawn message is executed first. Two messages which are not ordered in this way directly (i.e. via a shared entity) or indirectly (via a number of communications) may be executed in any order (see e.g. Figure 2D in which only one execution order is allowed: $m1$, $m2$, $m3$). This is expressed formally in the definition of the interworking sequencing operator. The interworking merge operator is explained below.

Although the semantics in [MvWW93] are consistent, we are not completely satisfied with it, especially with respect to empty entities and empty Interworkings. These notions imply introduction of the so-called *empty process* into the process algebra. This extended process algebra is also needed if we define *refinement* as introduced in [MvWW92]. The problem encountered in the semantics of [MvWW93] with respect to empty entities is the following. Consider the Interworkings from Figure 1. The difference between the Interworkings A and B is that B contains an entity s while A does not. In fact Interworking B specifies that there can be *no* communication with entity s , whereas Interworking A does not say anything about the interaction with s .

*Part of this research is funded by Philips Research Laboratories Eindhoven.

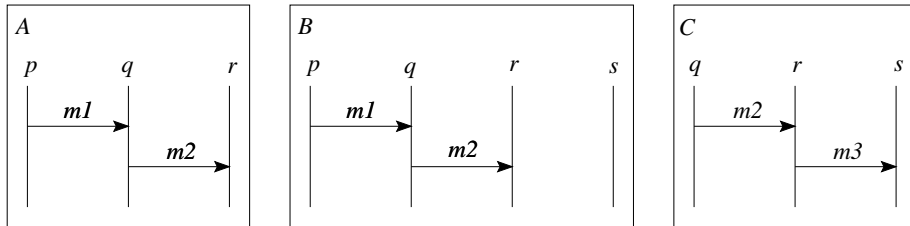


Figure 1: Interworkings A , B and C

If we merge Interworkings A and C (from Figure 1), we first calculate the set of common entities of A and C , which is $\{q, r\}$. The result is such that all communications between entities within this set are present in both A and C . The result is Interworking D shown in Figure 2.

In Interworking B entity s is present, although not participating in any communication action. If we merge Interworkings B and C , the common entities from B and C are $\{q, r, s\}$. Thus B and C have to comply also with respect to the communication of $m3$ from r to s . Since B does not have this communication, a deadlock results as depicted by two horizontal bars in the Interworking E from Figure 2.

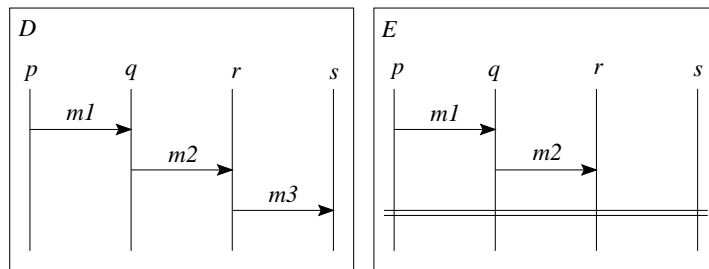


Figure 2: Interworkings D and E (merge of A resp. B with C)

We conclude that intuitively the merge of A and C is different from the merge of B and C . In the semantics of [MvWW93], however, A and B have exactly the same interpretation. The empty entity s is simply neglected, so $A \parallel_{iw} C$ and $B \parallel_{iw} C$ are equal. We solve this difference in intuition and semantics by enriching the process which is the interpretation of a given Interworking with explicit information about the entities which are present.

The second reason for revising the semantics of Interworkings is that we give a formal definition of the notion of *refinement*. An Interworking is a refinement of another Interworking if they have the same behaviour after aggregation of a number of entities into one single entity. This *implementation relation* is very useful for expressing levels of abstraction and thus allows for a top-down specification style.

This paper is organized as follows. In Section 2 we give a formal definition of the interworking operators. Section 3 contains several properties of the interworking sequencing and interworking merge operators. The Interworking refinement is defined in Section 4.

Acknowledgements We would like to thank Thijs Winter (Philips Hilversum) and Mark van Wijk for cooperating on a preliminary, although never published, version of Interworkings with refinement. Furthermore, we thank Jos Baeten, Twan Basten and Hans Mulder (all Eindhoven University of Technology) for their criticism. Their comments have been very helpful in obtaining the results we did. The anonymous reviewers and Jan Gerben Wijnstra (Philips Research Laboratories Eindhoven) are acknowledged for their comments on a preliminary version of this paper.

We are grateful to Loe Feijs (Philips Research Laboratories Eindhoven) for his comments on the refinement of Interworkings.

2 Process Algebra for Interworkings

In this section we will extend the process algebra $BPA_{iw}(A, EID, E)$ from [MvWW93] with the empty process (ε). The parameters A , EID and E are the set of atomic actions, a universe of entity identifiers and a mapping from atomic actions to entity identifiers, respectively. We start by giving the process algebra $BPA_{\delta, \varepsilon}(A)$ from [BW90, BV95]. This process algebra is extended to the process algebra $IWD_{\varepsilon}(A, EID, E)$, i.e., *Process Algebra for Interworking Diagrams*, with the operator interworking sequencing (\circ_{iw}) and some auxiliary operators (\mathbf{L}_{iw} , \mathbf{R}_{iw} , and \surd). Then we extend $IWD_{\varepsilon}(A, EID, E)$ with the E -interworking merge (\parallel_{iw}^E) and some auxiliary operators (\llbracket_{iw}^E , \lceil_{iw}^E). The resulting process algebra is called $IW_{\varepsilon}(A, EID, E)$. Finally, we extend the processes representing Interworkings with an additional label representing the entities from which the Interworking is built. This process algebra will be called $IWE_{\varepsilon}(A, EID, E)$, i.e. *Process Algebra for Interworkings with (empty) entities*. We give for each of the process algebras a structured operational semantics in the style of Plotkin [Plo81, Plo83].

In the case of Interworkings, the three parameters of the algebraic theories are instantiated as follows: $A = \{c(p, q, m) \mid p, q \in EID, m \in MID\}$, where MID is some set of *message identifiers*, EID is some set of *entity identifiers* and E is a function which associates to each atomic action from A a set of entity identifiers: $E(c(p, q, m)) = \{p, q\}$. In fact, with Interworkings, there are two parameters: the set of entity identifiers EID and the set of message identifiers MID , and a constructor function for the atomic actions $c : EID \times EID \times MID \rightarrow A$. We have chosen for the approach with three parameters to cover applications where an entity function must be defined explicitly, because it can not be obtained from the atomic actions.

2.1 Basic Process Algebra with Deadlock and Empty Process

We will give a brief introduction to the process algebra $BPA_{\delta, \varepsilon}(A)$ [BV95, BW90]. 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 all constants of the process algebra is denoted by $A_{\delta, \varepsilon}$.

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.

$x + y$	=	$y + x$	A1	$\delta + x$	=	x	A6
$(x + y) + z$	=	$x + (y + z)$	A2	$\delta \cdot x$	=	δ	A7
$x + x$	=	x	A3	$x \cdot \varepsilon$	=	x	A8
$(x + y) \cdot z$	=	$x \cdot z + y \cdot z$	A4	$\varepsilon \cdot x$	=	x	A9
$(x \cdot y) \cdot z$	=	$x \cdot (y \cdot z)$	A5				

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

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.

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' and the 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 [BV93].

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

Table 2: Structured Operational Semantics of $BPA_{\delta,\varepsilon}(A)$

Finally we would like to mention the following well-known result from literature (e.g. [BW90]): The process algebra $BPA_{\delta,\varepsilon}(A)$ is a sound and complete axiomatization of bisimulation equivalence, notation $\xrightarrow{\sim}$, on the closed $BPA_{\delta,\varepsilon}(A)$ terms. This result will be used in the following sections when relating the extended process algebras to $BPA_{\delta,\varepsilon}(A)$.

2.2 Axiomatization of Interworking Sequencing

In this section we will extend the process algebra $BPA_{\delta,\varepsilon}(A)$ from the previous section with an *entity function* on the atomic actions from A and the interworking sequencing operator \circ_w . The resulting process algebra is denoted by $IWD_\varepsilon(A, EID, E)$. The entity function $E : A \rightarrow \mathcal{P}(EID)$ associates to every atomic action from the set A a set of entity identifiers from EID . Intuitively the entities of the atomic action a are the functional blocks on which the atomic action is defined/executed; these entities are called *active entities*. The universe of entity identifiers EID and the entity function E are considered parameters of the process algebra.

Based on the entity function E on atomic actions we can associate to every closed term over the signature of the process algebra a set of entities; these are the entities to which the atomic actions of the process refer. This is done by extending the entity function on atomic actions to an entity function E on process terms. This extension is, for $a \in A$ and x and y arbitrary processes, defined in Table 3. For the atomic actions from A , viewed as a process, the entities are given already by the entity function on atomic actions. For the atomic actions ε and δ we take $E(\varepsilon) = \emptyset$ and $E(\delta) = \emptyset$. For more complex processes the entities are obtained from the atomic actions the process is built from.

$E(\varepsilon)$	$=$	\emptyset
$E(\delta)$	$=$	\emptyset
$E(a \cdot x)$	$=$	$E(a) \cup E(x)$
$E(x + y)$	$=$	$E(x) \cup E(y)$

Table 3: Active Entities of an Interworking

The interworking sequencing of two processes x and y is their parallel execution with the restriction that the right-hand side process may execute an action only if the entities of that action are disjoint from the entities of the left-hand side process. The interworking sequencing operator is similar to the *weak sequential composition* operator from [RW94]. As a starting point for the axiomatization we take the axioms for the interworking sequencing as they are given in [MvWW93]. To obtain an axiomatization of the interworking sequencing operator in the context of empty Interworkings an approach is followed which compares easily with the step from the free merge in a setting without the empty process to a free merge in a setting with the empty process [BW90].

The axiomatization of α_{iw} as presented in [MvWW93] uses the two auxiliary operators $\mathbf{L}\alpha_{iw}$ and $\mathbf{R}\alpha_{iw}$. The process $x \mathbf{L}\alpha_{iw} y$ behaves like the process $x \alpha_{iw} y$ with the restriction that the first action to be executed must originate from process x . The process $x \mathbf{R}\alpha_{iw} y$ also behaves like the process $x \alpha_{iw} y$ but this time with the restriction that the first action to be executed must be from process y .

Intuitively, we want ε to be a unit for the interworking sequencing, i.e. $\varepsilon \alpha_{iw} x = x \alpha_{iw} \varepsilon = x$. In particular we also want to have that $\varepsilon \alpha_{iw} \varepsilon = \varepsilon$. The interpretation of $x \mathbf{L}\alpha_{iw} y$ is that the process x is forced to do the first step. Since ε is unable to do any step, it seems plausible to define $\varepsilon \mathbf{L}\alpha_{iw} y = \delta$. Consequently, we also define $x \mathbf{R}\alpha_{iw} \varepsilon = \delta$. If we apply this in the definition of the sequencing operator as given in [MvWW93] we get $\varepsilon \alpha_{iw} \varepsilon = \varepsilon \mathbf{L}\alpha_{iw} \varepsilon + \varepsilon \mathbf{R}\alpha_{iw} \varepsilon = \delta + \delta = \delta$. This is not what we want and therefore we need the additional operator \surd as given in Table 4. This operator has also been used by Baeten and Weijland [BW90] in axiomatizing the free merge in a process algebra containing the empty process. The definition of the interworking sequencing operator is given in Table 4.

$x \alpha_{iw} y$	$=$	$x \mathbf{L}\alpha_{iw} y + x \mathbf{R}\alpha_{iw} y + \surd(x) \cdot \surd(y)$	S1
$\varepsilon \mathbf{L}\alpha_{iw} x$	$=$	δ	LS1
$\delta \mathbf{L}\alpha_{iw} x$	$=$	δ	LS2
$a \cdot x \mathbf{L}\alpha_{iw} y$	$=$	$a \cdot (x \alpha_{iw} y)$	LS3
$(x + y) \mathbf{L}\alpha_{iw} z$	$=$	$x \mathbf{L}\alpha_{iw} z + y \mathbf{L}\alpha_{iw} z$	LS4
$x \mathbf{R}\alpha_{iw} \varepsilon$	$=$	δ	RS1
$x \mathbf{R}\alpha_{iw} \delta$	$=$	δ	RS2
$x \mathbf{R}\alpha_{iw} a \cdot y$	$=$	$a \cdot (x \alpha_{iw} y)$	if $E(a) \cap E(x) = \emptyset$ RS3
$x \mathbf{R}\alpha_{iw} a \cdot y$	$=$	δ	if $E(a) \cap E(x) \neq \emptyset$ RS4
$x \mathbf{R}\alpha_{iw} (y + z)$	$=$	$x \mathbf{R}\alpha_{iw} y + x \mathbf{R}\alpha_{iw} z$	RS5
$\surd(\varepsilon)$	$=$	ε	T1
$\surd(\delta)$	$=$	δ	T2
$\surd(a \cdot x)$	$=$	δ	T3
$\surd(x + y)$	$=$	$\surd(x) + \surd(y)$	T4

Table 4: Axioms for interworking sequencing

The structured operational semantics of the interworking sequencing and of the auxiliary operators is given in Table 5. The term deduction system $T(IWD_\varepsilon(A, EID, E))$ consists of the deduction rules of $T(BPA_{\delta, \varepsilon}(A))$ and the deduction rules of Table 5.

Next, we will formulate some interesting theorems concerning this process algebra. These theorems relate the process algebra $IWD_\varepsilon(A, EID, E)$ to the process algebra $BPA_{\delta, \varepsilon}(A)$ and to the structured operational semantics as given by the term deduction systems.

Theorem 2.2.1 (Congruence) *Bisimulation equivalence is a congruence for the function symbols in the signature of $IWD_\varepsilon(A, EID, E)$.*

Proof It is straightforward to verify that the deduction rules of the term deduction system which

$\frac{x \downarrow, y \downarrow}{x \circ_{iw} y \downarrow}$	$\frac{x \xrightarrow{a} x'}{x \circ_{iw} y \xrightarrow{a} x' \circ_{iw} y}$	$\frac{y \xrightarrow{a} y', E(a) \cap E(x) = \emptyset}{x \circ_{iw} y \xrightarrow{a} x \circ_{iw} y'}$
$\frac{x \downarrow}{\sqrt{(x)} \downarrow}$	$\frac{x \xrightarrow{a} x'}{x \mathbf{L}\circ_{iw} y \xrightarrow{a} x' \circ_{iw} y}$	$\frac{y \xrightarrow{a} y', E(a) \cap E(x) = \emptyset}{x \mathbf{R}\circ_{iw} y \xrightarrow{a} x \circ_{iw} y'}$

Table 5: Structured Operational Semantics of interworking sequencing and auxiliary operators

consists of the deduction rules from Tables 2 and 5 are in *path* format. By the congruence theorem from Baeten and Verhoef [BV93] it follows immediately that bisimulation is a congruence on the closed $IWD_\varepsilon(A, EID, E)$ terms.

Theorem 2.2.2 (Soundness) *The process algebra $IWD_\varepsilon(A, EID, E)$ is a sound axiomatization of bisimulation equivalence on closed $IWD_\varepsilon(A, EID, E)$ terms.*

Proof Due to the fact that bisimulation is a congruence, we only have to verify the soundness of each axiom. For the axioms S1, LS3,4, and RS3,5 we relate the left-hand side to the right-hand side and we add the diagonal (i.e. we relate each term to itself). For the other axioms we only relate the left-hand side to the right-hand side.

Theorem 2.2.3 (Conservativity) *The process algebra $IWD_\varepsilon(A, EID, E)$ is a conservative extension of the process algebra $BPA_{\delta, \varepsilon}(A)$.*

Proof The proof of this theorem uses the approach of Verhoef [Ver94]. The theorem follows from the following observations:

- 1) Bisimulation is definable in terms of predicate and relation symbols only,
- 2) the process algebra $BPA_{\delta, \varepsilon}(A)$ is a complete axiomatization of bisimulation equivalence on closed $BPA_{\delta, \varepsilon}(A)$ terms (see [BV95, BW90]),
- 3) the process algebra $IWD_\varepsilon(A, EID, E)$ is a sound axiomatization of bisimulation equivalence on closed $IWD_\varepsilon(A, EID, E)$ terms (see Theorem 2.2.2),
- 4) the term deduction system $T(BPA_{\delta, \varepsilon}(A))$ is pure¹, well-founded² and in path format, and
- 5) the term deduction system $T(IWD_\varepsilon(A, EID, E))$ is in path format³.

Theorem 2.2.4 (Elimination) *The process algebra $IWD_\varepsilon(A, EID, E)$ has the elimination property for the process algebra $BPA_{\delta, \varepsilon}(A)$.*

Proof The term rewrite system associated with the axioms A3-A7 from Table 1 and the axioms from Tables 3 and 4 and the additional rewriting rules from Table 6 is strongly normalizing. This can be proven with the method of the lexicographical path ordering [KL80, Klo92]. Note that the additional rewriting rules are, for closed terms, derivable from the axioms of $IWD_\varepsilon(A, EID, E)$.

If we additionally show that every normal form of the closed terms is in fact a closed $BPA_{\delta, \varepsilon}(A)$ term then the theorem follows easily. Thereto, suppose that s is a normal form with respect to the term rewrite system and suppose that s is not a closed $BPA_{\delta, \varepsilon}(A)$ term. Then s must contain at

¹For a definition of *pure* term deduction systems see [BV93].

²For a definition of *well-founded* term deduction systems see [BV93].

³For a definition of the *path format* see [BV93].

$a \mathbf{L}_{\alpha_w} x$	\rightarrow	$a \cdot x$	
$x \mathbf{R}_{\alpha_w} a$	\rightarrow	$a \cdot x$	if $E(a) \cap E(x) = \emptyset$
$x \mathbf{R}_{\alpha_w} a$	\rightarrow	δ	if $E(a) \cap E(x) \neq \emptyset$
$\surd(a)$	\rightarrow	δ	

Table 6: Additional rewriting rules for $IWD_\varepsilon(A, EID, E)$

least one occurrence of the operators α_w , \mathbf{L}_{α_w} , \mathbf{R}_{α_w} , or \surd . Take a smallest subterm of s which is headed by one of these operators. In any case it follows that the operands of this operator are closed $BPA_{\delta,\varepsilon}(A)$ terms. From that it is easily seen that a rewrite rule must be applicable, which contradicts the assumptions. Therefore, we conclude that every normal form of a closed $IWD_\varepsilon(A, EID, E)$ term is a closed $BPA_{\delta,\varepsilon}(A)$ term.

Theorem 2.2.5 (Completeness) *The process algebra $IWD_\varepsilon(A, EID, E)$ is a complete axiomatization of bisimulation equivalence on closed $IWD_\varepsilon(A, EID, E)$ terms.*

Proof By the *General Completeness Theorem* of Verhoef [Ver94], the completeness of the process algebra $IWD_\varepsilon(A, EID, E)$ follows immediately from the propositions which are used in the proof of Theorem 2.2.3 and the fact that $IWD_\varepsilon(A, EID, E)$ has the elimination property for $BPA_{\delta,\varepsilon}(A)$ (see Theorem 2.2.4).

2.3 Axiomatization of Interworking Merge

In this section we will extend the process algebra $IWD_\varepsilon(A, EID, E)$ from the previous section with the interworking merge operator (\parallel_{iw}). The resulting process algebra is called $IWE_\varepsilon(A, EID, E)$. Thereto, we first describe $IW_\varepsilon(A, EID, E)$, the extension of $IWD_\varepsilon(A, EID, E)$ with the E -interworking merge operator (\parallel_{iw}^E). 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. After that we introduce tuples of process terms and entity sets. On this new structure we define the interworking merge operator.

As we have shown in the introduction there is a problem with axiomatizing the interworking merge operator. From an expression representing an Interworking it is not possible to determine the empty entities, since the Interworking can have empty entities which are not represented in the atomic actions describing the Interworking. In this section we will solve this problem by associating to every closed term of the process algebra a label denoting the entities which are present. With that additional information it is straightforward to give an axiomatization of the interworking merge. As was done in [MvWW93] the interworking merge is expressed in terms of the E -interworking merge operator and the common entities of the operands.

The axiomatization of the S -interworking merge as presented in [MvWW93] uses the auxiliary operators left S -interworking merge \llbracket_{iw}^S and synchronization interworking merge \mid_{iw}^S with S a set of atomic actions. We will use similar auxiliary operators only now labelled 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 $x \parallel_{iw}^E y$ 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 $x \llbracket_{iw}^E y$ behaves like the process $x \parallel_{iw}^E y$ with the restriction that the first action must come from process x and that action does not have to synchronize with an action from y . The process $x \mid_{iw}^E y$ behaves as the process $x \parallel_{iw}^E y$ with the restriction that the first action to be executed must be a synchronization.

Since we have defined the entities of the empty process to be the empty set, it seems plausible that we define $x \parallel_{iw} \varepsilon = x$, and in particular $\varepsilon \parallel_{iw}^E \varepsilon = \varepsilon$. The interpretation of $x \parallel_{iw}^E y$ and $x \mid_{iw}^E y$, however, make it reasonable to define $\varepsilon \parallel_{iw}^E x = \delta$ and $\varepsilon \mid_{iw}^E x = x \mid_{iw}^E \varepsilon = \delta$. With respect to the main axiom for the interworking merge from [MvWW93]: $x \parallel_{iw}^S y = x \parallel_{iw}^S y + y \parallel_{iw}^S x + x \mid_{iw}^S y$, we derive $\varepsilon \parallel_{iw}^E \varepsilon = \varepsilon \parallel_{iw}^E \varepsilon + \varepsilon \parallel_{iw}^E \varepsilon + \varepsilon \mid_{iw}^E \varepsilon = \delta + \delta + \delta = \delta$. In terms of Interworkings this means that two empty Interworkings are not consistent which is not what we want. Therefore, we need to redefine the interworking merge operator in a way similar to the definition of the sequencing operator. The new axioms for the E -interworking merge operator are given in Table 7. Recall that the axioms for the termination operator are given in Table 4.

$x \parallel_{iw}^E y$	$=$	$x \parallel_{iw}^E y + y \parallel_{iw}^E x + x \mid_{iw}^E y + \sqrt{(x)} \cdot \sqrt{(y)}$	EM1
$\varepsilon \parallel_{iw}^E x$	$=$	δ	LEM1
$\delta \parallel_{iw}^E x$	$=$	δ	LEM2
$a \cdot x \parallel_{iw}^E y$	$=$	$a \cdot (x \parallel_{iw}^E y)$	if $E(a) \not\subseteq E$ LEM3
$a \cdot x \parallel_{iw}^E y$	$=$	δ	if $E(a) \subseteq E$ LEM4
$(x + y) \parallel_{iw}^E z$	$=$	$x \parallel_{iw}^E z + y \parallel_{iw}^E z$	LEM5
$\varepsilon \mid_{iw}^E x$	$=$	δ	SEM1
$x \mid_{iw}^E \varepsilon$	$=$	δ	SEM2
$\delta \mid_{iw}^E x$	$=$	δ	SEM3
$x \mid_{iw}^E \delta$	$=$	δ	SEM4
$a \cdot x \mid_{iw}^E b \cdot y$	$=$	$a \cdot (x \parallel_{iw}^E y)$	if $a \equiv b \wedge E(a) \subseteq E$ SEM5
$a \cdot x \mid_{iw}^E b \cdot y$	$=$	δ	if $a \not\equiv b \vee E(a) \not\subseteq E$ SEM6
$(x + y) \mid_{iw}^E z$	$=$	$x \mid_{iw}^E z + y \mid_{iw}^E z$	SEM7
$x \mid_{iw}^E (y + z)$	$=$	$x \mid_{iw}^E y + x \mid_{iw}^E z$	SEM8

Table 7: Axioms of E -interworking merge

Next, we present a structured operational semantics for the operators which are introduced in this section. The term deduction system $T(IW_\varepsilon(A, EID, E))$ consists of the deduction rules of $T(IWD_\varepsilon(A, EID, E))$ and the deduction rules of Table 8.

$\frac{x \downarrow, y \downarrow}{x \parallel_{iw}^E y \downarrow}$	$\frac{x \xrightarrow{a} x', y \xrightarrow{a} y', E(a) \subseteq E}{x \parallel_{iw}^E y \xrightarrow{a} x' \parallel_{iw}^E y'}$
$\frac{x \xrightarrow{a} x', E(a) \not\subseteq E}{x \parallel_{iw}^E y \xrightarrow{a} x' \parallel_{iw}^E y}$	$\frac{y \xrightarrow{a} y', E(a) \not\subseteq E}{x \parallel_{iw}^E y \xrightarrow{a} x \parallel_{iw}^E y'}$
$\frac{x \xrightarrow{a} x', E(a) \not\subseteq E}{x \parallel_{iw}^E y \xrightarrow{a} x' \parallel_{iw}^E y}$	$\frac{x \xrightarrow{a} x', y \xrightarrow{a} y', E(a) \subseteq E}{x \mid_{iw}^E y \xrightarrow{a} x' \mid_{iw}^E y'}$

Table 8: Structured Operational Semantics of E -interworking merge and auxiliary operators

Before we turn to the interworking merge operator we list results similar to those of the previous section for the process algebra $IW_\varepsilon(A, EID, E)$. The proofs are omitted.

Theorem 2.3.1 (Congruence) *Bisimulation equivalence is a congruence for the function symbols in the signature of $IW_\varepsilon(A, EID, E)$.*

Theorem 2.3.2 (Soundness) *The process algebra $IW_\varepsilon(A, EID, E)$ is a sound axiomatization of bisimulation equivalence on closed $IW_\varepsilon(A, EID, E)$ terms.*

Theorem 2.3.3 (Conservativity) *The process algebra $IW_\varepsilon(A, EID, E)$ is a conservative extension of the process algebra $IWD_\varepsilon(A, EID, E)$.*

Theorem 2.3.4 (Elimination) *The process algebra $IW_\varepsilon(A, EID, E)$ has the elimination property for the process algebra $IWD_\varepsilon(A, EID, E)$.*

Theorem 2.3.5 (Completeness) *The process algebra $IW_\varepsilon(A, EID, E)$ is a complete axiomatization of bisimulation equivalence on closed $IW_\varepsilon(A, EID, E)$ terms.*

Now that we have given the axioms and structured operational semantics of the E -interworking merge we will discuss 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 can not be obtained from the process term representing it, 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 together with the active entities of x (i.e. $E(x)$) 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 $\langle x, E \rangle$. We do not require that the set E consists of all entities of the interworking; instead we require that the entities of the interworking are given by $E \cup E(x)$. The reason for this choice is that in this setting every tuple from which the dynamical behaviour describes an interworking, can be interpreted as an Interworking in the extended semantics. Such a tuple $\langle x, E \rangle$ will be called a labelled process. On labelled processes we define the operators $+$, \cdot , α_{iw} , \parallel_{iw}^E and \parallel_{iw} . The set of all labelled processes is called LP . The axioms for labelled processes are, for x, y processes, E, E_1 , and $E_2 \subseteq EID$, and $\oplus \in \{+, \cdot, \alpha_{iw}, \parallel_{iw}^E\}$, given in Table 9.

$\langle x, E \rangle$	$=$	$\langle x, E \cup E(x) \rangle$	if $E(x) \not\subseteq E$
$\langle x, E_1 \rangle \oplus \langle y, E_2 \rangle$	$=$	$\langle x \oplus y, E_1 \cup E_2 \rangle$	if $E(x) \subseteq E_1 \wedge E(y) \subseteq E_2$
$\langle x, E_1 \rangle \parallel_{iw} \langle y, E_2 \rangle$	$=$	$\langle x \parallel_{iw}^{E_1 \cap E_2} y, E_1 \cup E_2 \rangle$	if $E(x) \subseteq E_1 \wedge E(y) \subseteq E_2$

Table 9: Extension to labelled processes

Example 2.3.6 The labelled processes associated to the Interworkings A , B , and C from Figure 1 are the following.

$$\begin{aligned}
A &= \langle c(p, q, m1) \alpha_{iw} c(q, r, m2), \{p, q, r\} \rangle \\
B &= \langle c(p, q, m1) \alpha_w c(q, r, m2), \{p, q, r, s\} \rangle \\
C &= \langle c(q, r, m2) \alpha_w c(r, s, m3), \{q, r, s\} \rangle
\end{aligned}$$

Then we have the following computations for $A \parallel_{iw} C$ and $B \parallel_{iw} C$.

$$\begin{aligned}
A \parallel_{iw} C &= \langle c(p, q, m1) \alpha_w c(q, r, m2), \{p, q, r\} \rangle \parallel_{iw} \langle c(q, r, m2) \alpha_w c(r, s, m3), \{q, r, s\} \rangle \\
&= \langle c(p, q, m1) \alpha_w c(q, r, m2) \parallel_{iw}^{\{q, r\}} c(q, r, m2) \alpha_w c(r, s, m3), \{p, q, r, s\} \rangle \\
&= \langle c(p, q, m1) \alpha_w c(q, r, m2) \alpha_w c(r, s, m3), \{p, q, r, s\} \rangle
\end{aligned}$$

and

$$\begin{aligned}
B \parallel_{iw} C &= \langle c(p, q, m1) \alpha_{iw} c(q, r, m2), \{p, q, r, s\} \rangle \parallel_{iw} \langle c(q, r, m2) \alpha_w c(r, s, m3), \{q, r, s\} \rangle \\
&= \langle c(p, q, m1) \alpha_{iw} c(q, r, m2) \parallel_{iw}^{\{q, r, s\}} c(q, r, m2) \alpha_w c(r, s, m3), \{p, q, r, s\} \rangle \\
&= \langle c(p, q, m1) \alpha_{iw} c(q, r, m2) \alpha_w \delta, \{p, q, r, s\} \rangle
\end{aligned}$$

Clearly, the expressions do not denote the same interworking in this extended setting. In other words, the labelled processes are not entity bisimilar (see Definition 2.3.7).

Observe that the role of the empty process as a neutral element for the sequential composition, the interworking sequencing and the interworking merge is, with respect to closed labelled processes, taken over by the labelled process $\langle \varepsilon, \emptyset \rangle$, i.e. $\langle \varepsilon, \emptyset \rangle \cdot x = x = x \cdot \langle \varepsilon, \emptyset \rangle$, $\langle \varepsilon, \emptyset \rangle \alpha_w x = x = x \alpha_w \langle \varepsilon, \emptyset \rangle$, and $\langle \varepsilon, \emptyset \rangle \parallel_{iw} x = x = x \parallel_{iw} \langle \varepsilon, \emptyset \rangle$. The role of deadlock as the neutral element for the alternative composition is taken over by the labelled process $\langle \delta, \emptyset \rangle$, i.e. $\langle \delta, \emptyset \rangle + x = x = x + \langle \delta, \emptyset \rangle$. The role of deadlock as the left zero element for the sequential composition operator is taken over by the labelled process $\langle \delta, EID \rangle$, i.e. $\langle \delta, EID \rangle \cdot x = \langle \delta, EID \rangle$. These properties of the labelled processes can all easily be derived from the axioms from Table 9 and the axioms for the operators in the process algebra.

Next, we define a structured operational semantics for labelled processes. The deduction rules of the term deduction system $T(IWE_\varepsilon(A, EID, E))$ are given by the deduction rules of $T(IW_\varepsilon(A, EID, E))$ and the deduction rules from Table 10.

$\frac{x \downarrow}{\langle x, E \rangle \downarrow}$	$\frac{x \xrightarrow{a} x'}{\langle x, E \rangle \xrightarrow{a} \langle x', E \cup E(x) \rangle}$
$\frac{x \oplus y \downarrow}{\langle x, E \rangle \oplus \langle y, F \rangle \downarrow}$	$\frac{x \oplus y \xrightarrow{a} z}{\langle x, E \rangle \oplus \langle y, F \rangle \xrightarrow{a} \langle z, E \cup F \cup E(x) \cup E(y) \rangle}$
$\frac{\langle x, E \rangle \downarrow, \langle y, F \rangle \downarrow}{\langle x, E \rangle \parallel_{iw} \langle y, F \rangle \downarrow}$	$\frac{x \parallel_{iw}^{(E \cup E(x)) \cap (F \cup E(y))} y \xrightarrow{a} z}{\langle x, E \rangle \parallel_{iw} \langle y, F \rangle \xrightarrow{a} \langle z, E \cup F \cup E(x) \cup E(y) \rangle}$

Table 10: Structured Operational Semantics of labelled processes

Definition 2.3.7 The closed *LP* terms $\langle s, E \rangle$ and $\langle t, F \rangle$ are *entity bisimilar*, $\langle s, E \rangle \xleftrightarrow{\text{e}} \langle t, F \rangle$, if and only if $s \xleftrightarrow{\text{e}} t$ and $E \cup E(x) = F \cup E(y)$.

Theorem 2.3.8 (Congruence) *Entity bisimulation equivalence is a congruence for the function symbols in the signature of $IWE_\varepsilon(A, EID, E)$ which are defined on LP terms.*

Proof The theorem from [BV93] as used in the previous theorems on congruence is not applicable in this case. This theorem is only formulated for strong bisimulation equivalence. Nevertheless we will see that Theorem 2.3.8 is not too hard to prove. Suppose that $\langle x_1, E_1 \rangle \xleftrightarrow{\text{e}} \langle x_2, E_2 \rangle$ and $\langle y_1, F_1 \rangle \xleftrightarrow{\text{e}} \langle y_2, F_2 \rangle$. Now we have to prove, for $\otimes \in \{+, \cdot, \alpha_w, \parallel_{iw}^E, \parallel_{iw}\}$, that $\langle x_1, E_1 \rangle \otimes \langle y_1, F_1 \rangle \xleftrightarrow{\text{e}} \langle x_2, E_2 \rangle \otimes \langle y_2, F_2 \rangle$. From the assumptions we have the following: $x_1 \xleftrightarrow{\text{e}} x_2$, $y_1 \xleftrightarrow{\text{e}} y_2$, $E_1 \cup E(x_1) = E_2 \cup E(x_2)$ and $F_1 \cup E(y_1) = F_2 \cup E(y_2)$.

We will first verify the termination predicate. Suppose that $\langle x_1, E_1 \rangle \oplus \langle y_1, F_1 \rangle \downarrow$. According to the deduction rules this is only the case if $x_1 \oplus y_1 \downarrow$. From $x_1 \xleftrightarrow{\text{e}} x_2$ and $y_1 \xleftrightarrow{\text{e}} y_2$ we obtain $x_2 \oplus y_2 \downarrow$. Using the same deduction rule as before we have $\langle x_2, E_2 \rangle \oplus \langle y_2, F_2 \rangle \downarrow$. In the other direction the proof is analogous. Suppose that $\langle x_1, E_1 \rangle \parallel_{iw} \langle y_1, F_1 \rangle \downarrow$. Then we have $\langle x_1, E_1 \rangle \downarrow$

and $\langle y_1, F_1 \rangle \downarrow$. But then also $x_1 \downarrow$ and $y_1 \downarrow$. By the assumptions we then have $x_2 \downarrow$ and $y_2 \downarrow$. Then we can deduce $\langle x_2, E_2 \rangle \downarrow$ and $\langle y_2, E_2 \rangle \downarrow$ and from that $\langle x_2, E_2 \rangle \parallel_{iw} \langle y_2, F_2 \rangle \downarrow$. In the other direction the proof is analogous.

Next, we will show that every step from the left-hand side can be mimicked by the right-hand side, and vice versa. Also we will show that the resulting labelled processes are entity bisimilar. Suppose that $\langle x_1, E_1 \rangle \oplus \langle y_1, F_1 \rangle \xrightarrow{a} \langle z, G \rangle$. Inspection of the deduction rules gives us that we must have that $x_1 \oplus y_1 \xrightarrow{a} z$ and $G = E_1 \cup F_1 \cup E(x_1) \cup E(y_1)$. From this we get with the assumptions $x_2 \oplus y_2 \xrightarrow{a} z$ and $G = E_2 \cup F_2 \cup E(x_2) \cup E(y_2)$. Therefore, with the same deduction rule we deduce $\langle x_2, E_2 \rangle \oplus \langle y_2, F_2 \rangle \xrightarrow{a} \langle z, G \rangle$. The proof in the other direction is analogous.

Suppose that $\langle x_1, E_1 \rangle \parallel_{iw} \langle y_1, F_1 \rangle \xrightarrow{a} \langle z, G \rangle$. We use the following abbreviations $E'_1 = E_1 \cup E(x_1)$, $E'_2 = E_2 \cup E(x_2)$, $F'_1 = F_1 \cup E(y_1)$ and $F'_2 = F_2 \cup E(y_2)$. Then we must have $x_1 \parallel_{iw}^{E'_1 \cap F'_1} y_1 \xrightarrow{a} z$ and $G = E'_1 \cup F'_1$. By the assumptions we then also have $x_2 \parallel_{iw}^{E'_2 \cap F'_2} y_2 \xrightarrow{a} z$ and $G = E'_2 \cup F'_2$. From this we obtain $\langle x_2, E_2 \rangle \parallel_{iw} \langle y_2, F_2 \rangle \xrightarrow{a} \langle z, G \rangle$. The proof in the other direction is analogous.

Theorem 2.3.9 (Soundness) *The process algebra $IWE_\varepsilon(A, EID, E)$ is a sound axiomatization of bisimulation equivalence on closed $IW_\varepsilon(A, EID, E)$ terms. The process algebra $IWE_\varepsilon(A, EID, E)$ is a sound axiomatization of entity bisimulation on closed LP terms.*

Proof For the first proposition see Theorem 2.3.2 and observe that we did not add any axioms relating closed $IW_\varepsilon(A, EID, E)$ terms. We will prove the second proposition. Since entity bisimulation is a congruence for the closed terms of LP we only have to show that the axioms are sound. For the first axiom relate the left-hand side to the right-hand side. For the other axioms we additionally relate each term to itself.

Theorem 2.3.10 (Conservativity) *The process algebra $IWE_\varepsilon(A, EID, E)$ is a conservative extension of the process algebra $IW_\varepsilon(A, EID, E)$.*

Proof The theorem follows from the following observations:

- 1) Bisimulation is definable in terms of predicate and relation symbols only,
- 2) $IW_\varepsilon(A, EID, E)$ is a complete axiomatization of bisimulation on closed $IW_\varepsilon(A, EID, E)$ terms (see Theorem 2.3.5),
- 3) $IWE_\varepsilon(A, EID, E)$ is a sound axiomatization of bisimulation on closed $IW_\varepsilon(A, EID, E)$ terms (see Theorem 2.3.9), and
- 4) $T(IW_\varepsilon(A, EID, E))$ is pure, well-founded and in path format, and
- 5) $T(IWE_\varepsilon(A, EID, E))$ is in path format.

Definition 2.3.11 *Basic LP terms* are defined inductively as follows:

- 1) if s is a closed $IW_\varepsilon(A, EID, E)$ term and $E \subseteq EID$ such that $E(s) \subseteq E$, then $\langle s, E \rangle$ is a basic LP term
- 2) no other closed LP terms are basic LP terms

Theorem 2.3.12 (Elimination) *For every closed LP term s there exists a basic LP term t such that $IWE_\varepsilon(A, EID, E) \vdash s = t$.*

Proof This theorem is proven with induction on the structure of a closed LP term. First consider the case $s \equiv \langle s', E' \rangle$ (s' a closed $IW_\varepsilon(A, EID, E)$ term and $E' \subseteq EID$). If $E(s') \subseteq E'$ then s is a basic LP term. If $E(s') \not\subseteq E'$ then we have $\langle s', E' \rangle = \langle s', E' \cup E(s') \rangle$ which is a basic LP term. Next, consider the case $s \equiv s_1 \oplus s_2$ (s_1, s_2 closed LP terms) for $\oplus \in \{+, \cdot, \alpha_w, \parallel_{iw}^E\}$. Then we have by induction that s_1 and s_2 are basic LP terms, i.e. $s_1 \equiv \langle t_1, E_1 \rangle$ and $s_2 \equiv \langle t_2, E_2 \rangle$ for some t_1, t_2 closed $IW_\varepsilon(A, EID, E)$ terms and $E_1, E_2 \subseteq EID$. Then we can apply the second axiom to obtain $s = \langle t_1 \oplus t_2, E_1 \cup E_2 \rangle$ which is a basic LP term. Finally, consider the case $s \equiv s_1 \parallel_{iw} s_2$. Again by induction we have that s_1 and s_2 are basic LP terms. Therefore, we have that $s_1 \equiv \langle t_1, E_1 \rangle$ with $E(t_1) \subseteq E_1$ and that $s_2 \equiv \langle t_2, E_2 \rangle$ with $E(t_2) \subseteq E_2$. Then we can apply the third axiom and obtain $s \equiv \langle t_1 \parallel_{iw}^{E_1 \cap E_2} t_2, E_1 \cup E_2 \rangle$ which is a closed LP term.

Theorem 2.3.13 (Completeness) *The process algebra $IWE_\varepsilon(A, EID, E)$ is a complete axiomatization of entity bisimulation on closed LP terms.*

Proof By the elimination theorem we only have to prove this theorem for basic LP terms. Let $\langle s, E_1 \rangle$ and $\langle t, E_2 \rangle$ be basic LP terms such that $\langle s, E_1 \rangle \rightleftharpoons \langle t, E_2 \rangle$. By the definition of entity bisimulation we have $s \rightleftharpoons t$ and $E_1 \cup E(s) = E_1 = E_2 = E_2 \cup E(t)$. In Theorem 2.3.5 we proved that $IW_\varepsilon(A, EID, E)$ is a complete axiomatization of bisimulation equivalence on closed $IW_\varepsilon(A, EID, E)$ terms. So we have $\langle s, E_1 \rangle = \langle t, E_2 \rangle$.

3 Properties

In this section we will show that the interworking sequencing is commutative under the assumption that the active entities of the operands are disjoint and that the interworking sequencing is associative. We will also prove that the interworking merge is both commutative and associative. The interworking merge as defined by [MvWW93] did not have the associativity property. This is a direct consequence of our decision to maintain the entities of an Interworking statically. We can illustrate this with the following example.

Example 3.1 In the semantics of [MvWW93] the Interworkings as shown in Figure 3 are given by $A = c(p, q, m)$ and $B = c(p, q, n)$, whereas in this paper they are represented by $A = \langle c(p, q, m), \{p, q\} \rangle$ and $B = \langle c(p, q, n), \{p, q\} \rangle$.

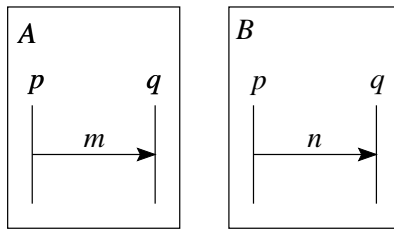


Figure 3: Example Interworkings

We compute the expressions $(A \parallel_{iw} A) \parallel_{iw} B$ and $A \parallel_{iw} (A \parallel_{iw} B)$ in both settings. In the semantics of [MvWW93] we have the following computations.

$$\begin{aligned}
 (A \parallel_{iw} A) \parallel_{iw} B &= (c(p, q, m) \parallel_{iw} c(p, q, m)) \parallel_{iw} c(p, q, n) \\
 &= c(p, q, m) \parallel_{iw} c(p, q, n) \\
 &= \delta
 \end{aligned}$$

$$\begin{aligned}
A \parallel_{iw} (A \parallel_{iw} B) &= c(p, q, m) \parallel_{iw} (c(p, q, m) \parallel_{iw} c(p, q, n)) \\
&= c(p, q, m) \parallel_{iw} \delta \\
&= c(p, q, m) \cdot \delta
\end{aligned}$$

In the semantics of Interworkings as presented in this paper we have the following computations. Denote the set $\{p, q\}$ by K .

$$\begin{aligned}
(A \parallel_{iw} A) \parallel_{iw} B &= (\langle c(p, q, m), K \rangle \parallel_{iw} \langle c(p, q, m), K \rangle) \parallel_{iw} \langle c(p, q, n), K \rangle \\
&= \langle c(p, q, m) \parallel_{iw}^K c(p, q, m), K \rangle \parallel_{iw} \langle c(p, q, n), K \rangle \\
&= \langle c(p, q, m), K \rangle \parallel_{iw} \langle c(p, q, n), K \rangle \\
&= \langle c(p, q, m) \parallel_{iw}^K c(p, q, n), K \rangle \\
&= \langle \delta, K \rangle
\end{aligned}$$

$$\begin{aligned}
A \parallel_{iw} (A \parallel_{iw} B) &= \langle c(p, q, m), K \rangle \parallel_{iw} (\langle c(p, q, m), K \rangle \parallel_{iw} \langle c(p, q, n), K \rangle) \\
&= \langle c(p, q, m), K \rangle \parallel_{iw} \langle c(p, q, m) \parallel_{iw}^K c(p, q, n), K \rangle \\
&= \langle c(p, q, m), K \rangle \parallel_{iw} \langle \delta, K \rangle \\
&= \langle c(p, q, m) \parallel_{iw}^K \delta, K \rangle \\
&= \langle \delta, K \rangle
\end{aligned}$$

These two computations illustrate the difference fairly well. In the computation of $A \parallel_{iw} B$ in the semantics of [MvWW93] we lost information on the entities which are present, whereas in the second computation we did not. Observe that in the definition of interworking merge as presented in this paper $(A \parallel_{iw} A) \parallel_{iw} B$ and $A \parallel_{iw} (A \parallel_{iw} B)$ are entity bisimilar.

Proposition 3.2 (Commutativity of α_w and \parallel_{iw}) For closed $IW_\varepsilon(A, EID, E)$ terms x and y and sets of entities E, E_1 and E_2 we have

$$E(x) \cap E(y) = \emptyset \Rightarrow x \alpha_w y = y \alpha_w x \quad (1)$$

$$E(x) \cap E(y) = \emptyset \Rightarrow \langle x, E_1 \rangle \alpha_w \langle y, E_2 \rangle = \langle y, E_2 \rangle \alpha_w \langle x, E_1 \rangle \quad (2)$$

$$x \parallel_{iw}^E y = y \parallel_{iw}^E x \quad (3)$$

$$\langle x, E_1 \rangle \parallel_{iw}^E \langle y, E_2 \rangle = \langle y, E_2 \rangle \parallel_{iw}^E \langle x, E_1 \rangle \quad (4)$$

$$\langle x, E_1 \rangle \parallel_{iw} \langle y, E_2 \rangle = \langle y, E_2 \rangle \parallel_{iw} \langle x, E_1 \rangle \quad (5)$$

Proof For proofs of the propositions (1) and (3) we refer to [Ren93]. Proposition (2) follows immediately from proposition (1). Proposition (4) follows immediately from proposition (3). Proposition (5) can be proven as follows. Denote $E_1 \cup E(x)$ and $E_2 \cup E(y)$ by E'_1 and E'_2 respectively.

$$\begin{aligned}
\langle x, E_1 \rangle \parallel_{iw} \langle y, E_2 \rangle &= \langle x, E'_1 \rangle \parallel_{iw} \langle y, E'_2 \rangle \\
&= \langle x \parallel_{iw}^{E'_1 \cap E'_2} y, E'_1 \cup E'_2 \rangle \\
&= \langle x, E'_1 \rangle \parallel_{iw}^{E'_1 \cap E'_2} \langle y, E'_2 \rangle \\
&= \langle y, E'_2 \rangle \parallel_{iw}^{E'_1 \cap E'_2} \langle x, E'_1 \rangle \\
&= \langle y \parallel_{iw}^{E'_1 \cap E'_2} x, E'_1 \cup E'_2 \rangle \\
&= \langle y, E'_2 \rangle \parallel_{iw} \langle x, E'_1 \rangle \\
&= \langle y, E_2 \rangle \parallel_{iw} \langle x, E_1 \rangle
\end{aligned}$$

Proposition 3.3 (Associativity of \circ_{iw} and $\|_{iw}$) For closed $IW_\varepsilon(A, EID, E)$ terms x , y , and z and $E_1, E_2, E_3 \subseteq EID$ we have

$$(x \circ_{iw} y) \circ_{iw} z = x \circ_{iw} (y \circ_{iw} z) \quad (1)$$

$$(\langle x, E_1 \rangle \circ_{iw} \langle y, E_2 \rangle) \circ_{iw} \langle z, E_3 \rangle = \langle x, E_1 \rangle \circ_{iw} (\langle y, E_2 \rangle \circ_{iw} \langle z, E_3 \rangle) \quad (2)$$

$$(x \|_{iw}^{E_1 \cap E_2} y) \|_{iw}^{(E_1 \cup E_2) \cap E_3} z = x \|_{iw}^{E_1 \cap (E_2 \cup E_3)} (y \|_{iw}^{E_2 \cap E_3} z) \quad (3)$$

$$(\langle x, E_1 \rangle \|_{iw} \langle y, E_2 \rangle) \|_{iw} \langle z, E_3 \rangle = \langle x, E_1 \rangle \|_{iw} (\langle y, E_2 \rangle \|_{iw} \langle z, E_3 \rangle) \quad (4)$$

Proof For a proof of proposition (1) we refer to [Ren93]. Proposition (2) follows easily from proposition (1):

$$\begin{aligned} (\langle x, E_1 \rangle \circ_{iw} \langle y, E_2 \rangle) \circ_{iw} \langle z, E_3 \rangle &= \langle x \circ_{iw} y, E_1 \cup E_2 \rangle \circ_{iw} \langle z, E_3 \rangle \\ &= (\langle x \circ_{iw} y \rangle \circ_{iw} z, E_1 \cup E_2 \cup E_3) \\ &= \langle x \circ_{iw} (y \circ_{iw} z), E_1 \cup E_2 \cup E_3 \rangle \\ &= \langle x, E_1 \rangle \circ_{iw} \langle y \circ_{iw} z, E_2 \cup E_3 \rangle \\ &= \langle x, E_1 \rangle \circ_{iw} (\langle y, E_2 \rangle \circ_{iw} \langle z, E_3 \rangle) \end{aligned}$$

Proposition (3) is proven with simultaneous induction on the total number of symbols in x , y and z of the following propositions. We have used the following abbreviations: $S = E_1 \cap E_2 \cap E_3$, $A = (E_1 \cap E_2) - S$, $B = (E_2 \cap E_3) - S$, and $C = (E_1 \cap E_3) - S$.

$$(x \|_{iw}^{A \cup S} y) \|_{iw}^{B \cup C \cup S} z = x \|_{iw}^{A \cup C \cup S} (y \|_{iw}^{B \cup S} z) \quad (5)$$

$$(x \|_{iw}^{A \cup S} y) \|_{iw}^{B \cup C \cup S} z = x \|_{iw}^{A \cup C \cup S} (y \|_{iw}^{B \cup S} z) \quad (6)$$

$$(x \|_{iw}^{A \cup S} y) \|_{iw}^{B \cup C \cup S} z = x \|_{iw}^{A \cup C \cup S} (y \|_{iw}^{B \cup S} z) \quad (7)$$

$$\sqrt{(x \|_{iw}^{A \cup C} y) \cdot \sqrt{z}} = \sqrt{(x) \cdot \sqrt{(y \|_{iw}^{B \cup S} z)}} \quad (8)$$

$$(x \|_{iw}^{A \cup S} y) \|_{iw}^{B \cup C \cup S} z = x \|_{iw}^{A \cup C \cup S} (y \|_{iw}^{B \cup S} z) \quad (9)$$

We will not prove these propositions. Proposition (4) follows from proposition (3) as follows. We use the following abbreviations: $F_1 = E_1 \cup E(x)$, $F_2 = E_2 \cup E(y)$, and $F_3 = E_3 \cup E(z)$.

$$\begin{aligned} (\langle x, E_1 \rangle \|_{iw} \langle y, E_2 \rangle) \|_{iw} \langle z, E_3 \rangle &= (\langle x, F_1 \rangle \|_{iw} \langle y, F_2 \rangle) \|_{iw} \langle z, F_3 \rangle \\ &= \langle x \|_{iw}^{F_1 \cap F_2} y, F_1 \cup F_2 \rangle \|_{iw} \langle z, F_3 \rangle \\ &= (\langle x \|_{iw}^{F_1 \cap F_2} y \rangle \|_{iw}^{(F_1 \cup F_2) \cap F_3} z, F_1 \cup F_2 \cup F_3) \\ &= \langle x \|_{iw}^{F_1 \cap (F_2 \cup F_3)} y \rangle \|_{iw}^{F_2 \cap F_3} z, F_1 \cup F_2 \cup F_3 \\ &= \langle x, F_1 \rangle \|_{iw} \langle y \|_{iw}^{F_2 \cap F_3} z, F_2 \cup F_3 \rangle \\ &= \langle x, F_1 \rangle \|_{iw} (\langle y, F_2 \rangle \|_{iw} \langle z, F_3 \rangle) \\ &= \langle x, E_1 \rangle \|_{iw} (\langle y, E_2 \rangle \|_{iw} \langle z, E_3 \rangle) \end{aligned}$$

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.

Let $f : EID \rightarrow EID$ be a partial mapping from entities to entities. An Interworking x can be an f -refinement of another Interworking y , denoted by $x \sqsubseteq_f y$. This is the case if, after renaming of a set of entities of x into one entity of y (according to f) and after removal of all internal actions on the refined entity within the refined interworking and the removal of all internal actions on the

refining entities within the refining interworking, the behaviour of both Interworkings is equal. The mapping f is partial in order to distinguish between an entity p which is not refined at all ($p \notin \text{dom}(f)$) and an entity p which is refined by (amongst others) an entity p ($f(p) = p$). For an example of interworking refinement see Figure 4. The entities $q1$ and $q2$ refine the entity q .

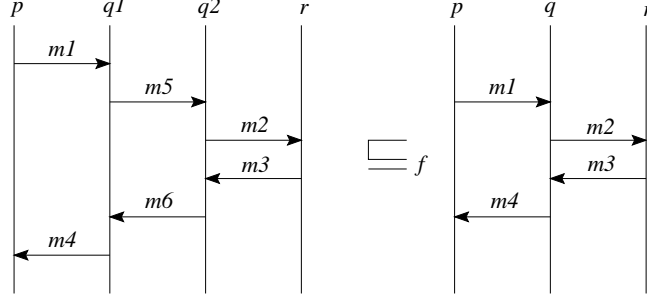


Figure 4: Interworking Refinement

The intuition is that the external behaviour of a single entity within the Interworking y can be refined into, or implemented by, a collective behaviour of a number of entities within the Interworking x . So, the emphasis is on inter-entity communication and not so much on intra-entity behaviour. Besides the singular f -entity refinement discussed above, it is also allowed to consider a number of refinements at the same time: *multiple refinement*. An example of multiple refinement is given in Figure 5. The entity p is refined by the entities p_1 and p_2 , and the entity q is refined by the entities q_1 and q_2 .

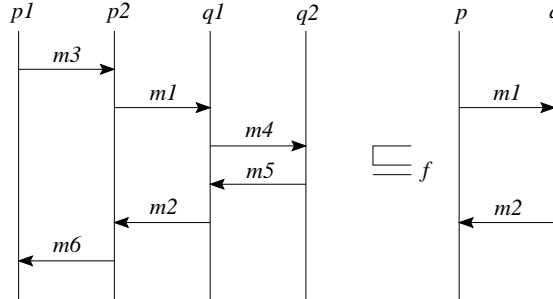


Figure 5: Multiple Interworking Refinement

The mapping from refining entities to refined entities is provided by the partial function f . Before we define this refinement formally 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 be a set and let $f : F \rightarrow F$ be a partial function. The total function associated with f , notation f^* , is for all $x \in F$, defined by

$$f^*(x) = \begin{cases} f(x) & \text{if } x \in \text{dom}(f) \\ x & \text{if } x \notin \text{dom}(f) \end{cases}$$

Let $f : EID \rightarrow EID$ be a partial function, then the renaming operator ρ_f related to f is defined by the axioms in Table 11. This renaming operator resembles the renaming operator ρ_f from [BB88].

$\rho_f(\varepsilon)$	=	ε
$\rho_f(\delta)$	=	δ
$\rho_f(c(p, q, m))$	=	$c(f^*(p), f^*(q), m)$
$\rho_f(a \cdot x)$	=	$\rho_f(a) \cdot \rho_f(x)$
$\rho_f(x + y)$	=	$\rho_f(x) + \rho_f(y)$

Table 11: Entity Renaming function on processes

In Table 12 the entity renaming operator on processes is extended to labelled processes. Note that also the entity component of a labelled process is renamed with respect to the mapping f .

$\rho_f(\langle x, E \rangle)$	=	$\langle \rho_f(x), \{f^*(e) \mid e \in E\} \rangle$
--------------------------------	---	--

Table 12: Entity Renaming function on labelled processes

In order to remove all internal actions on entities we are not interested in, we substitute the empty process ε for them. The entities for which we remove the internal actions are given by $\text{rng}(f)$. We define the set of all internal actions on the entities of a set E as follows:

$$\text{Int}(E) = \{a \in A \mid E(a) \subseteq E \wedge |E(a)| = 1\}$$

Let I be a set of atomic actions, in Table 13 we define the operator ε_I that renames atomic actions from I into ε . This operator is taken from [Vra91].

$\varepsilon_I(\varepsilon)$	=	ε
$\varepsilon_I(\delta)$	=	δ
$\varepsilon_I(a \cdot x)$	=	$\varepsilon_I(x)$ if $a \in I$
$\varepsilon_I(a \cdot x)$	=	$a \cdot \varepsilon_I(x)$ if $a \notin I$
$\varepsilon_I(x + y)$	=	$\varepsilon_I(x) + \varepsilon_I(y)$

Table 13: Renaming atomic actions into ε

In Table 14 the operator for removing internal communications is extended to labelled processes. By renaming actions into ε it can be the case that entity information is removed completely from the process expression. Therefore, we first make sure that all entity information is contained in the entity component of the labelled process.

Let $f : EID \rightarrow EID$ be a refinement mapping, the f -refinement relation on labelled processes is then defined by the equation in Table 15.

Next, we extend this notion of refinement with a fixed mapping to a notion of refinement which abstracts from this mapping. This notion of refinement is called *entity refinement*. Interworking x is an entity refinement of Interworking y , notation $x \sqsubseteq_f y$ if and only if there exists a refinement mapping f such that $x \sqsubseteq_f y$.

Example 4.1 As an illustration of this algebraic definition of refinement the refinement relation between the Interworkings in Figure 5 is computed. The left-hand side Interworking will be called

$$\varepsilon_I(\langle x, E \rangle) = \langle \varepsilon_I(x), E \cup E(x) \rangle$$

Table 14: Removing internal communications from labelled processes

$$x \sqsubseteq_f y \quad \text{iff} \quad \varepsilon_{Int(rng(f))}(y) = \varepsilon_{Int(rng(f))}(\rho_f(x))$$

Table 15: f -Refinement

A and the right-hand side Interworking B . Semantically these Interworkings are represented by

$$\begin{aligned} A &= \langle c(p1, p2, m3) \alpha_w c(p2, q1, m1) \alpha_w c(q1, q2, m4) \alpha_w c(q2, q1, m5) \\ &\quad \alpha_w c(q1, p2, m2) \alpha_w c(p2, p1, m6), \emptyset \rangle \\ B &= \langle c(p, q, m1) \alpha_w c(q, p, m2), \emptyset \rangle \end{aligned}$$

Elimination of the α_w yields the following equations

$$\begin{aligned} A &= \langle c(p1, p2, m3) \cdot c(p2, q1, m1) \cdot c(q1, q2, m4) \cdot c(q2, q1, m5) \\ &\quad \cdot c(q1, p2, m2) \cdot c(p2, p1, m6), \{p1, p2, q1, q2\} \rangle \\ B &= \langle c(p, q, m1) \cdot c(q, p, m2), \{p, q\} \rangle \end{aligned}$$

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 A according to f .

$$\rho_f(A) = \langle c(p, p, m3) \cdot c(p, q, m1) \cdot c(q, q, m4) \cdot c(q, q, m5) \cdot c(q, p, m2) \cdot c(p, p, m6), \{p, q\} \rangle$$

The set of actions which should be removed is given by

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

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

$$\begin{aligned} \varepsilon_{Int(rng(f))}(\rho_f(A)) &= \langle c(p, q, m1) \cdot c(q, p, m2), \{p, q\} \rangle \\ \varepsilon_{Int(rng(f))}(B) &= \langle c(p, q, m1) \cdot c(q, p, m2), \{p, q\} \rangle \end{aligned}$$

We can conclude that Interworking A is an f -refinement of Interworking B .

For the entity refinement relation we have the following properties.

Proposition 4.2 (Reflexivity) *For all closed labelled processes x we have $x \sqsubseteq x$.*

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

Proposition 4.3 (Transitivity) *For all closed labelled processes x, y , and z we have*

$$x \sqsubseteq y \quad \text{and} \quad y \sqsubseteq z \quad \text{implies} \quad x \sqsubseteq z$$

$$x \sqsubseteq y \quad \text{iff} \quad \exists f : EID \rightarrow EID \quad x \sqsubseteq_f y$$

Table 16: Entity refinement

Proof Let F be some set and let $f : F \rightarrow F$ 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 as follows

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

Suppose that there exist $f, g : EID \rightarrow EID$ such that $x \sqsubseteq_f y$ and $y \sqsubseteq_g z$. It is our claim that $x \sqsubseteq_{g^{rang(f)} \circ f^{dom(g)}} z$. 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 $x = c(p, p, m)$ and $y = c(p, p, n)$. Then we have $x \sqsubseteq y$ and $y \sqsubseteq x$, but we do not have $x = y$. 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 for Interworkings in which we solved the problems encountered in a former semantics and which allows a definition of refinement. An empty interworking is simply represented by the empty process. The anomaly with respect to empty entities in the context of an interworking merge has been solved by attributing the process expressions with the set of entities involved. The definition of refinement presented here is a little easier than a former definition. We have derived several properties and obtained sound and complete theories.

It is expected that this new semantics has only limited impact on the already existing tools for Interworkings. Furthermore, due to the formal definition also an implementation of refinement seems plausible.

References

- [BB88] J.C.M. Baeten and J.A. Bergstra. Global Renaming Operators in Concrete Process Algebra. *Information and Computation*, 78, 1988.
- [BK84a] J.A. Bergstra and J.W. Klop. Process Algebra for Synchronous Communication. *Information & Control*, 60:109–137, 1984.
- [BK84b] J.A. Bergstra and J.W. Klop. The Algebra of Recursively Defined Processes and the Algebra of Regular Processes. In J. Paredaens, editor, *Proceedings 11th ICALP*, volume 172 of *Lecture Notes in Computer Science*, pages 82–95. Springer-Verlag, 1984. Extended abstract, full version appeared in [PVvV95].
- [BV93] J.C.M. Baeten and C. Verhoef. A Congruence Theorem for Structured Operational Semantics with Predicates. In E. Best, editor, *CONCUR'93*, volume 715 of *Lecture Notes in Computer Science*, pages 477–492. Springer-Verlag, 1993.
- [BV95] 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*, 1995. To appear.
- [BW90] J.C.M. Baeten and W.P. Weijland. *Process Algebra*. Cambridge Tracts in Theoretical Computer Science 18. Cambridge University Press, Cambridge, 1990. ISBN 0-521-40043-0.
- [IT94] ITU-T. Z.120 Messages sequence chart (MSC), 1994.

- [KL80] S. Kamin and J.-J. Lévy. Two Generalizations of the Recursive Path Ordering. Unpublished manuscript, 1980.
- [Klo92] J.W. Klop. Term Rewriting Systems. In S. Abramsky, D.M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science, volume II*, pages 1–116. Oxford University Press, 1992.
- [MvWW92] S. Mauw, M. van Wijk, and T. Winter. Syntax and Semantics of Synchronous Interworkings VI, Refinement. Unpublished manuscript, 1992.
- [MvWW93] 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*, Proceedings of the Sixth SDL Forum, pages 167–178, Darmstadt, 1993. Elsevier Science Publishers, Amsterdam. ISBN 0-444-81486-8.
- [MW93] S. Mauw and T. Winter. A Prototype Toolset for Interworkings. *Philips Telecommunication Review*, 51(3), 1993.
- [Plo81] G.D. Plotkin. A Structural Approach to Operational Semantics. Technical Report DIAMI FN-19, Computer Science Department, Aarhus University, 1981.
- [Plo83] G.D. Plotkin. An Operational Semantics for CSP. In *Proceedings of the Conference on the Formal Description of Programming Concepts*, volume 2, Garmisch, 1983.
- [PVvV95] A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors. *Algebra of Communicating Processes, Utrecht 1994*, Workshops in Computing. Springer-Verlag, 1995. ISBN 0-387-19909-8.
- [Ren93] M.A. Reniers. Verificatie van Enige Eigenschappen van Interworkings. Unpublished manuscript in Dutch, 1993.
- [RW94] 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.
- [vdBG95] J. van den Brink and W.O.D. Griffioen. Formal Semantics of Interworkings with Discrete Absolute Time. In A. Ponse, C. Verhoef, and S.F.M. van Vlijmen, editors, *Algebra of Communicating Processes, Utrecht 1994*, Workshops in Computing, pages 106–123. Springer-Verlag, 1995. See [PVvV95].
- [Ver94] C. Verhoef. A General Conservative Extension Theorem in Process Algebra. In E.-R. Olderog, editor, *Programming Concepts, Methods and Calculi (PROCMET '94)*, volume 56 of *IFIP Transactions A: Computer Science and Technology*, pages 149–168. North-Holland, 1994.
- [Vra91] J.L.M. Vrancken. *Studies in Process Algebra, Algebraic Specifications and Parallelism*. PhD thesis, University of Amsterdam, 1991.