Consistency of Partial Process Specifications

Maarten Steen, John Derrick, Eerke Boiten, Howard Bowman

Computing Laboratory, University of Kent at Canterbury
Canterbury, Kent CT2 7NF, UK. M.W.A.Steen@ukc.ac.uk

Abstract. The structuring of the specification and development of distributed systems according to *viewpoints*, as advocated by the Reference Model for Open Distributed Processing, raises the question of when such viewpoint specifications may be considered *consistent* with one another. In this paper, we analyse the notion of consistency in the context of formal process specification. It turns out that different notions of correctness give rise to different consistency relations. Each notion of consistency is formally characterised and placed in a spectrum of consistency relations. An example illustrates the use of these relations for consistency checking.

1 Introduction

There is a growing awareness in distributed software engineering that the development of complex distributed systems can no longer be seen as a linear, top-down activity. It is now widely advocated to structure the specification and development of such systems according to, so called, *viewpoints*. Prominent examples of viewpoint oriented development models are the Reference Model for Open Distributed Processing (RM-ODP) [9], the Viewpoint Oriented Software Engineering (VOSE) framework [5], and object oriented analysis and design models, such as [2].

In contrast with the traditional 'waterfall' model of development, where an initial, abstract specification is stepwise refined to a final, concrete specification, viewpoint models allow specifiers to split up the complete specification of a complex system into a number of viewpoint specifications each concentrating on a particular concern or aspect of the system. Individual viewpoint specifications can then be developed further relatively independent of one another. The RM-ODP, for example, defines five viewpoints — enterprise, information, computational, engineering, and technology — from which distributed systems may be described.

One of the main problems in any multiple viewpoint approach to specification is defining and establishing that the various viewpoint specifications are *consistent* with one another. This problem becomes particularly challenging when we consider that different specification techniques may be applicable to different viewpoints. The ODP information viewpoint, for example, can be expressed quite naturally in Z, whereas LOTOS is considered more suitable for the computational viewpoint [16].

In some viewpoint models consistency is defined as a simple set of syntactic constraints. The Booch method [2] (supported by the Rational Rose[1] tool) for object oriented design, for example, requires that there is a corresponding operation in a Class Diagram for each message in a Sequence Diagram. Here, however, we are concerned with *behavioural*, or *semantic*, consistency.

In this paper, we analyse the consistency problem for a substantial number of process algebraic specification techniques. Process algebra provides a rich theory for the specification of behaviour. Therefore, this work should provide the formal foundations for consistency checking techniques for more 'user-friendly' behavioural specification notations, such as State Charts and Sequence Diagrams. In fact, the consistency relations identified in this paper are directly applicable to all specification formalisms of which the semantics can be expressed using labelled transition systems, traces, refusals or failures, e.g., CSP [8], CCS [15], and Object-Z [6].

2 Process Specification

We introduce a simple process algebraic language similar to CCS and CSP for the description of process behaviour. The syntax is borrowed from LOTOS [1]:

$$\mathcal{P} ::= \textsf{\textbf{stop}} \mid \alpha;\mathcal{P} \mid \mathcal{P} \,[]\, \mathcal{P} \mid \mathcal{P} \,|[\mathsf{A}]|\, \mathcal{P} \mid \textsf{\textbf{hide}}\ A\ \textsf{\textbf{in}}\ \mathcal{P} \mid X$$

Here it is assumed that a set of action labels $\mathbf{L}$ is given. Then, $\alpha \in \mathbf{L} \cup \{\tau\}$; $\tau \notin \mathbf{L}$ is the unobservable, or internal, action; $A \subseteq \mathbf{L}$; and X is a process name. We will assume that a definition exists for each process name used. Process definitions are written $X := p$, where p is a behaviour expression that can again contain process names, including possibly X itself, thus making the definition recursive.

Semantically, process behaviour can be modelled in many different ways. In the following, we consider labelled transition systems, traces, refusals and some combinations of the latter two.

2.1 Labelled Transition Systems

Definition 1. *A* labelled transition system *is a structure* $(S, L, \longrightarrow, s_0)$*, where* S *is a set of states,* L *is a set of action labels,* $\longrightarrow \subseteq S \times (L \cup \{\tau\}) \times S$ *is a transition relation, and* $s_0 \in S$ *is the initial state.*

Each behaviour description is associated, in the usual manner, with a labelled transition system through the axioms and inference rules given in Table 1.

Often labelled transition systems are considered to be too concrete to abstractly specify system behaviour. It is therefore customary to interpret process specifications via, so called, *implementation relations* [13, 3]. These are relations between a domain of implementations and a domain of specifications that formalise a particular notion of correctness. They may, for example, abstract from

[1] Rational Rose is a trade mark of the Rational Software Corporation.

Table 1. Inference rules

	$\vdash \alpha; p \xrightarrow{\alpha} p$
$p \xrightarrow{\alpha} p'$	$\vdash p \,[]\, q \xrightarrow{\alpha} p'$
$q \xrightarrow{\alpha} q'$	$\vdash p \,[]\, q \xrightarrow{\alpha} q'$
$p \xrightarrow{\alpha} p', \alpha \notin A$	$\vdash p \,\vert[\mathsf{A}]\vert\, q \xrightarrow{\alpha} p' \,\vert[\mathsf{A}]\vert\, q$
$q \xrightarrow{\alpha} q', \alpha \notin A$	$\vdash p \,\vert[\mathsf{A}]\vert\, q \xrightarrow{\alpha} p \,\vert[\mathsf{A}]\vert\, q'$
$p \xrightarrow{\alpha} p', q \xrightarrow{\alpha} q', \alpha \in A$	$\vdash p \,\vert[\mathsf{A}]\vert\, q \xrightarrow{\alpha} p' \,\vert[\mathsf{A}]\vert\, q'$
$p \xrightarrow{\alpha} p', \alpha \notin A$	$\vdash \textsf{\textbf{hide}}\ A\ \textsf{\textbf{in}}\ p \xrightarrow{\alpha} \textsf{\textbf{hide}}\ A\ \textsf{\textbf{in}}\ p'$
$p \xrightarrow{\alpha} p', \alpha \in A$	$\vdash \textsf{\textbf{hide}}\ A\ \textsf{\textbf{in}}\ p \xrightarrow{\alpha} \textsf{\textbf{hide}}\ A\ \textsf{\textbf{in}}\ p'$
$p \xrightarrow{\alpha} p', X := p$	$\vdash X \xrightarrow{\alpha} p'$

the internal behaviour of an implementation and only verify whether the externally observable behaviour corresponds to the behaviour described in the specification.

2.2 Traces and Refusals

Let $\mathbf{L}^*$ denote the set of all strings over the set of observable actions $\mathbf{L}$. Elements of $\mathbf{L}^*$ are also called *traces*. The empty string, or empty trace, is denoted ϵ and σ is used to range over $\mathbf{L}^*$. Concatenation of traces is represented by juxtaposition.

In Table 2 the notion of transition is generalised to traces. We further define $Tr(p)$, the set of traces of a process p, $Out(p,\sigma)$, the set of possible actions after the trace σ, and $Ref(p,\sigma)$, the sets of actions refused by a process p after the trace σ:

Definition 2.

$$\begin{aligned}
Tr(p) &\overset{def}{=} \{\sigma \in \mathbf{L}^* \mid p \overset{\sigma}{\Longrightarrow}\} \\
Out(p,\sigma) &\overset{def}{=} \{a \in \mathbf{L} \mid \exists p' \bullet p \overset{\sigma}{\Longrightarrow} p' \ \ and\ \ p' \overset{a}{\Longrightarrow}\} \\
Ref(p,\sigma) &\overset{def}{=} \{X \subseteq \mathbf{L} \mid \exists p' \bullet p \overset{\sigma}{\Longrightarrow} p' \ \ and\ \forall a \in X \bullet p' \overset{a}{\not\Longrightarrow}\}
\end{aligned}$$

Table 2. Trace relations

Notation	Meaning
$\overset{\epsilon}{\Longrightarrow}$	$(\xrightarrow{\tau})^*$, i.e., the reflexive and transitive closure of $\xrightarrow{\tau}$
$p \overset{a\sigma}{\Longrightarrow} p'$	$\exists q, q' \bullet p \overset{\epsilon}{\Longrightarrow} q \xrightarrow{a} q' \overset{\sigma}{\Longrightarrow} p'$
$p \overset{\sigma}{\Longrightarrow}$	$\exists p' \bullet p \overset{\sigma}{\Longrightarrow} p'$
$p \overset{\sigma}{\not\Longrightarrow}$	$\nexists p' \bullet p \overset{\sigma}{\Longrightarrow} p'$

2.3 Implementation Relations

A large number of implementation relations has been defined over labelled transition systems [7]; each one capturing a different notion of correctness. In this paper, we consider only the most prominent trace and/or refusal based implementation relations from process algebra. Our selection is largely based on a pioneering study on implementation relations by Brinksma et al. [3].

Definition 3. *Let $p, s \in \mathcal{P}$ be processes, then we define the following relations:*

name	*denotation*	*definition*
trace refinement	$p \leq_{\mathrm{tr}} s$	$Tr(p) \subseteq Tr(s)$
trace equivalence	$p \approx_{\mathrm{tr}} s$	$Tr(p) = Tr(s)$
conformance	$p\,\mathbf{conf}\,s$	$\forall \sigma \in Tr(s) \bullet Ref(p, \sigma) \subseteq Ref(s, \sigma)$
reduction	$p\,\mathbf{red}\,s$	$p \leq_{\mathrm{tr}} s$ *and* $p\,\mathbf{conf}\,s$
extension	$p\,\mathbf{ext}\,s$	$s \leq_{\mathrm{tr}} p$ *and* $p\,\mathbf{conf}\,s$
testing equivalence	$p \approx_{\mathrm{te}} s$	$p\,\mathbf{red}\,s$ *and* $s\,\mathbf{red}\,p$

Perhaps the simplest implementation relation is *trace refinement.* It only verifies that the implementation cannot perform sequences of observable actions (traces) that are not allowed by the specification. This is useful for capturing, so called, *safety properties.* However, we cannot use it to specify that anything *must* happen. *Trace equivalence* is slightly stronger in that it requires that the implementation and specification have the same possible traces. Another notion of validity is captured by the *conformance* relation (**conf**), derived from testing theory. It requires for each trace of the specification, that the implementation can only refuse to do whatever the specification refuses after that trace. The *reduction* relation (**red**), sometimes referred to as testing preorder or failure preorder, is the intersection of trace refinement and conformance. It gives rise to a specification technique with which one can specify both that certain actions must happen and that certain traces are not allowed. The *extension* relation, on the other hand, allows that more traces are added in the implementation, as long as the implementation is still conformant to its specification. The strongest implementation relation considered here is *testing equivalence.* It requires that the observable behaviour of implementation resp. specification cannot be distinguished through external testing.

Process specifications, and in fact any other trace/refusal based specifications, can be interpreted under any of the implementation relations defined above to yield a different specification formalism [10] for system behaviour. In a multiple viewpoint approach to specification potentially all these formalisms may be used simultaneously. Below, we show how different viewpoints may require different implementation relations to adequately capture their intended meaning.

2.4 Example Viewpoint Specifications

Consider the specification of a simple vending machine using the ODP viewpoints. (It is outside the scope of this paper to give definitions for the five ODP viewpoints. The interested reader is referred to [14] or the standard itself [9].)

From the **enterprise viewpoint** one might like to specify the following policies, divided in permissions and obligations:

Permissions The system is permitted to exhibit any of the following traces of behaviour: $\{\epsilon, \text{coin}, \text{coin.coffee}, \text{coin.tea}, \text{coin.coffee.coin}, \text{coin.tea.coin}, \ldots\}$. This could be captured by the following specification, when interpreted under the trace refinement relation ($\leq_{\text{tr}}$):

$\mathsf{Perm} := \mathsf{coin};\ (\mathsf{coffee};\ \mathsf{Perm}\ \ []\ \ \mathsf{tea};\ \mathsf{Perm})$

Obligations The system user is obliged to always first insert a coin into the machine. The following specification captures this. Here we have decided to interpret the specification under the extension relation (**ext**), so the specification does not prohibit any other behaviour.

$\mathsf{Obl} := \mathsf{coin};\ \mathbf{stop}$

From the **computational viewpoint** the system is viewed as a computational object providing a computational interface upon which its environment (the user) can invoke one of three operations: coin, coffee and tea.

$$\begin{aligned}\mathsf{Comp} := &\quad \tau;\ \mathsf{coin};\ (\tau;\ \mathsf{coffee};\ \mathsf{Comp}\ \ []\ \ \tau;\ \mathsf{tea};\ \mathsf{Comp})\\ &[]\ \ \tau;\ \mathsf{coffee};\ \mathsf{Comp}\\ &[]\ \ \tau;\ \mathsf{tea};\ \mathsf{Comp}\end{aligned}$$

If the coin operation is invoked, the system will respond by offering its environment either coffee or tea. In case one of the other two operations is invoked by the environment, the system will return to its initial state. Non-determinism is used to indicate that not all of these operations need to be present in an implementation. Therefore, any reduction (**red**) is considered a correct implementation.

From the **engineering viewpoint** the system might be viewed as being composed of two components, a money handler (MH) and a drinks dispenser (DD), that communicate via a channel. As the channel is only introduced for internal communication it is hidden from the environment. The following specification of the engineering viewpoint is interpreted under the testing equivalence relation ($\approx_{\text{te}}$).

$$\begin{aligned}&\mathsf{Eng} := \mathbf{hide}\ \mathsf{channel}\ \mathbf{in}\ \ \mathsf{MH}\ \ |[\mathsf{channel}]|\ \ \mathsf{DD}\\ &\mathsf{MH} := \mathsf{coin};\ \mathsf{channel};\ \mathsf{MH}\\ &\mathsf{DD} := \mathsf{channel};\ (\mathsf{coffee};\ \mathsf{DD}\ \ []\ \ \mathsf{tea};\ \mathsf{DD})\end{aligned}$$

The obvious question now is whether all these viewpoint specifications are consistent with one another.

3 Consistency

The purpose of this section is to define (necessary and sufficient) conditions for viewpoint specifications to be consistent. For the moment we will concentrate on *binary consistency*, i.e., consistency between two specifications. Informally, we call two specifications consistent if, and only if, they have at least one implementation in common, i.e., if there is an implementation that satisfies both specifications. The definition of consistency is thus parameterised on the notion of correctness that each specification is subjected to. As we have shown above, different viewpoint specifications may be subjected to interpretation under differing implementation relations. Therefore, each combination of implementation relations, $\mathbf{imp}_1, \mathbf{imp}_2$, gives rise to a different consistency relation, denoted $\mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}$.

Definition 4. *Let* $\mathbf{imp}_1, \mathbf{imp}_2$ *be implementation relations, then consistency between specifications subject to* $\mathbf{imp}_1$ *and specifications subject to* $\mathbf{imp}_2$ *is a binary relation* $\mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}$ *such that, for any* $s_1, s_2 \in \mathcal{P}$,

$$s_1\, \mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}\, s_2 \overset{def}{\Longleftrightarrow} \exists p \in \mathcal{P} \bullet p\, \mathbf{imp}_1\, s_1 \wedge p\, \mathbf{imp}_2\, s_2 .$$

Considering $\leq_{\mathrm{tr}}$, $\approx_{\mathrm{tr}}$, $\mathbf{conf}$, $\mathbf{red}$, $\mathbf{ext}$ and $\approx_{\mathrm{te}}$ as instantiations for $\mathbf{imp}_1$ and $\mathbf{imp}_2$ in the definition of binary consistency, we obtain 36 different notions of consistency. Whenever $\mathbf{imp}_1 = \mathbf{imp}_2$, we speak of *balanced consistency*, denoted $\mathcal{C}^2_{\mathbf{imp}}$. Section 3.1 deals with these (six) cases. The issue of *unbalanced consistency*, the remaining 30 cases, is discussed in section 3.2. Omitted proofs may be found in [17].

It is useful sometimes to use the following alternative characterisation of consistency as the composition of two implementation relations:

Proposition 5. *For any two implementation relations* $\mathbf{imp}_1, \mathbf{imp}_2$,

$$\mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2} = \mathbf{imp}_1^{-1} \circ \mathbf{imp}_2 .$$

3.1 Balanced Consistency

This section largely summarises results from [18], where we considered only the balanced consistency problem.

Since both specifications (in the binary case) are subject to the same implementation relation, binary, balanced consistency is a *symmetric* relation.

Proposition 6. *For any implementation relation* $\mathbf{imp}$, $\mathcal{C}^2_{\mathbf{imp}} = (\mathcal{C}^2_{\mathbf{imp}})^{-1}$.

We consider the six cases of binary, balanced consistency, denoted $\mathcal{C}^2_{\mathbf{imp}}$ for $\mathbf{imp} \in \{\leq_{\mathrm{tr}}, \approx_{\mathrm{tr}}, \mathbf{conf}, \mathbf{red}, \mathbf{ext}, \approx_{\mathrm{te}}\}$. For two of these, $\mathbf{imp}$ is instantiated with an equivalence relation. It is easily established that the consistency relation is equal to the implementation relation in those cases. Of the four remaining balanced consistency relations, three turn out to hold for any two specifications.

Theorem 7.

1. $\mathcal{C}^2_{\approx_{\mathrm{tr}}} = \approx_{\mathrm{tr}}$
2. $\mathcal{C}^2_{\approx_{\mathrm{te}}} = \approx_{\mathrm{te}}$

3. $\mathcal{C}^2_{\leq_{\mathrm{tr}}} = \mathcal{P} \times \mathcal{P}$
4. $\mathcal{C}^2_{\mathbf{conf}} = \mathcal{P} \times \mathcal{P}$
5. $\mathcal{C}^2_{\mathbf{ext}} = \mathcal{P} \times \mathcal{P}$

Proof. The first two results follow from the symmetry and transitivity of $\approx_{\mathrm{tr}}$ and $\approx_{\mathrm{te}}$. The remaining cases are proved by exibiting a bottom element in the respective refinement lattices. Such a bottom element is presented by a process $\perp$ such that $\forall s \bullet \perp \mathbf{imp}\, s$. The existence of such a bottom element implies consistency, since $s_1\, \mathcal{C}^2_{\mathbf{imp}}\, s_2 \Leftrightarrow \exists p \bullet p\, \mathbf{imp}\, s_1 \wedge p\, \mathbf{imp}\, s_2$.

3. $\forall s \bullet \mathsf{\mathbf{stop}} \leq_{\mathrm{tr}} s$, hence **stop** is the required bottom element.
4. Define a process Run, that can perform all possible traces and never refuses any action, as follows[2]:

   $\mathsf{Run} := \Sigma\{\mathsf{a};\, \mathsf{Run} \mid \mathsf{a} \in \mathbf{L}\}$

   Observe that, $\forall \sigma \in \mathbf{L}^* \bullet \mathit{Ref}(\mathsf{Run}, \sigma) = \{\emptyset\}$. Therefore, $\forall s \bullet \mathsf{Run}\, \mathbf{conf}\, s$.
5. The process Run, defined above, also has more traces than any other process, i.e. $\forall s \bullet \mathit{Tr}(\mathsf{Run}) = \mathbf{L}^* \supseteq \mathit{Tr}(s)$. Therefore, $\forall s \bullet \mathsf{Run}\, \mathbf{ext}\, s$. □

The following theorem gives a sufficient condition for two specifications (say s_1 and s_2) to be consistent with respect to reduction. The condition requires that s_1 and s_2 can at least refuse all the actions they may not both do after a certain trace.

Theorem 8. *Let $s_1, s_2 \in \mathcal{P}$ be two specifications, then $s_1\, \mathcal{C}^2_{\mathbf{red}}\, s_2$ if:*

$$\forall \sigma \in \mathit{Tr}(s_1) \cap \mathit{Tr}(s_2) \bullet \mathbf{L} \backslash (\mathit{Out}(s_1, \sigma) \cap \mathit{Out}(s_2, \sigma)) \in \mathit{Ref}(s_1, \sigma) \cap \mathit{Ref}(s_2, \sigma)$$

Proof. See [18].

3.2 Unbalanced Consistency

Unbalanced consistency is more complicated than the balanced case. First of all, there are many more cases of unbalanced consistency. Moreover, unlike balanced consistency relations, unbalanced ones are not symmetric. However, there is a close relationship between $\mathcal{C}_{\mathbf{imp}_1, \mathbf{imp}_2}$ and $\mathcal{C}_{\mathbf{imp}_2, \mathbf{imp}_1}$.

Proposition 9. *For any two implementation relations $\mathbf{imp}_1, \mathbf{imp}_2$,*

$$\mathcal{C}_{\mathbf{imp}_2, \mathbf{imp}_1} = \mathcal{C}^{-1}_{\mathbf{imp}_1, \mathbf{imp}_2}\,.$$

Since it is easy to derive the inverse of a relation (just swap the arguments), this proposition gives an easy recipe for deriving $\mathcal{C}_{\mathbf{imp}_2, \mathbf{imp}_1}$ from the relation with the implementation relations reversed $\mathcal{C}_{\mathbf{imp}_1, \mathbf{imp}_2}$. It halves our problem of finding 30 consistency conditions.

For the remaining 15 cases, observe that all implementation relations are *reflexive*. The following proposition therefore allows us to derive at least a sufficient condition for consistency to hold in each of these cases.

[2] The operator Σ generalises the choice operator (_ [] _).

Proposition 10. *Given a consistency relation* $\mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}$*, such that* $\mathbf{imp}_1$ *is reflexive,*

$$\mathbf{imp}_2 \subseteq \mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}\,.$$

Proof. From reflexivity of $\mathbf{imp}_1$, it follows that $\mathbf{Id} \subseteq \mathbf{imp}_1^{-1}$. And, by monotonicity of $\circ$, $\mathbf{Id} \subseteq \mathbf{imp}_1^{-1} \Rightarrow \mathbf{imp}_2 \subseteq \mathbf{imp}_1^{-1} \circ \mathbf{imp}_2 = \mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}$. □

Under the condition that the inverse of $\mathbf{imp}_1$ is stronger than $\mathbf{imp}_2$ and $\mathbf{imp}_2$ is a transitive relation, $\mathbf{imp}_2$ is both a necessary and sufficient condition. This result applies to six of the remaining cases.

Theorem 11. *Given a consistency relation* $\mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}$*, such that*

- $\mathbf{imp}_1$ *is reflexive,*
- $\mathbf{imp}_2$ *is transitive, and*
- $\mathbf{imp}_1^{-1} \subseteq \mathbf{imp}_2$*,*

then $\mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2} = \mathbf{imp}_2$.

Proof. By Prop. 10, we have $\mathbf{imp}_2 \subseteq \mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}$. In the other direction, we derive by monotonicity of $\circ$ and transitivity of $\mathbf{imp}_2$, that $\mathbf{imp}_1^{-1} \subseteq \mathbf{imp}_2 \Rightarrow \mathbf{imp}_1^{-1} \circ \mathbf{imp}_2 \subseteq \mathbf{imp}_2 \circ \mathbf{imp}_2 \subseteq \mathbf{imp}_2$. □

Corollary 12.

1. $\mathcal{C}_{\approx_{\mathrm{te}},\leq_{\mathrm{tr}}} = \leq_{\mathrm{tr}}$
2. $\mathcal{C}_{\approx_{\mathrm{te}},\approx_{\mathrm{tr}}} = \approx_{\mathrm{tr}}$
3. $\mathcal{C}_{\approx_{\mathrm{te}},\mathbf{red}} = \mathbf{red}$
4. $\mathcal{C}_{\approx_{\mathrm{te}},\mathbf{ext}} = \mathbf{ext}$
5. $\mathcal{C}_{\approx_{\mathrm{tr}},\leq_{\mathrm{tr}}} = \leq_{\mathrm{tr}}$
6. $\mathcal{C}_{\mathbf{ext},\leq_{\mathrm{tr}}} = \leq_{\mathrm{tr}}$

Since testing equivalence is stronger than all other implementation relations, and because it is an equivalence, we almost always have $\mathcal{C}_{\approx_{\mathrm{te}},\mathbf{imp}_2} = \mathbf{imp}_2$. The only case that is missing, is when $\mathbf{imp}_2 = \mathbf{conf}$. Even though $\mathbf{conf}$ is not transitive, we still have the same result.

Theorem 13. $\mathcal{C}_{\approx_{\mathrm{te}},\mathbf{conf}} = \mathbf{conf}$

Proof. By Prop. 10 we have $\mathbf{conf} \subseteq \mathcal{C}_{\approx_{\mathrm{te}},\mathbf{conf}}$. For inclusion in the other direction, observe that, by Prop. 5 and symmetry of $\approx_{\mathrm{te}}$, $\mathcal{C}_{\approx_{\mathrm{te}},\mathbf{conf}} = \approx_{\mathrm{te}} \circ \mathbf{conf}$. We now prove $\approx_{\mathrm{te}} \circ \mathbf{conf} \subseteq \mathbf{conf}$ by extensionality: $\forall s_1, s_2 \in \mathcal{P}$,

$$\begin{aligned}
&\quad s_1 \approx_{\mathrm{te}} \circ\, \mathbf{conf}\, s_2 \\
&\Leftrightarrow \exists p \bullet s_1 \approx_{\mathrm{te}} p \wedge p\, \mathbf{conf}\, s_2 \\
&\Leftrightarrow \exists p \bullet (\forall \sigma \in \mathbf{L}^* \bullet \mathit{Ref}(s_1,\sigma) = \mathit{Ref}(p,\sigma)) \\
&\quad \wedge (\forall \sigma \in \mathit{Tr}(s_2) \bullet \mathit{Ref}(p,\sigma) \subseteq \mathit{Ref}(s_2,\sigma)) \\
&\Rightarrow \forall \sigma \in \mathit{Tr}(s_2) \bullet \mathit{Ref}(s_1,\sigma) \subseteq \mathit{Ref}(s_2,\sigma) \\
&\Leftrightarrow s_1\, \mathbf{conf}\, s_2
\end{aligned}$$

□

Of the remaining consistency relations, one holds for any two specifications.

Theorem 14. $\mathcal{C}_{\mathbf{ext},\mathbf{conf}} = \mathcal{P} \times \mathcal{P}$

Proof. Use the same witness as in the proofs of $\mathcal{C}^2_{\mathbf{conf}} = \mathcal{C}^2_{\mathbf{ext}} = \mathcal{P} \times \mathcal{P}$. □

The remaining two consistency relations with **ext** coincide with trace refinement.

Theorem 15.

1. $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{ext}} = \geq_{\mathrm{tr}}$
2. $\mathcal{C}_{\mathbf{red},\mathbf{ext}} = \geq_{\mathrm{tr}}$

Proof. In one direction, inclusion follows by a simple monotonicity argument:

1. Since $\mathbf{ext} \subseteq \geq_{\mathrm{tr}}$, it follows that $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{ext}} = \approx_{\mathrm{tr}} \circ \mathbf{ext} \subseteq \approx_{\mathrm{tr}} \circ \geq_{\mathrm{tr}} = \geq_{\mathrm{tr}}$.
2. Since $\mathbf{red}^{-1} \subseteq \geq_{\mathrm{tr}}$ and $\mathbf{ext} \subseteq \geq_{\mathrm{tr}}$, it follows that $\mathcal{C}_{\mathbf{red},\mathbf{ext}} = \mathbf{red}^{-1} \circ \mathbf{ext} \subseteq \geq_{\mathrm{tr}} \circ \geq_{\mathrm{tr}} = \geq_{\mathrm{tr}}$.

In the other direction, we need to exhibit a common implementation for any two specifications s_1, s_2 such that $s_1 \geq_{\mathrm{tr}} s_2$. In both cases, such a common implementation is given by the deterministic process with the same traces as s_1. □

In an earlier version of this paper, we defined a relation $\mathbf{cons} \subseteq \mathcal{P} \times \mathcal{P}$ at this point (see definition 18) and proposed that being in this relation provided a sufficient and necessary condition for four of the remaining consistency relations, viz. $\mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}}$, $\mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{red}}$, $\mathcal{C}_{\mathbf{red},\mathbf{conf}}$, and $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}}$. However, we now know this not to be the case. Although **cons** is indeed a precise characterisation of $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}}$ (see theorem 19) and it plays a role in the characterisation of $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{red}}$ (see theorem 20), $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}}$ does not coincide with the other three aforementioned consistency relations. We can, however, establish a relative ordering between the four relations.

Proposition 16.

1. $\mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{red}} = \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}}$
2. $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}} \subset \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}}$
3. $\mathcal{C}_{\mathbf{red},\mathbf{conf}} \subset \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}}$

Proof.

1. Firstly, since $\mathbf{red} \subseteq \mathbf{conf}$, it follows that $\mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{red}} = \geq_{\mathrm{tr}} \circ \mathbf{red} \subseteq \geq_{\mathrm{tr}} \circ \mathbf{conf} = \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}}$. Secondly, suppose $\exists p \bullet p \leq_{\mathrm{tr}} s_1 \wedge p \,\mathbf{conf}\, s_2$, but $p \not\leq_{\mathrm{tr}} s_2$. There must then be a $\sigma \in \mathit{Tr}(p) \cap \mathit{Tr}(s_2)$ such that $a \in \mathit{Out}(p,\sigma) \setminus \mathit{Out}(s_2,\sigma)$ for some $a \in \mathbf{L}$. However, then $\{a\} \in \mathit{Ref}(s_2,\sigma)$ so we can remove the a-transition from p without invalidating that $p \leq_{\mathrm{tr}} s_1$ and $p \,\mathbf{conf}\, s_2$. Now, let p' be the process constructed from p by removing all these violating transition and we clearly have $p' \leq_{\mathrm{tr}} s_1$ and $p' \,\mathbf{red}\, s_2$.
2. Since $\approx_{\mathrm{tr}} \subseteq \geq_{\mathrm{tr}}$, it follows that $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}} = \approx_{\mathrm{tr}} \circ \mathbf{conf} \subseteq \geq_{\mathrm{tr}} \circ \mathbf{conf} = \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}}$. Moreover, there exist specifications s_1, s_2 such that $s_1 \,\mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}}\, s_2$, but $\neg(s_1 \,\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}}\, s_2)$ (see example 17).

3. Since $\mathbf{red}^{-1} \subseteq \geq_{\mathrm{tr}}$, it follows that $\mathcal{C}_{\mathbf{red},\mathbf{conf}} = \mathbf{red}^{-1} \circ \mathbf{conf} \subseteq \geq_{\mathrm{tr}} \circ \mathbf{conf} = \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}}$. Moreover, there exist specifications s_1, s_2 such that $s_1 \, \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}} \, s_2$, but $\neg(s_1 \, \mathcal{C}_{\mathbf{red},\mathbf{conf}} \, s_2)$ (see example 17). $\square$

Example 17. Consider the following specifications:

$s_1 :=$ a; **stop** [] b; **stop**
$s_2 := \tau$; a; **stop** [] b; c; **stop**

then we have $s_1 \, \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}} \, s_2$, because a; **stop** is a common implementation, but not $s_1 \, \mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}} \, s_2$ and not $s_1 \, \mathcal{C}_{\mathbf{red},\mathbf{conf}} \, s_2$. In the latter two cases, any common implementation would have to perform b initially and then refuse c to be an implementation of s_1, but such a process can never be conformant to s_2, which requires c after b. $\square$

Definition 18. *Define a relation* **cons** $\subseteq \mathcal{P} \times \mathcal{P}$ *as follows:*

$$p \, \mathbf{cons} \, q \overset{def}{\iff} \forall \sigma \in \mathit{Tr}(p) \cap \mathit{Tr}(q) \bullet (\mathbf{L} \backslash \mathit{Out}(p, \sigma)) \in \mathit{Ref}(q, \sigma).$$

The relation **cons** characterises $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}}$, as is shown in the following theorem. In order for a process p to be '*trace-conf consistent*' with a process q, q must be able to refuse everything that p cannot do after a certain trace σ common to both p and q.

Theorem 19. $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{conf}} = \mathbf{cons}$

Proof. Firstly from left to right. Assuming that $\exists p \bullet p \approx_{\mathrm{tr}} s_1 \;\; \wedge \;\; p \, \mathbf{conf} \, s_2$ we need to show that $s_1 \, \mathbf{cons} \, s_2$. Suppose not. By definition of **cons** this means that $\mathbf{L} \backslash \mathit{Out}(s_1, \sigma) \notin \mathit{Ref}(s_2, \sigma)$ for some trace $\sigma \in \mathit{Tr}(s_1) \cap \mathit{Tr}(s_2)$. From the assumption that $p \approx_{\mathrm{tr}} s_1$ it follows that $\mathit{Out}(p, \sigma) = \mathit{Out}(s_1, \sigma)$ and therefore that $\mathbf{L} \backslash \mathit{Out}(p, \sigma) \notin \mathit{Ref}(s_2, \sigma)$. However, for p to be a valid process (e.g., see [12, p. 62]), we must have $\mathbf{L} \backslash \mathit{Out}(p, \sigma) \in \mathit{Ref}(p, \sigma)$, which contradicts that $p \, \mathbf{conf} \, s_2$.

Secondly, from right to left. Assume $s_1 \, \mathbf{cons} \, s_2$. Next, construct a process p with the following traces and refusals:

$$\begin{aligned}
&\mathit{Tr}(p) = \mathit{Tr}(s_1) \\
&\mathit{Ref}(p, \sigma) = \mathit{Ref}(s_2, \sigma), \text{ if } \sigma \in \mathit{Tr}(s_2) \\
&\mathit{Ref}(p, \sigma) = \wp(\mathbf{L} \backslash \mathit{Out}(p, \sigma)), \text{ if } \sigma \in \mathit{Tr}(p) \backslash \mathit{Tr}(s_2)
\end{aligned}$$

It immediately follows that $p \approx_{\mathrm{tr}} s_1$ and $p \, \mathbf{conf} \, s_2$. However, it still needs to be verified that the combination of traces and refusals satisfy certain properties in order for them to define a valid process (e.g., see [12, p. 62]). Most of these properties follow trivially from the given definitions, but the following may require some formal justification:

$$X \in \mathit{Ref}(p, \sigma) \Rightarrow X \cup (L \backslash \mathit{Out}(p, \sigma)) \in \mathit{Ref}(p, \sigma)$$

By contradiction: suppose $X \cup (L \backslash \mathit{Out}(p, \sigma)) \notin \mathit{Ref}(p, \sigma)$ for some $\sigma \in \mathit{Tr}(p)$ such that $X \in \mathit{Ref}(p, \sigma)$. If $\sigma \notin \mathit{Tr}(s_2)$, then we have a straightforward contradiction, because then $\mathbf{L}\backslash Out(p,\sigma) \in Ref(p,\sigma)$ by definition. Otherwise, there must be some $a \in \mathbf{L}\backslash Out(p,\sigma)$ such that $\{a\} \notin Ref(p,\sigma)$, since $X \in Ref(p,\sigma)$. From the fact that $Tr(p) = Tr(s_1)$, we also know that $a \notin Out(s_1,\sigma)$. However, then it follows by $s_1\,\mathbf{cons}\,s_2$, that $\{a\} \in Ref(s_2,\sigma)$, which contradicts that $\{a\} \notin Ref(p,\sigma)$, because $Ref(p,\sigma) = Ref(s_2,\sigma)$ by definition. $\square$

Theorem 20. $\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{red}} = \leq_{\mathrm{tr}} \cap\, \mathbf{cons}$

Proof. In one direction, a simple calculation suffices:

$$\begin{aligned}\mathcal{C}_{\approx_{\mathrm{tr}},\mathbf{red}} &= \approx_{\mathrm{tr}} \circ\, \mathbf{red} = \approx_{\mathrm{tr}} \circ (\leq_{\mathrm{tr}} \cap\, \mathbf{conf}) \\ &\subseteq (\approx_{\mathrm{tr}} \circ \leq_{\mathrm{tr}}) \cap (\approx_{\mathrm{tr}} \circ\, \mathbf{conf}) = \leq_{\mathrm{tr}} \cap\, \mathbf{cons}\end{aligned}$$

In the other direction, assume $s_1 \leq_{\mathrm{tr}} s_2$ and $s_1\,\mathbf{cons}\,s_2$ for some s_1, s_2. By $s_1\,\mathbf{cons}\,s_2$, we have $p \approx_{\mathrm{tr}} s_1$ and $p\,\mathbf{conf}\,s_2$ for some p. By $s_1 \leq_{\mathrm{tr}} s_2$, it then follows that $p \leq_{\mathrm{tr}} s_2$ and therefore that $p\,\mathbf{red}\,s_2$. $\square$

3.3 Summary of Consistency Results

By instantiating the general definition of binary consistency with the implementation relations defined in section 2.3, 36 different notions of consistency were obtained. For most of these notion of consistency a necessary and sufficient condition has been derived, in the form of a characterising relation, under which two specifications can be considered consistent. Eventhough we did not yet find such characterising relations for $\mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{conf}} = \mathcal{C}_{\leq_{\mathrm{tr}},\mathbf{red}}$ and $\mathcal{C}_{\mathbf{red},\mathbf{conf}}$, we conjecture that they exist nevertheless. In the following we denote these two unknown relations $\mathbf{cs}_1$ and $\mathbf{cs}_2$, resp. The obtained results are summarised in Table 3.

In order to verify the consistency of two specifications s_1, s_2 interpreted via implementation relations $\mathbf{imp}_1, \mathbf{imp}_2$, respectively, look up the relation in the row labelled by $\mathbf{imp}_1$ and the column labelled by $\mathbf{imp}_2$. Say this is a relation C. Now, if $s_1\; C\; s_2$, then $s_1\,\mathcal{C}_{\mathbf{imp}_1,\mathbf{imp}_2}\,s_2$ holds.

Table 3. Consistency conditions

	$\leq_{\mathrm{tr}}$	$\approx_{\mathrm{tr}}$	$\mathbf{conf}$	$\mathbf{red}$	$\mathbf{ext}$	$\approx_{\mathrm{te}}$
$\leq_{\mathrm{tr}}$	$\mathcal{P}\times\mathcal{P}$	$\geq_{\mathrm{tr}}$	$\mathbf{cs}_1$	$\mathbf{cs}_1$	$\geq_{\mathrm{tr}}$	$\geq_{\mathrm{tr}}$
$\approx_{\mathrm{tr}}$	$\leq_{\mathrm{tr}}$	$\approx_{\mathrm{tr}}$	$\mathbf{cons}$	$\leq_{\mathrm{tr}} \cap\, \mathbf{cons}$	$\geq_{\mathrm{tr}}$	$\approx_{\mathrm{tr}}$
$\mathbf{conf}$	$\mathbf{cs}_1{}^{-1}$	$\mathbf{cons}^{-1}$	$\mathcal{P}\times\mathcal{P}$	$\mathbf{cs}_2{}^{-1}$	$\mathcal{P}\times\mathcal{P}$	$\mathbf{conf}^{-1}$
$\mathbf{red}$	$\mathbf{cs}_1{}^{-1}$	$\geq_{\mathrm{tr}} \cap\, \mathbf{cons}^{-1}$	$\mathbf{cs}_2$	$\mathcal{C}^2_{\mathbf{red}}$	$\geq_{\mathrm{tr}}$	$\mathbf{red}^{-1}$
$\mathbf{ext}$	$\leq_{\mathrm{tr}}$	$\leq_{\mathrm{tr}}$	$\mathcal{P}\times\mathcal{P}$	$\leq_{\mathrm{tr}}$	$\mathcal{P}\times\mathcal{P}$	$\mathbf{ext}^{-1}$
$\approx_{\mathrm{te}}$	$\leq_{\mathrm{tr}}$	$\approx_{\mathrm{tr}}$	$\mathbf{conf}$	$\mathbf{red}$	$\mathbf{ext}$	$\approx_{\mathrm{te}}$

Fig. 1 relates the consistency relations in terms of their relative strength. The strongest consistency relation ($\approx_{\mathrm{te}}$) can be found at the bottom of the spectrum; the weakest relation ($\mathcal{P}\times\mathcal{P}$) at the top. A line between two relations indicates

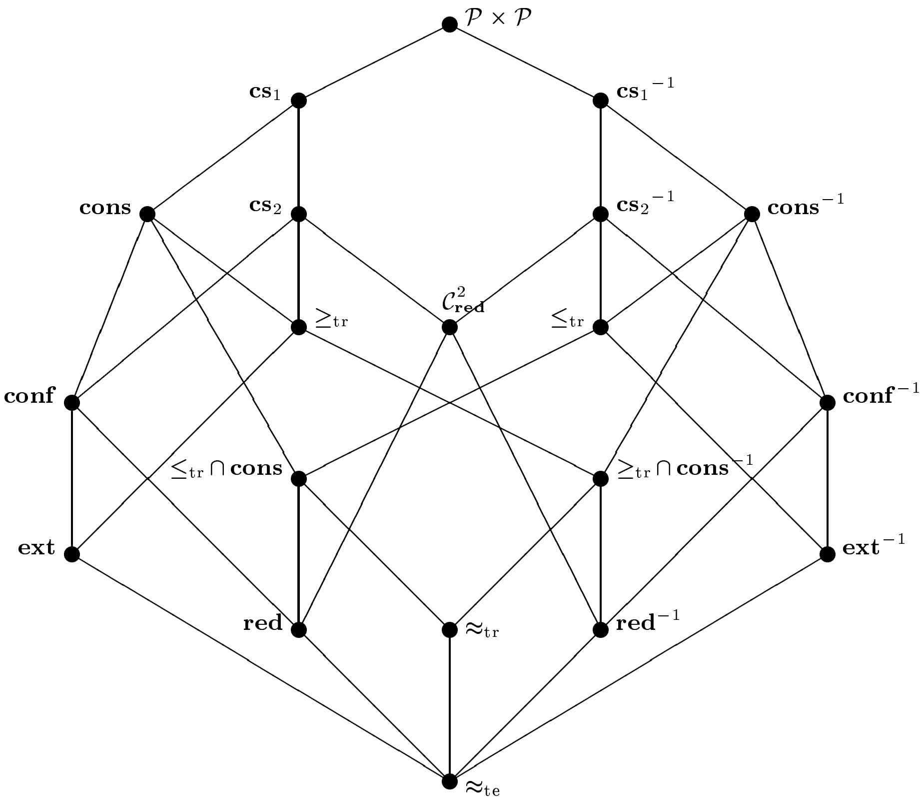

Fig. 1. The spectrum of consistency relations

that the lower one is included in the higher one. It is always sufficient to verify a strictly stronger relation rather than the required notion of consistency.

The relationships depicted in the bottom half of Fig. 1 are mostly well-known results from the literature [13, 12, 3]. The other relationships between consistency relations usually follow from a straightforward monotonicity argument as in Prop. 16 or directly from the definitions.

3.4 Consistency Checking Example

Using the results obtained above, we can now verify the pair-wise consistency of the specifications in Sect. 2.4:

– (Obl, $\mathbf{ext}$) and (Perm, $\leq_{\mathrm{tr}}$) are consistent, because $\mathsf{Obl} \leq_{\mathrm{tr}} \mathsf{Perm}$.
– (Obl, $\mathbf{ext}$) and (Comp, $\mathbf{red}$) are consistent, because $\mathsf{Obl} \leq_{\mathrm{tr}} \mathsf{Comp}$.
– (Eng, $\approx_{\mathrm{te}}$) and (Obl, $\mathbf{ext}$) are consistent, because $\mathsf{Eng}\,\mathbf{ext}\,\mathsf{Obl}$.
– (Perm, $\leq_{\mathrm{tr}}$) and (Comp, $\mathbf{red}$) are consistent, because $\mathsf{Perm}\,\mathbf{red}\,\mathsf{Comp}$, which is a sufficient condition for consistency by Prop. 10.

- (Eng, $\approx_{te}$) and (Perm, $\leq_{tr}$) are *not* consistent, because Eng $\not\leq_{tr}$ Perm. The problem here is that Eng has a trace <coin.coin>, which is not allowed by Perm. This is due to the concurrency in Eng.
- (Eng, $\approx_{te}$) and (Comp, **red**) are *not* consistent, because Eng $\not\mathbf{red}$ Comp. Almost the same problem as above. Eng cannot refuse to do a coin-action, after the initial coin, whereas Comp cannot do such an action.

The main problem with the engineering specification is that it allows a new coin to be inserted already before the last drink has been taken. The inconsistency can be resolved here by adding another synchronisation between the two parts of the engineering specification (the same channel can be used for this):

NewEng := **hide** channel **in** MH |[channel]| DD
MH := coin; channel; channel; MH
DD := channel; (coffee; channel; DD [] tea; channel; DD)

With such a synchronisation in place the money handler will refuse the next coin until the previous drink has been taken out. The new engineering specification is consistent with both the permissions from the enterprise viewpoint and the computational specification.

With the revised engineering specification the set of viewpoint specifications is also *globally* consistent — there exists an implementation that satisfies all four specifications. The common implementation is the engineering description NewEng (see Fig. 2).

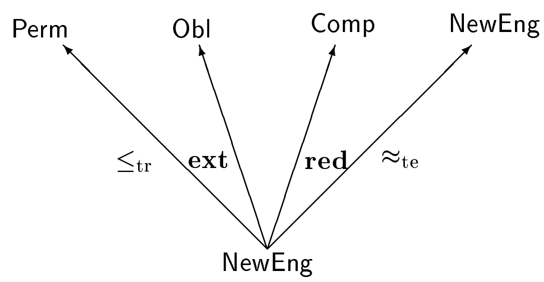

Fig. 2. Global consistency

4 Conclusion

We have presented characterisations of all possible, i.e., balanced and unbalanced, binary consistency relations between six different trace and/or refusal based specification formalisms for process behaviour. These consistency relations are vital if formal specifications are to be used in a multiple viewpoint approach to specification, as is advocated, e.g., by the RM-ODP [9].

Various other approaches to partial process specification have been suggested in the literature [4, 11, 12], some with associated consistency conditions. However, those authors do not consider, what we have called, unbalanced consistency relations.

Ongoing research at the University of Kent focuses on the 'translation' of the consistency relations to consistency checking techniques and tools for more 'user-friendly', graphical specification notations. The main question here is "what implementation relations are (implicitly) assumed by specifiers of State Charts, Sequence Diagrams, etc?"

Another topic for further study is how to deal with specifications at different levels of abstraction. A single action in an enterprise specification may correspond to a more complicated behaviour in the computational specification. In order to support consistency checking between such specifications, we need to consider also implementation relations that incorporate some form of action refinement.

Acknowledgements

We would like to thank Guy Leduc for his comments on an earlier version of this paper.

References

1. T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14:25–59, 1987.
2. G. Booch. *Object oriented design with applications*. Benjamin/Cummings, 1991.
3. E. Brinksma, G. Scollo, and C. Steenbergen. LOTOS specifications, their implementations and their tests. In *Protocol Specification, Testing and Verification VI*, pages 349–360. IFIP, 1987.
4. R. Cleaveland and B. Steffen. A preorder for partial process specifications. In J. C. M. Baeten and J. W. Klop, editors, *CONCUR '90: Theories of Concurrency: Unification and Extension*, LNCS 458, pages 141–151. Springer-Verlag, 1990.
5. A. Finkelstein, J. Kramer, B. Nuseibeh, L. Finkelstein, and M. Goedicke. Viewpoints: a framework for integrating multiple perspectives in system development. *International Journal on Software Engineering and Knowledge Engineering, Special issue on Trends and Research Directions in Software Engineering Environments*, 2(1):31–58, March 1992.
6. C. Fischer and G. Smith. Combining CSP and Object-Z: Finite or infinite trace semantics. In T. Mizuno, N. Shiratori, T. Higashino, and A. Togashi, editors, *FORTE/PSTV'97*, pages 503–518, Osaka, Japan, November 1997. Chapman & Hall.
7. R. J. van Glabbeek. The linear time – branching time spectrum II; the semantics of sequential systems with silent moves (extended abstract). In E. Best, editor, *CONCUR'93*, LNCS 715, pages 66–81. Springer-Verlag, 1993.
8. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.
9. ISO/IEC JTC1/SC21/WG7. Basic Reference Model of Open Distributed Processing. ISO 10746, 1993. Parts 1–4.

10. K. G. Larsen. Ideal specification formalism = expressivity + compositionality + decidability + testability + $\cdots$. In *CONCUR'90. Theories of Concurrency: Unification and Extensions*, LNCS 458, pages 33–56. Springer-Verlag, 1990.
11. K. G. Larsen. Modal specifications. In J. Sifakis, editor, *Automatic Verification Methods for Finite State Systems: Proceedings*, LNCS 407, pages 232–246. Springer-Verlag, 1990.
12. G. Leduc. *On the Role of Implementation Relations in the Design of Distributed Systems using LOTOS*. PhD thesis, University of Liège, Belgium, June 1991.
13. G. Leduc. A framework based on implementation relations for implementing LOTOS specifications. *Computer Networks and ISDN Systems*, 25:23–41, 1992.
14. P. F. Linington. RM-ODP: The Architecture. In K. Raymond and L. Armstrong, editors, *Open Distributed Processing II*, pages 15–33. Chapman & Hall, February 1995.
15. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
16. R. Sinnott and K. J. Turner. Modeling ODP viewpoints. In H. Kilov, W. Harvey, and H. Mili, editors, *Workshop on Precise Behavioral Specifications in Object-Oriented Information Modeling, OOPSLA 1994*, pages 121–128. OOPSLA, October 1994.
17. M. W. A. Steen. *Consistency and Composition of Process Specifications*. PhD thesis, University of Kent at Canterbury, May 1998. Submitted for examination.
18. M. W. A. Steen, H. Bowman, and J. Derrick. Composition of LOTOS specifications. In P. Dembiński and M. Średniawa, editors, *Protocol Specification, Testing and Verification XV*, pages 87–102. Chapman & Hall, 1995.