A Formal Framework for Viewpoint Consistency *

H. Bowman, M.W.A. Steen, E.A. Boiten and J. Derrick

Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK.
(Phone: + 44 1227 764000, Fax 44 1227 762811
Email: {H.Bowman,mwas,E.A.Boiten,J.Derrick}@ukc.ac.uk.)

Abstract. *Multiple Viewpoint* models of system development are becoming increasingly important. Each viewpoint offers a different perspective on the target system and system development involves parallel refinement of the multiple views. Viewpoints related approaches have been considered in a number of different guises by a spectrum of researchers. Our work particularly focuses on the use of viewpoints in Open Distributed Processing (ODP) which is an ISO/ITU standardisation framework. The requirements of viewpoints modelling in ODP are very broad and, hence, demanding. Multiple viewpoints, though, prompt the issue of consistency between viewpoints. This paper describes a very general interpretation of consistency which we argue is broad enough to meet the requirements of consistency in ODP. We present a formal framework for this general interpretation; highlight basic properties of the interpretation and locate restricted classes of consistency. Strategies for checking consistency are also investigated. Throughout we illustrate our theory using the formal description technique LOTOS. Thus, the paper also characterises the nature of and options for consistency checking in LOTOS.

Keywords: Viewpoints, LOTOS, Development Models, Open Distributed Processing, Process Algebra, FDTs.

1 Introduction

System development has classically been viewed in terms of the *waterfall model* of development [48] or some derivative of the model. A single thread of system development is prescribed by the waterfall model, as depicted in figure 1. Specifications are repeatedly refined from an abstract expression of global requirements to a concrete realisation. In such models the validation question to be resolved concerns whether the $n+1$st specification is a valid refinement of the nth specification according to a particular refinement relation. Such refinement relations characterise the manner in which properties of an abstract specification are preserved in a refined specification; for example, refinements may preserve *safety properties* (i.e. statements that something bad cannot happen) or *liveness properties* (i.e. statements that something good must happen).

However, it is now widely recognized that the waterfall model has limitations as a paradigm for system development. Perhaps the most significant limitation of the model is that it presupposes that a full set of requirements for the target system can be identified at the initial stage of system development. This is a restrictive and unrealistic assumption. In practice, the required functionality of a system is identified in a far more fluid and unstructured manner, with requirements evolving incrementally during development as the target system becomes more fully understood; see [52] for a discussion of fluid identification of requirements.

In response to its perceived limitations, adaptations of the waterfall model have been made in a number of directions, e.g. cyclic development [17], rapid prototyping [48], adding feedback [48]. The particular adaptation that we will consider in this paper is the *viewpoints model* of system development. This approach involves dividing the system horizontally relative to the vertical orientation of development. This division is according to a group of views or viewpoints, see figure 2. Each viewpoint offers a different perspective on the system being developed. Such viewpoint modelling is loosely analogous to the use of three angled projection in technical drawing, i.e. plan view and two side elevations.

Importantly, viewpoints support fluid system development since the viewpoint specifications can be iterated between in any arbitrary manner. Thus, functionality can be added to any of the viewpoints at any point during development, often as the result of developments of other viewpoints. In particular, a complete set of requirements is not enforced at the start of development.

*This work was partially funded by British Telecom Research Labs., Martlesham, Ipswich, U.K. and the Engineering and Physical Sciences Research Council under grant number GR/K13035.

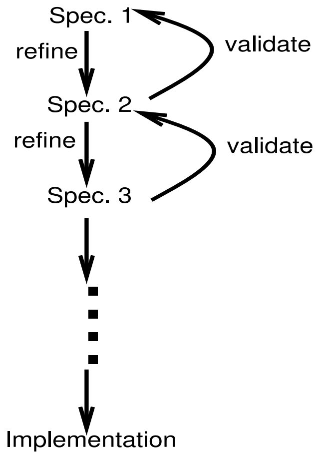

Figure 1: Waterfall Model

Notable proponents of viewpoints modelling include [46] [27] [48] [2] [26]. In addition, a related use of viewpoints can be found in object oriented design methodologies, such as [8] [9]. In fact, variants of viewpoints modelling have been investigated for some time in a number of guises, e.g. aspects [39], partial specification [2] [7] [42] [38], views [37], multiple paradigm specification [54], putting theories together in institutions [26] [28], diagrams [9] and viewpoints [27] [43] [8].

These models typically prompt the central issue of viewpoint consistency, i.e. how to check that multiple specifications of the system do not conflict with one another and are "in some sense" *consistent*. Thus, the central validation question posed by viewpoints is a *horizontal* relating of specifications in contrast to the traditional *vertical* relating of classic waterfall models. In particular, the inherent fluidity of viewpoint specification is reflected in validation scenarios for viewpoint models. Specifically, arbitrary evolution/development of viewpoints is interleaved with snap shot consistency checks, i.e. one off relatings of the viewpoint specifications at a particular point in system development. This is in contrast to classic waterfall development for which a rigid order of development and validation is prescribed.

Our perspective on consistency is tinged by the particular application of viewpoints that our work has been targeted at, viz. the viewpoints model defined in the ISO/ITU Open Distributed Processing (ODP) standardisation framework [36]. ODP defines a generic framework to support the *open* interworking of distributed systems components. A central tenet of ODP is the use of viewpoints in order to decompose the task of specifying distributed systems. ODP supports five viewpoints, *Enterprise*, *Information*, *Computational*, *Engineering* and *Technology*. It is beyond the scope of this paper to give a full introduction to ODP viewpoints modelling, the interested reader is referred to [14], however, in contrast to many other viewpoint models, ODP viewpoints are predefined and in this sense *static*, i.e. new viewpoints cannot be added. Each of the viewpoints has a specific purpose and is targeted at a particular class of specification.

A number of different interpretations can be imposed on the ODP viewpoints model. One such interpretation that we will dwell on here (and which we personally advocate) is that ODP viewpoints define a decomposition of the system development process[1]. This is in contrast to many other viewpoints approaches which target a single phase of system development. For example, the viewpoints model of Finkelstein and co-workers [27] focuses on decomposition in the requirements capture phase of system development: viewpoints are used as a device to decompose the complete system specification at a particular point of system development. They are thus a natural progression from traditional modularization and decomposition paradigms such as subroutines, modules, abstract data types and objects. In contrast, ODP viewpoints can be viewed as decomposing the entire development trajectory. In fact, there is a relationship between the five ODP viewpoints and the phases of system development (we should emphasize though that the relationship is very loose and was not the main motivation

[1] However, it should be emphasized that none of the theory which we present in this paper is specific to this interpretation and in fact, we believe our framework is general enough to embrace all interpretations of ODP viewpoints. [16] gives evidence for this belief by showing that all the currently proposed interpretations of ODP consistency, each of which reflects a different viewpoints interpretation, can be embraced by our framework.

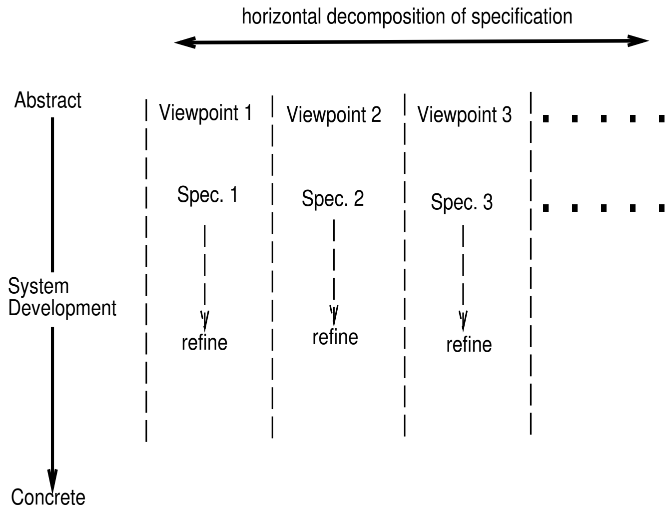

Figure 2: Viewpoints Model

behind the ODP viewpoints model). The following list highlights this relationship:

- The enterprise viewpoint offers a *global requirements capture*.
- The information model defines an information *specification*.
- The computational viewpoint offers an object based interaction model that supports *abstract system design*. This is abstract in the sense that it avoids implementation details such as issues of physical distribution.
- The engineering viewpoint is concerned with prescribing *implementation* mechanisms for the target system.
- The technology viewpoint highlights a possible *realisation* of the system in terms of existing reusable components.

Crucially though, all "phases" of system development exist simultaneously and can be concurrently evolved. Thus, any view on the system development, from the most abstract to the most concrete, can be refined at any point and the complete description of the system comprises specification from all viewpoints.

Another aspect of ODP viewpoints is that it is generally accepted that different viewpoints will be specified in different languages. This is because Formal Description Techniques (FDTs) are variously applicable to the specification requirements of the different viewpoints. For example, Z [47] has been proposed for the information viewpoint and LOTOS [7] for the computational viewpoint.

Figure 3 [20] depicts the relationships that are involved in ODP viewpoints modelling. *Development* yields a specification that defines the system being described more closely. The term development embraces many mechanisms for evolving descriptions towards implementations, one of which is refinement. Because all five viewpoint specifications will eventually be realized by one system, there must be a way to combine specifications from different viewpoints; this is known as *unification*. For specifications in different FDTs to be unified, a *translation* mechanism is needed to transform a specification in one language to a specification in another language. *Consistency* is a relation between groups of specifications.

In our work on consistency we distinguish between *intra* and *inter* language consistency. Intra language consistency considers how multiple specifications in the same language can be shown to be consistent, while inter language consistency considers relations between specifications in different FDTs. The latter issue is a significantly more demanding topic than the former.

In order to inform the interpretation of consistency we choose it is worth considering what we require of such a definition. We offer the following list as a set of general requirements. The consistency definition we seek must,

- be applicable *intra language* for many different FDTs, e.g. must make sense between two Z specifications and also between two LOTOS specifications;

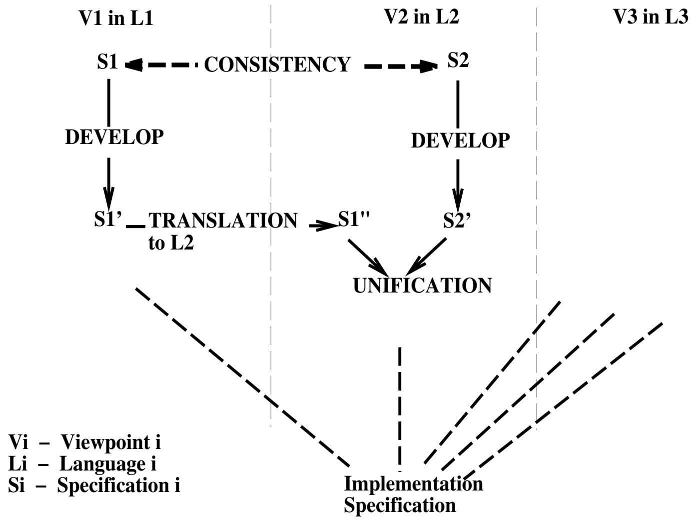

Figure 3: Relating Viewpoints

- be applicable *inter language* between different FDTs, e.g. relate a Z specification to a LOTOS specification.
- support different *classes of consistency check*. There are many different forms of consistency and the appropriate check to apply depends on the viewpoint specifications being considered and the relationship between these viewpoints [15]. For example, it would be inappropriate to check two specifications which express exactly corresponding functionality with the same notion of consistency that is applicable to checking consistency between specifications which extend each other's functionality.
- support global consistency. Much of the work, to date, on consistency in ODP has only considered the case of two viewpoints (what we will call *binary consistency*); for full generality we need any arbitrary number of viewpoints greater than zero.
- allow viewpoints to relate to the target system in different ways. Thus, not only are there different forms of consistency check, but within a consistency check, specifications are related in different ways. For example, the enterprise specification is likely to express global requirements, while the computational specification defines an interaction model. Thus, the relationship between the system being developed and the enterprise specification is very different from the relationship of the system to the computational specification.

This final point prompts our work on, so called, *unbalanced consistency* in which each viewpoint is potentially related to the system under development by a different development relation. For example, the enterprise viewpoint may be related by a logical satisfaction relation while the computational viewpoint may be related by a behavioural conformance relation. Note also that unbalanced consistency is needed to support inter language consistency. This aspect of our work represents a significant departure from existing theoretical work on relating partial specifications, e.g. [2] [54], which has generally restricted itself to what we call, *balanced consistency.*

The contribution of this paper is to define a general interpretation of consistency that satisfies all the above requirements and to make an extensive investigation of the properties of this definition, thus, clarifying the characteristics of arbitrarily general consistency checking. Particular classes of consistency checking which exhibit more manageable properties are then considered. In addition, the paper highlights general strategies for checking consistency according to the prescribed definition. Particular emphasis is placed on the issue of obtaining global consistency, of any arbitrary number of viewpoints, from a series of smaller consistency checks, e.g. binary consistency checks. Throughout we illustrate the consistency checking problem using LOTOS and Z; although, particular emphasis will be placed on LOTOS. Thus, a further contribution of this paper is to characterise consistency checking in the LOTOS setting. As a reflection of this the LOTOS illustrations presented will be relatively extensive.

In addition, in order to clarify the relationship between our work and ODP we present a running ODP example. This is the multiple viewpoint specification of a telecommunication service which has each of its viewpoints described in LOTOS. We illustrate how the consistency of the viewpoint specification can be checked using the techniques introduced in the paper.

The example is deliberately "broad" rather than "deep", i.e. many partial specifications/ viewpoints are included, each of which is rather straightforward. In this way we are able to illustrate the essence of multi viewpoint consistency checking within the bounds of this paper - a more complex example would have swamped the rest of the paper. However a larger case study illustration of our techniques can be found in [5].

The next two sections of this paper provide preliminary background material. Section 2 defines a set of notational conventions and section 3 presents background on LOTOS, its development relations and semantic models. Then section 4 presents our interpretation of consistency, proves some properties of the definition and identifies a number of classes of consistency, viz. *binary consistency*, *complete consistency*, *balanced consistency* and *inter language consistency*. Section 5 highlights basic strategies for checking global consistency. In particular, the pivotal concept of a least developed unification is presented. Section 6 investigates the existence of least developments in the general case of unbalanced consistency and then section 7 considers the same issue in the more restricted setting of balanced consistency. Section 8 reflects on the nature of consistency checking in LOTOS. We then discuss related work in section 9 and present concluding remarks in section 10. An appendix containing proofs of some of the results used in the paper is included at the end of the paper.

2 Preliminaries 1: General Notation

First we present the notation that we will work with. This reflects the search for a general interpretation of consistency by defining very general notational conventions.

It is worth noting here that in selecting our basic notation an important decision is already made: should a categorical or classic set theoretic framework be used. In contrast to some other important research in this area, in particular the theory of institutions [28], we have employed a set theoretic model. Our preference for a classic set theoretic approach is that it integrates more naturally with the ODP model and existing research on the model. In particular, categorical methods are not used within the ODP community.

Descriptions and Relations Between Descriptions. We begin by assuming a set DES of formal descriptions, which contains both formal specifications in languages such as LOTOS and Z and semantic descriptions in notations such as labelled transition systems and ZF set theory.

We assume a set DEV of *development relations*; members of this set relate pairs of descriptions in DES. DEV embraces all possible ways of relating descriptions, e.g. refinement relations or semantic maps. For a particular relation $r \in DEV$, where $r \subseteq DES \times DES$ we define the left and right projections (which are respectively the co-domain and domain of the relation) of r as: $p_l(r) = \{ \ ds \mid \exists ds' \ s.t. \ (ds, ds') \in r \ \}$ and $p_r(r) = \{ \ ds \mid \exists ds' \ s.t. \ (ds', ds) \in r\}$. DEV is subdivided into $intraDEV$, the set of *intra language development relations*, and SEM, the set of *semantic maps*. Importantly, although members of $intraDEV$ and of SEM have very different functions, both can be viewed as relations between pairs of descriptions.

Members of $intraDEV$ are classic development relations within a single formal technique, e.g. the LOTOS or Z refinement relations. Members of SEM are semantic maps between descriptions in formal techniques. Typically they map descriptions from one formal technique to a second formal technique.

Formal Description Techniques. Descriptions are written in formal techniques. The set of all such techniques is denoted FT. Formal techniques are pairs; they are elements of $\mathcal{P}(DES) \times \mathcal{P}(DEV)$. Thus, every formal technique is characterised by the set of possible descriptions in the notation and a set of associated development relations. We require that the right projection of all elements of DEV contains a subset of DES. For a particular formal technique ft we denote the set of all descriptions in ft as DES_{ft} and the set of all development relations as DEV_{ft}. We will also use the notation $intraDEV_{ft}$ to denote the intra language development relations of ft and SEM_{ft} to denote the semantic maps of ft.

Development Relations. These are written dv and if $X \ dv \ X'$ then, in some sense, X is a valid development of X'[2]. Our concept of a development relation generalises all notions of evolving a formal description towards an

[2] Some authors write development relations the other way around, i.e. $X' \ dv \ X$ means that X is a development of X'. However, our choice of orientation corresponds to the direction LOTOS relations are classically written

implementation and thus embraces the many such notions that have been proposed. In particular, DEV contains *refinement* relations, *equivalences* and relations which can broadly be classed as *implementation* relations [42], such as the LOTOS conformance relation **conf**. These different classes of development are best distinguished by their basic properties. Refinement is typically reflexive and transitive (i.e. a preorder); equivalences are reflexive, symmetric and transitive; and implementation relations only need to be reflexive. The distinction between refinement and implementation relations is particularly significant; transitivity is a crucial property in enabling incremental development of specifications towards realizations and implementation relations are typically lacking in this respect.

In general though, we do not require that development relations support any specific properties. In particular, we cannot even assume reflexivity in the general case. This is because, in order to support inter language consistency checking, we will allow development relations to relate descriptions in different notations, in these circumstances reflexivity is not a sensible concept. We will return to this issue in section 4.1.

We introduce the concept of a *terminal element* for a development set.

Definition 1 *Given a set of descriptions $S \subseteq DES$ and a development relation $dv \in DEV$ then $X \in S$ is a terminal element iff $X\ dv\ X'$ for all $X' \in S$. The set of terminal elements is denoted $t(S, dv)$. If such an element does not exist then $t(S, dv) = \emptyset$.*

So, a terminal element is a bottom element for the development ordering in a particular set.

We must also consider what interpretation of equivalence (which we denote $\asymp$) we should adopt. The interpretation that we adopt is:-

$X \asymp_{dv} X'$ iff $\forall Y \in DES,\ Y\ dv\ X \iff Y\ dv\ X'$

which states that two equivalent descriptions have identical development sets, i.e. every description that is a development of one will be a development of the other. This demonstrates that during system development we really can choose any one of a set of equivalent specifications without affecting the possibilities of future development.

$\asymp_{dv}$ can easily be shown to be an equivalence and, in addition, importantly, no properties are required of dv for $\asymp_{dv}$ to be an equivalence. In particular, even if dv is not reflexive or transitive $\asymp_{dv}$ will be an equivalence.

Another standard interpretation of equivalence between specifications is that they are developments of one another. With transitivity of dv this interpretation gives us that two specifications in any cycle by the relation dv are equivalent. However, if dv is in fact a preorder we can obtain that $\asymp_{dv} = dv \cap dv^{-1}$. Thus, we will use these two interpretations interchangeably if development is a preorder. We summarise these results in the following proposition.

Proposition 1
(i) $\asymp_{dv}$ is an equivalence.
(ii) If dv is a preorder then $\asymp_{dv} = dv \cap dv^{-1}$.
(iii) If dv is a preorder then dv is a partial order with identity $\asymp_{dv}$, i.e. elements are viewed to be equal if they are related by $\asymp_{dv}$.

Proof
The only part of this proposition that is not completely trivial is (ii), so we include this proof. Firstly, assume $X \asymp_{dv} X'$, but since by reflexivity, $X\ dv\ X$ this gives us $X\ dv\ X'$ and similarly, $X'\ dv\ X$. Secondly, assume $X\ dv \cap dv^{-1}\ X'$ and take $Y \in DES$, such that $Y\ dv\ X$, but using our assumption and transitivity of dv we get $Y\ dv\ X'$ and similarly, we can show that $Y\ dv\ X'$ implies $Y\ dv\ X$. □

We will also use the following standard concepts from set theory.

Definition 2 *$X \in DES_{ft}$ is a lower bound of $S \subseteq DES_{ft}$ for dv iff $\forall X' \in S,\ X\ dv\ X'$. The set of all lower bounds of S with respect to dv is denoted, $lb(S, dv)$; this set may be empty.*

A lower bound of S is a development of all elements of S. Notice that lower bounds and terminal elements are different concepts; a terminal element must be in the identified set of descriptions while a lower bound has no such constraint. In standard fashion we can also define the concept of a greatest lower bound.

Definition 3 *For $S \subseteq DES_{ft}$ $X \in glb(S, dv)$ is a lower bound such that all other lower bounds are a development of X; i.e. $glb(S, dv) \subseteq lb(S, dv)\ \wedge\ (\forall X \in lb(S, dv)\ \wedge\ \forall X' \in glb(S, dv),\ X\ dv\ X')$. If a greatest lower bound does not exist then $glb(S, dv) = \emptyset$.*

Notice that if dv is a preorder then $t(S, dv)$ and $glb(S, dv)$ are equivalence classes under $\asymp_{dv}$.

3 Preliminaries 2: Background on LOTOS and Introduction of Running Example

LOTOS [33] is a process algebra based specification language used for the formal description of distributed and concurrent systems (see [7] for a general introduction). LOTOS was developed to formally describe the services and protocols of the Open Systems Interconnection Reference Model [32, 35, 34]. Currently, LOTOS is also being used for the specification of ODP systems and standards [36].

The LOTOS language has two parts: a behavioural part and a data part. Most of our work will be with the behavioural part; we will refer to basic LOTOS in the text of this paper when we mean only the behavioural part of the language. The behavioural part is a process algebraic language, related to CCS [45] and CSP [30], in which systems are described in terms of the temporal relationship between externally observable actions.

We will actually use only a subset of LOTOS; the following abstract syntax defines this subset:

$$P \ ::= \ stop \mid \mu;P \mid P_1 \,[]\, P_2 \mid choice\ a \in A \ []\ P \mid P_1 \,|[G]|\, P_2 \mid hide\ G\ in\ P \mid X$$

where P, P', Q, Q', P_1 and P_2 will be used to denote arbitrary LOTOS processes, $\mu \in Act \cup \{i\}$, $A \subseteq Act$ or $A \subseteq Gate$, $G \subseteq Gate$ and X is a process variable. Act is the set of all observable actions and i is the distinguished hidden or internal action. We use α to range over Act and μ to range over $Act \cup \{i\}$. $Gate$ is the set of all gate names, gates locate actions, i.e. they indicate the interaction point at which the action occurs.

We have the null process *stop* (which is synonymous with deadlock); action prefix in order to define sequencing; binary and generalised choice in order to define alternatives, parallel composition (which is parameterised on the gates that must synchronise), the hiding operator and a process variable to invoke behaviour and, possibly, create recursion. We assume that process definition has the form:

$$X := P$$

Although at some points we will associate data parameters with such definitions, e.g. $X(y : T_1, z : T_2) := P$ defines a process named X with formal parameters y (of type T_1) and z (of type T_2). The generalised choice operator defines an arbitrary choice of processes, for example,

$$choice\ a \in \{b, c, d\}\ []\ a;P \ \equiv\ b;P\ []\ c;P\ []\ d;P$$

Also notice that we can create non-deterministic choices. For example,

$$i;P\ []\ a;P'$$

offers a non-symmetric non-determinsitic choice between offering an a or evolving internally to behave as P.

In addition our ODP example will use value passing actions. These are constructed from a gate reference and a value attribute, e.g. the two action instances,

$$g!5 \quad \text{and} \quad g?x : Nat$$

can synchronise to create an action g_5, whereby the value 5 is observed at the gate g. As a by product of this observation the value 5 is bound to the variable x.

In the following $\mathcal{L}_P \subseteq Act$ is the alphabet of observable actions associated with a certain process P; $\mathcal{L}_\mathcal{P}^*$ denotes strings (or traces) over $\mathcal{L}_\mathcal{P}$; the constant $\epsilon \in \mathcal{L}_\mathcal{P}^*$ denotes the empty string, and the variable σ ranges over $\mathcal{L}_\mathcal{P}^*$.

3.1 Two Semantic Models

Labelled Transition Systems. Basic LOTOS has a well-defined operational semantics which maps basic LOTOS behaviour expressions onto Labelled Transition Systems (LTSs). Because this mapping exists and we can express any LTS in basic LOTOS, we can use processes and their corresponding LTSs interchangeably. In particular, relations defined on transition systems are likewise applicable to processes.

A labelled transition system is a tuple, $\langle S, \mathcal{L}, T, s_0 \rangle$. S is a set of states which ranges over the possible process behaviours that the system can reach; $\mathcal{L}$ is a set of action labels; T is a set of transitions of the form $P \xrightarrow{\mu} P'$; and s_0 is a starting state. Notice that without loss of generality we often denote particular LTSs as the diagrams that they induce, e.g. those in figure 6.

Notation	Meaning
$P \xrightarrow{\mu} P'$	denotes a transition, i.e. P can do μ and consequently behaves as P'.
$\stackrel{\epsilon}{\Longrightarrow}$	reflexive and transitive closure of $\xrightarrow{i}$
$P \stackrel{\alpha\sigma}{\Longrightarrow} P'$	$\exists Q, Q'. P \stackrel{\epsilon}{\Longrightarrow} Q \xrightarrow{\alpha} Q' \stackrel{\sigma}{\Longrightarrow} P'$
$P \stackrel{\sigma}{\Longrightarrow}$	$\exists P'. P \stackrel{\sigma}{\Longrightarrow} P'$
$P \stackrel{\sigma}{\not\Longrightarrow}$	$\nexists P'. P \stackrel{\sigma}{\Longrightarrow} P'$

Table 1: Derived transition denotations

Refusal Semantics. In table 1 the notion of transition is generalised to traces. Using this notation we can define the following:

$Tr(P) = \{\sigma \in \mathcal{L}_P^* \mid P \stackrel{\sigma}{\Longrightarrow}\}$, denotes the set of traces of a process P.
$P\ after\ \sigma = \{P' \mid P \stackrel{\sigma}{\Longrightarrow} P'\}$, denotes the set of all states reachable from P by the trace σ.
$Ref(P, \sigma) = \{X \mid \exists P' \in (P\ after\ \sigma),\ \text{s.t.}\ \forall \alpha \in X : P' \stackrel{\alpha}{\not\Longrightarrow}\}$, denotes the refusals of P after σ.
$out(P, \sigma) = \{\alpha \mid \sigma\alpha \in Tr(P)\}$, denotes the set of possible observable actions after the trace σ.

In trace/refusal semantics the behaviour of a process P is characterised by its trace set and the refusals for all traces in that trace set. Stability and divergence properties can also be considered [41], however, the standard LOTOS development relations do not consider these categories, so, we will restrict ourselves to just the trace refusal characterisation of LOTOS specifications. The preference for not including divergence and stability in the standard semantic model arises from the non-catastrophic interpretation of divergence employed in LOTOS, which contrasts with the interpretation classically used with CSP failure semantics.

3.2 LOTOS Development Relations

We reiterate the standard definitions of the most prominent LOTOS development relations. We use the following simple basic LOTOS specifications to illustrate these relations:

$$
\begin{aligned}
R_1 &:= a; b; stop \\
R_2 &:= a; stop \\
R_3 &:= i; c; stop \\
R_4 &:= c; stop \\
R_5 &:= R_1 \,[]\, R_2 \\
R_6 &:= R_2 \,[]\, R_3 \\
R_7 &:= R_4 \,[]\, R_2
\end{aligned}
$$

3.2.1 Trace preorder

Perhaps the simplest meaningful notion of development for LOTOS is trace preorder. This defines refinement as reducing the traces that can be engaged in. The relation is defined as follows:

Definition 4 (trace preorder)
Given two process specifications P and Q, then P is a trace refinement of Q, denoted $P \leq_{tr} Q$, iff:

- $Tr(P) \subseteq Tr(Q)$, *or equivalently*
- $\forall \sigma \in \mathcal{L}_P^*. P \stackrel{\sigma}{\Longrightarrow}$ *implies* $Q \stackrel{\sigma}{\Longrightarrow}$.

In terms of the example basic LOTOS specifications just highlighted, it is straightforward to see that, for example, $R_2 \leq_{tr} R_1 \leq_{tr} R_5$ and $R_2 \leq_{tr} R_6$; but, $\neg(R_1 \leq_{tr} R_2)$.

Intuitively, trace preorder ensures that safety properties are preserved through refinement. Safety properties state that "something bad should not happen", where something bad can be interpreted as a certain trace. Thus, if an abstract specification does not perform a certain degenerate trace then the concrete specification (by trace preorder) cannot perform the trace. Notice that all safety properties hold for the empty trace ϵ or, in other words, *stop* is a trace refinement of any specification.

3.2.2 Conformance

The problem with trace preorder is that it does not ensure that liveness (or deadlock) properties are preserved. A liveness property states that "something good must eventually happen". However, by trace preorder all specifications can be refined to *stop*, i.e. to the process that does nothing. Thus, the "good things" that the abstract specification is able to perform can be refined out.

However, we may wish to ensure that a development of a specification does not deadlock in a situation where the specification would not deadlock, in other words, every trace that the specification *must* do, the development *must* do as well. This requirement is formalised by the **conf** relation [18] [19], which has been adopted as the primary interpretation of conformance for LOTOS.

Definition 5 (conformance)
*Given two process specifications P and Q, then P conforms to Q, denoted P **conf** Q, iff:*

- $\forall \sigma \in Tr(Q).Ref(P,\sigma) \subseteq Ref(Q,\sigma)$, *or equivalently*
- $\forall \sigma \in Tr(Q)$ *and* $\forall A \subseteq \mathcal{L}_P \cup \mathcal{L}_Q$ *we have*
  if $\exists P' \in (P \ after\ \sigma)$ *such that* $\forall \alpha \in A.P' \stackrel{\alpha}{\not\Rightarrow}$,
  then $\exists Q' \in (Q \ after\ \sigma)$ *such that* $\forall \alpha \in A.Q' \stackrel{\alpha}{\not\Rightarrow}$

If we consider our sample specifications again, the reader can check that the following relationships hold: R_1 **conf** R_2, but it is not that case that R_2 **conf** R_1. The latter of these is because after the trace a, R_2 can refuse b, but R_1 cannot.

We will also use the following two development relations which are symmetric subsets of **conf**. These relations are called **conf** symmetric, denoted **cs**, and extended **conf** symmetric, denoted **xcs**. Both relations are introduced for technical reasons. In particular, the introduction of symmetric variants of **conf** arises because this is the notion of compatibility between specifications (called *behavioural compatibility*) used in the architectural semantics of the ODP reference model [36]. For a fuller discussion of the motivation behind these relations the interested reader is referred to [4, 50].

Definition 6 (conf symmetric)
*Given two process specifications P and Q, then P **cs** Q iff P **conf** $Q \ \wedge \ Q$ **conf** P.*

Definition 7 (extended conf symmetric)
*Given two process specifications P and Q, then P **xcs** Q iff P **cs** $Q \ \wedge \ Tr(P) \supseteq Tr(Q)$.*

An alternative derivation of **xcs** is: P **xcs** Q iff P **ext** $Q \ \wedge \ Q$ **conf** P, see section 3.2.4 for the definition of **ext**.

To illustrate these two relations we can see that R_6 **cs** R_3, but $\neg(R_1$ **cs** $R_2)$. In addition, R_6 **xcs** R_3, but $\neg(R_3$ **xcs** $R_6)$.

3.2.3 Reduction

A refinement relation that combines both the preservation of safety and liveness properties is the reduction relation, **red**, defined in [18]. This relation is based upon the classic CSP refinement relation [30] and is also equivalent to "must testing" [29], in both cases, modulo the handling of divergence. **red** interpretes refinement as the reduction of non-determinism in a specification. A typical development strategy using **red** would refine a non-deterministic abstract specification into a deterministic or more nearly deterministic concrete specification.

Definition 8 (reduction)
*Given two process specifications P and Q, then P (deterministically) reduces Q, denoted P **red** Q, iff:*

1. $P \ \leq_{tr} \ Q$, *and*
2. P **conf** Q

By way of illustration, we can see that R_1 **red** R_5, R_2 **red** R_5 and R_3 **red** R_6.

3.2.4 Extension

Inherent in reduction is that the concrete specification cannot "do more" than the abstract specification, i.e. traces cannot be increased. However, as Brinksma argues [18] in some circumstances we would like to add functionality when refining. The extension relation allows for this possibility. Thus it enables new possible traces to be added in a refinement, while preserving the liveness properties of the specification.

Definition 9 (extension)
*Given two process specifications P and Q, then P extends Q, denoted P **ext** Q, iff:*

1. $Tr(P) \supseteq Tr(Q)$, *and*
2. P **conf** Q

By way of illustration R_7 **ext** R_4 and R_7 **ext** R_2, but $\neg(R_4$ **ext** $R_7)$.

3.2.5 Testing Equivalence

A standard interpretation of equivalence is given by the testing equivalence. Intuitively, specifications are testing equivalent if they cannot be distinguished by testing.

Definition 10 (testing equivalence)
*Given two process specifications P and Q, then P is testing equivalent to Q, denoted P **te** Q, iff:*

- P **red** Q *and* Q **red** P, *or equivalently*
- P **ext** Q *and* Q **ext** P, *or equivalently*
- P **xcs** Q *and* Q **xcs** P, *or equivalently*
- $Tr(P) = Tr(Q) \ \wedge \ \forall \sigma \in Tr(P). Ref(P, \sigma) = Ref(Q, \sigma)$.

Notice that $P \ \mathbf{te} \ Q \iff P \ \mathbf{red} \cap \mathbf{red}^{-1} \ Q \iff P \ \mathbf{ext} \cap \mathbf{ext}^{-1} \ Q \iff P \ \mathbf{xcs} \cap \mathbf{xcs}^{-1} \ Q$, so, testing equivalence plays the role of identity, in the sense of $\asymp$, for the preorders **red**, **ext** and **xcs**.

To illustrate this relation, it can be checked that R_3 **te** R_4.

3.2.6 Bisimulation Equivalences

An alternative interpretation of identity is given by the bisimulation equivalences, *strong* and *weak bisimulation* [45]. These offer stronger interpretations of equivalence based upon the observable behaviour of specifications. The definition of weak bisimulation equivalence, $\approx$, of LOTOS processes is given by the following two definitions:

Definition 11 (weak bisimulation relation)
A binary relation $\mathcal{R}$ over LOTOS processes is a weak bisimulation if $P_1 \ \mathcal{R} \ P_2$ implies, $\forall \sigma \in \mathcal{L}_P^* \cup \mathcal{L}_Q^*$

1. *if* $\exists P_1'.P_1 \stackrel{\sigma}{\Longrightarrow} P_1'$ *then* $\exists P_2'.P_2 \stackrel{\sigma}{\Longrightarrow} P_2'$ *and* $P_1' \mathcal{R} P_2'$; *and*
2. *if* $\exists P_2'.P_2 \stackrel{\sigma}{\Longrightarrow} P_2'$ *then* $\exists P_1'.P_1 \stackrel{\sigma}{\Longrightarrow} P_1'$ *and* $P_1' \mathcal{R} P_2'$.

Definition 12 (weakly bisimilar)
Two LOTOS processes P_1 and P_2 are weakly bisimilar, denoted $P_1 \approx P_2$, if there exists a weak bisimulation relation $\mathcal{R}$ such that $P_1 \ \mathcal{R} \ P_2$.

Strong bisimulation, denoted $\sim$, is defined in a similar manner to weak bisimulation, except i actions are matched in addition to observable actions. Hence strong bisimulation is an even stronger notion of observational identity than weak bisimulation.

3.2.7 Discussion: Properties of the Development Relations

Apart from **cs** and **xcs** the properties of the development relations presented above have been well documented in the literature. Some of these properties are reviewed here, proofs of these results can be found in the appendix.

Proposition 2
(i) $\leq_{tr}$, **red** *and* **ext** *are preorders.*
(ii) **te**, $\sim$, *and* $\approx$ *are equivalences.*
(iii) $\sim \subset \approx \subset$ **te** $\subset$ **cs**
(iv) **conf** *is reflexive, but neither symmetric nor transitive.*
(v) **cs** *is reflexive and symmetric, but not transitive.*
(vi) **xcs** *is a preorder.*

Thus, $\leq_{tr}$, **red**, **ext** and **xcs** can be classed together as *refinement* relations; **te**, $\sim$, and $\approx$ can be classed together as *equivalences*; while **conf** and **cs** are weaker *implementation* relations.

3.3 Running Example

In this subsection we describe a simple ODP system which will be used as a running example throughout the paper. The basic scenario is the multiple viewpoint specification of a telecommunications service. The service has the following general behaviour:

> The service accepts requests (the action *request*) to open up communication channels and then offers different possible varieties of channel, e.g. just an audio connection (the action *transA*) or a (full sound and image) video connection (the action *transV*). Actually many different types of connection could be provided.

The following will be assumed in the example:

- A set *ID* of user identifiers is assumed. These are used for accounting purposes. When a user of the system requests a communication he/she specifies their identifier and the cost of the communication is charged to them. The set *ID* is constructed as follows - $ID = ID' \cup \{def\}$, where *def* indicates a "default" identifier. Thus, in situations where personalised charging is not being used, e.g. within a particular commercial organization, a default identifier can be provided.
- The set of gates in this domain is denoted GG and it has the following subsets:
  - $F = \{\, request\,,\, transA\,,\, transV\,\}$ where *transA* transmits audio and *transV* transmits video;
  - $H = \{\, transA\,,\, transV\,,\, transT_1\,, ...,\, transT_n\,\}$ where $T_1, .., T_n$ are alternative communication types, perhaps data links or various qualities of video.
- π is a function which takes a set of *gate* names and returns the set of all possible *action*s that can be generated from that set. It associates data with the gates in all relevant possible ways. For example, if the gates a and b can only have one data attribute of type boolean then,

$$\pi(\{a, b\}) = \{\, a!true\,,\, a!false\,,\, b!true\,,\, b!false\,\}$$

Now we provide viewpoint specifications for the enterprise, computational and engineering viewpoints.

Enterprise. This viewpoint is itself composed of a number of partial specifications, each enforces a different enterprise constraint on the target system. We have one permission and two obligations.

- *Permission:* The enterprise permits the system to be constructed using gates in the set F. This is because the organisation only permits certain transmission media types to be used in their domain. We can enforce this constraint by associating the reduction relation with the following specification:

  Perm := (choice b∈F [] i; b?x:ID; Perm) [] i; stop

So, *Perm* allows any arbitrary non-deterministic behaviour on the actions *request?x:ID*, *transA?x:ID*, and *transV?x:ID*. Thus, any specification that does not use an observable action other than one of these three will be a reduction of *Perm*.

In fact, we could interpret this specification with the relation $\leq_{tr}$ and get the same effect. Indeed, in order to illustrate different varieties of consistency check we will at different stages during the sequel interpret this specification with $\leq_{tr}$, **conf** and **red**. Note that the relations $\leq_{tr}$ and **conf** define **red** (i.e. $\leq_{tr} \cap$**conf**). So, by considering these other two relations we implicitly consider part of **red** itself.

- *Obligation 1:* The system is obliged to allow, i.e. offer, a (non-default) request immediately. This effect is obtained by associating the extension relation with the specification.

  $$Obl1 := request?x{:}ID';\ stop$$

- *Obligation 2:* The enterprise specification also requires that every *transA* or *transV* must be preceded by a *request*, *ID*s must match (between requests and transmissions) and a "new" request cannot be performed until the last one has been matched to a transmission. We enforce this constraint by associating trace preorder with the following specification[3]:

  $$\begin{array}{rcl} Obl2 & := & (choice\ a\in \pi(GG\text{-}F)\ []\ a;\ Obl2) \\ & & []\ request?x{:}ID';\ Trans(x) \\ Trans(x{:}ID') & := & (choice\ a\in \pi(GG\text{-}F)\ []\ a;\ Trans(x)) \\ & & []\ transA!x;\ Obl2 \\ & & []\ transV!x;\ Obl2 \end{array}$$

  This specification will allow all traces that satisfy this constraint[4].

Computational. Our scenario is that a "generic" service interface is provided by the computational viewpoint. This defines a spectrum of allowed computational behaviour, which will be specialized according to the constraints imposed by the other viewpoints. In particular, the enterprise constraints will specialize the computational viewpoint according to the needs of a particular organisation. The computational specification is as follows:

$$\begin{array}{rcl} Comp & := & i;\ request?x{:}ID';\ (choice\ b\in H\ []\ i;\ b!x;\ Comp) \\ & & []\ (choice\ b\in H\ []\ i;\ b!def;\ Comp) \end{array}$$

The specification offers a number of possible behaviours - the first branch accepts requests (with identifiers) and then offers a non-deterministic choice of transmission with any of the possibly available media types, with (as discussed earlier), the identifier included for accounting purposes. The second branch offers a default behaviour. Thus, the service could be specialized to one where charging is not required and the service can transmit with the default value, *def*, associated.

Engineering. The system is composed of two components - a request handler (*RH*) and an IO handler (*IOH*) which is itself composed of a number of tranmission devices: here audio and video devices. These two top level components of the engineering viewpoint, *RH* and *IOH*, communicate via a channel. As the channel is only used for internal communication, it is hidden from the environment. The viewpoint is specified as follows:

$$\begin{array}{rcl} Eng & := & hide\ channel\ in\ RH\ |[channel]|\ IOH \\ RH & := & request?x{:}ID';\ channel!x;\ RH \\ IOH & := & channel?x{:}ID';\ (transA!x;\ IOH\ []\ transV!x;\ IOH) \end{array}$$

The specification offers an external choice of transmitting on the audio or on the video channel and the user can select between them. Of course in reality, the engineering behaviour would be much more complex. However, we

[3]In fact, we could avoid the function π by using generalised choice over data, e.g. *choice* $a\in \pi(D)$ *[](a;B)* is equivalent to *choice* $b\in D$ *[] (choice x:T [] b!x;B)* where T is the type associated with the action a. However we prefer to use the π function as it leads to a more concise presentation.

[4]In fact, as is common with enterprise specification, this constraint could more easily be expressed using a logical notation. However, since we are restricting ourselves to LOTOS illustrations we have to give a slightly cumbersome formulation.

abstract from this complexity in the context of this illustrative example. The behaviour is interpreted with the testing equivalence relation.

During the remainder of this paper we will consider (incrementally) how these viewpoints can be checked for consistency. We will begin by considering different pairwise consistency relationships, i.e. between pairs of viewpoints, and then we will consider the global consistency of the example.

4 A General Interpretation of Consistency

We are now in a position to introduce our general interpretation of consistency and to clarify the basic properties of the interpretation. This section is divided into a number of subsections, the first introduces the definition and then the following subsections consider different classes of consistency: *binary consistency*, *complete consistency*, *balanced consistency* and *inter language consistency*. The section concludes with a discussion.

4.1 Consistency Definition

Broadly a consistency check is a function from a group of descriptions, $X_1, X_2, ..., X_n$ to a boolean; true is returned if all the descriptions are consistent and false otherwise. This check is parameterised upon a corresponding group of development relations, $dv_1, dv_2, ..., dv_n$, one per description; it is denoted, $C(dv_1, X_1)(dv_2, X_2)...(dv_n, X_n)$, a shorthand for which is $\overset{1..n}{C}(dv_i, X_i)$.

Type Correctness. The validity of the check has two elements: *type correctness* and *consistency.*

Definition 13 (Type Correctness)
$C(dv_1, X_1)(dv_2, X_2)...(dv_n, X_n)$ *is type correct iff* $(X_1 \in p_r(dv_1) \ \wedge \ X_2 \in p_r(dv_2) \ \wedge \ ... \wedge \ X_n \in p_r(dv_n)) \ \wedge$ $(p_l(dv_1) \cap p_l(dv_2) \cap ... \cap p_l(dv_n) \neq \emptyset)$.

Type correctness ensures, firstly, that for every description the corresponding development relation, i.e. dv_i for X_i, is correctly typed with regard to the description. In addition, type correctness ensures that the target types of the relations have some intersection. This check has the function of determining that the consistency check being attempted is sensible. Type correctness will not be an issue for intra language consistency, but will be necessary when determining an appropriate inter language consistency check to apply.

Illustration 1 *An inter language consistency check between Z and LOTOS, which relates Z by the standard Z refinement relation, denoted $\sqsubseteq$, [47] and LOTOS by reduction would typically not be type correct because the co-domains of $\sqsubseteq$ and* **red** *have no intersection. However, if a common semantics for LOTOS and Z were defined, such as the extended transition system considered in [25], and adaptations of $\sqsubseteq$ and* **red** *were made to relate Z specifications respectively LOTOS specifications to the common semantics, then type correct consistency checks could be defined.*

When writing $C(dv_1, X_1)(dv_2, X_2)...(dv_n, X_n)$, unless otherwise stated, we will assume the check has already been shown to be type correct.

Consistency. Once type correctness has been determined we can investigate consistency. Intuitively we view n specifications $X_1, X_2, ..., X_n$ as consistent if and only if there exists a physical implementation which is a realization of all the specifications, i.e. X_1, X_2 through to X_n can be implemented in a single system.

This interpretation of consistency has similarities to satisfaction in a logical setting. A conjunction of propositions $\phi_1 \ \wedge \ \phi_2 \ \wedge \ .... \ \wedge \ \phi_n$ is satisfiable if there exists a single model which individually satisfies all the propositions.

Illustration 2 *Figures 4(a) and 4(b) illustrates our intuition of consistency, in both depictions descriptions are related to their set of possible realisations by a relation implements (denoted* **imp***). Thus, the Venn diagrams in the implementation plane depict the set of possible realisations of each description. It should be clear that the three descriptions in figure 4(a) are consistent because their set of possible implementations intersect, i.e. they have at least one common implementation. In contrast, the three descriptions in figure 4(b) are not consistent, although, the pairs S4 and S5 and S5 and S6 are mutually consistent.*

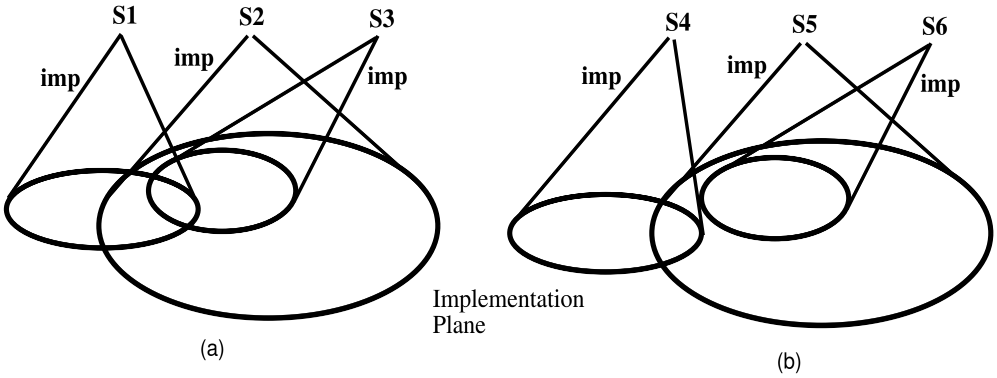

Figure 4: Common Implementation Models

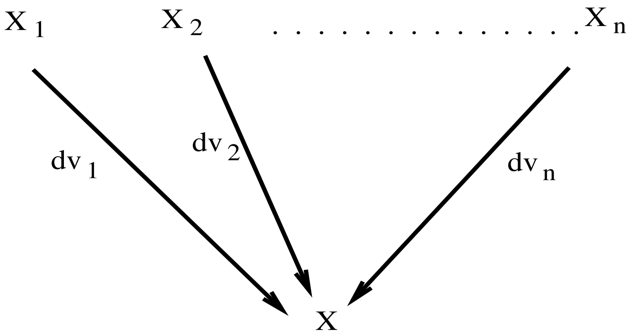

Figure 5: A Consistency Check

However, rather than talk explicitly about implementation, as this interpretation does, we would like to work purely in the formal setting and define consistency purely in terms of descriptions and relations between descriptions. Thus, we define consistency in terms of a common (formal) description, X, and a list of development relations, $dv_1, dv_2, ..., dv_n$. Definition 14 states that n descriptions are consistent if and only if there exists a description that is a development of X_1 according to dv_1, X_2 according to dv_2, through to X_n according to dv_n. See figure 5 for an illustration.

Definition 14 (Consistency)
$\overset{1..n}{C}(dv_i, X_i)$ *holds, iff* $\exists X \in DES \ . \ X \ \ dv_1 \ X_1 \ \wedge ... \wedge \ X \ dv_n \ X_n$.

For n descriptions to be consistent this definition requires that X is a common development of X_i for all i between 1 and n. Notice that we allow the descriptions to be related to their common development in different ways, i.e. if $dv_i \neq dv_j$. As discussed in section 1 this is needed in order to support unbalanced consistency as required by ODP viewpoints.

In most cases $X_1, X_2, ..., X_n$ in the above definition will all be specifications, however, X will commonly be a semantic representation. In particular, if some of $X_1, X_2, ..., X_n$ are in different languages then X is very likely to be in a common semantic notation. The properties that enable a semantic notation to be suitable for representing common developments of specifications in different formal techniques will be discussed in section 4.5. If $X_1, X_2, ..., X_n$ are in the same formal technique then $\overset{1..n}{C}(dv_i, X_i)$ is called an *intra language* consistency check and if for some i and j between 1 and n, X_i and X_j are in different formal techniques then $\overset{1..n}{C}(dv_i, X_i)$ is called an *inter language* consistency check.

In previous presentations of consistency we have often included a check for implementability in our basic definition. This check is called internal validity and it ensures that the common development, X in definition 14, is truly implementable, it is denoted $\Psi(X)$. Such a check was justified on the grounds that descriptions relate to physical implementations in different ways for different specification languages. In particular, for a language such as Z valid specifications can be defined which are not implementable. Consequently, in some specification languages it may be possible for a group of descriptions to have a common development, but not to be consistent, since the common development is not itself implementable.

However, in this paper we avoid explicitly referring to internal validity in our consistency definition. This is because the check can be incorporated with development. For example, if we wish to check $X_1, ..., X_n$ for consistency (and internal validity) using development relations $dv'_1, ..., dv'_n$ we can perform the consistency check $\overset{1..n}{C}(dv_i, X_i)$ where $X \; dv_i \; Y$ if and only if $X \; dv'_i \; Y$ and $\Psi(X)$, and we have a check that conforms to the format of our simple consistency check of definition 14. Acknowledging that development may be enhanced in this way leads to a conceptually simpler and more uniform treatement of consistency and of the theory surrounding it.

Unification. The concept of a unification, as highlighted in figure 3, can now be easily formalized as the common development of definition 14. The set of all such common developments is defined in the obvious way:

Definition 15 (Unification Set)
$\mathcal{U}(dv_1, X_1)(dv_2, X_2)...(dv_n, X_n) = \{X \in DES \mid X \; dv_1 \; X_1 \;\; \wedge ... \wedge \;\; X \; dv_n \; X_n\}$.

$\overset{1..n}{\mathcal{U}}(dv_i, X_i)$ is used as a shorthand for $\mathcal{U}(dv_1, X_1)(dv_2, X_2)...(dv_n, X_n)$. Notice $\overset{1..n}{C}(dv_i, X_i)$ holds if and only if $\overset{1..n}{\mathcal{U}}(dv_i, X_i) \neq \emptyset$.

The following result can be immediately observed.

Proposition 3
$\{(dv_1, X_1), ..., (dv_n, X_n)\} \supseteq \{(dv'_1, X'_1), ..., (dv'_m, X'_m)\} \Longrightarrow \mathcal{U}(dv_1, X_1)...(dv_n, X_n) \subseteq \mathcal{U}(dv'_1, X'_1)...(dv'_m, X'_m)$

Proof
Take $X \in \overset{1..n}{\mathcal{U}}(dv_i, X_i)$ i.e. $X \; dv_1 \; X_1 \;\; \wedge ... \wedge \;\; X \; dv_n \; X_n$. For any X'_j s.t. $1 \leq j \leq m$, $X \; dv'_j \; X'_j$ since by our hypothesis $\exists X_i$ (for $1 \leq i \leq n$) such that $dv_i = dv'_j$ and $X_i = X'_j$. So, $X \in \mathcal{U}(dv'_1, X'_1)...(dv'_m, X'_m)$, as required.
$\square$

This proposition expresses the obvious result that a unification of n specifications is a unification of a subset of the n specifications. An immediate corollary of proposition 3 is:

Corollary 1
$\overset{1..n}{\mathcal{U}}(dv_i, X_i) \subseteq \mathcal{U}(dv_i, X_i)...(dv_j, X_j)$ *for* $1 \leq i, j \leq n$.

The following sections consider a number of different classes of consistency.

4.2 Binary Consistency

An important special class of consistency is binary consistency, i.e. the consistency check $C(dv_1, X_1)(dv_2, X_2)$. Binary consistency is a binary relation and is often written, $X_1 \; C_{dv_1, dv_2} \; X_2$.

4.2.1 Basic Properties

The possibility of inter language consistency makes it difficult to obtain general properties for this binary relation.

Proposition 4
Binary consistency is in general neither (i) reflexive, (ii) symmetric or (iii) transitive.

Proof
(i) Reflexivity is the case $C(dv, X)(dv', X)$, which is equivalent to $C(dv \cap dv', X)$; this could be false if either dv or dv' are not reflexive [5].

[5] One reason for development not being reflexive is if it incorporates an implementability/internal validity check, as discussed in the previous section. In such a circumstance an unimplementable specification might not be viewed as a development/implementation of itself.

(ii) Assuming $C(dv_1, X_1)(dv_2, X_2)$ is true, in the case of inter language consistency $C(dv_1, X_2)(dv_2, X_1)$ is likely not even to be type correct. Thus, in its most general form, symmetry of consistency does not even yield a type correct consistency check.

(iii) Assuming $C(dv_1, X_1)(dv_2, X_2)$ and $C(dv_3, X_2)(dv_4, X_3)$ hold then transitivity requires us to show that X_1 and X_3 are consistent, however, according to what development relations will we check consistency? The transitivity variant that we would like is that $C(dv_1, X_1)(dv_4, X_3)$ follows from the assumptions. However, nothing in our assumption guarantees that, $p_l(dv_1) \cap p_l(dv_4) \neq \emptyset$, thus, $C(dv_1, X_1)(dv_4, X_3)$ may not be type correct. Furthermore, even if we assume type correctness of $C(dv_1, X_1)(dv_4, X_3)$, consistency will not always hold, since $C(dv_1, X_1)(dv_2, X_2)$ and $C(dv_3, X_2)(dv_4, X_3)$ are likely to have different common developments that cannot be related (the second example of illustration 2, depicted in figure 4(b), highlights such a situation). $\square$

However, if we restrict ourselves to reflexive development (which would make sense in languages where all specifications are implementable) we can obtain reflexivity of consistency.

Proposition 5
If dv_1, $dv_2 \in DEV_{ft}$ are reflexive on DES_{ft}, then $\forall X \in DES_{ft}$, $C(dv_1, X)(dv_2, X)$ holds, i.e. consistency is reflexive.

Proof
Reflexivity of dv_1 and dv_2 gives us reflexivity of $dv_1 \cap dv_2$ which implies that X is the required common development. $\square$

This proposition implies that consistency is reflexive for a language such as basic LOTOS in which development is at least reflexive.

4.2.2 Embracing Development

One motivation for considering unbalanced consistency is to enable us to address situations in which a viewpoint is "conceptually" a direct development of a second viewpoint. Such relations between viewpoints are not strictly in accordance with viewpoints modelling, but for many particular viewpoint models such a conceptual relationship between viewpoints may arise. For example, some researchers like to think of specific pairs of ODP viewpoints as developments of one another, e.g. the engineering viewpoint may be seen as a refinement of the computational viewpoint. Thus, we would like to embrace the standard development relations into our interpretation of consistency, i.e. to instantiate consistency in such a way that the relation induced between viewpoints is equivalent to development. For LOTOS this means giving instantiations of consistency that model the LOTOS development relations, **conf**, **red**, **ext**, etc.

The following general results characterise the relationship between preorder refinement and consistency.

Proposition 6
If dv is a preorder then $\forall X_1, X_2 \in DES_{ft}$,

(i) $X_1 \; dv \; X_2 \iff X_1 \; C_{dv^{-1}, dv} \; X_2$.
(ii) $X_1 \; dv \; X_2 \iff X_1 \; C_{(dv \cap dv^{-1}), dv} \; X_2$.

Proof
((i) $\Longrightarrow$) $X_1 \; dv \; X_2$ by assumption, but also $X_1 \; dv^{-1} \; X_1$ by reflexivity of dv, so, X_1 is a common development.
((i) $\Longleftarrow$) Assume $\exists X$ *s.t.* $X_1 \; dv \; X \wedge X \; dv \; X_2$ then by transitivity of dv, $X_1 \; dv \; X_2$.
((ii) $\Longrightarrow$) $X_1 \; dv \; X_2$ by assumption, also $X_1 \; dv \cap dv^{-1} \; X_1$ by reflexivity.
((ii) $\Longleftarrow$) $X_1 \; C_{(dv \cap dv^{-1}), dv} \; X_2 \implies X_1 \; C_{dv^{-1}, dv} \; X_2$ and $X_1 \; C_{dv^{-1}, dv} \; X_2 \implies X_1 \; dv \; X_2$ (by (i) $\Longleftarrow$). $\square$

Corollary 2
For dv a preorder, $dv = C_{dv^{-1}, dv} = C_{\asymp_{dv}, dv}$.

LOTOS Illustration 1 *We have the following specific instantiations of corollary 2 for LOTOS preorders:-*

Proposition 7
(i) $\leq_{tr} = C_{\leq_{tr}^{-1}, \leq_{tr}} = C_{\asymp_{\leq_{tr}}, \leq_{tr}}$; *(ii)* $\mathbf{red} = C_{\mathbf{red}^{-1}, \mathbf{red}} = C_{\mathbf{te}, \mathbf{red}}$; *(iii)* $\mathbf{ext} = C_{\mathbf{ext}^{-1}, \mathbf{ext}} = C_{\mathbf{te}, \mathbf{ext}}$; *and*
(iv) $\mathbf{xcs} = C_{\mathbf{xcs}^{-1}, \mathbf{xcs}} = C_{\mathbf{te}, \mathbf{xcs}}$.

Since **conf** *and* **cs** *are not transitive we have to work a bit harder to relate these notions of development. Firstly, we note the following negative result:-*

Proposition 8
$\mathbf{conf} \not\supseteq C_{\mathbf{conf}^{-1},\mathbf{conf}}$

Proof
Let $P_1 := b; stop[]i; a; stop$, $P_2 := b; c; stop[]i; a; stop$ *and* $P := i; a; stop$ *then,* P *is the required common development to give* $P_1 \; C_{\mathbf{conf}^{-1},\mathbf{conf}} \; P_2$, *but* $\neg(P_1 \; \mathbf{conf} \; P_2)$. $\square$

However, the following stronger result enables us to embrace **conf**.

Proposition 9
$\mathbf{conf} = C_{\mathbf{te},\mathbf{conf}}$.

Proof
$(\mathbf{conf} \subseteq C_{\mathbf{te},\mathbf{conf}})$
Assume $P_1 \; \mathbf{conf} \; P_2$, *but in addition from reflexivity of* **te**, $P_1 \; \mathbf{te} \; P_1$ *and, thus,* P_1 *is the required common development.*

$(C_{\mathbf{te},\mathbf{conf}} \subseteq \mathbf{conf})$
Take P *such that* $P \; \mathbf{te} \; P_1$ *and* $P \; \mathbf{conf} \; P_2$. *If we expand these out we get:*
$Tr(P) = Tr(P_1) \;\wedge\; \forall \sigma \in Tr(P),\; Ref(P,\sigma) = Ref(P_1,\sigma) \;\;\wedge$
$\forall \sigma \in Tr(P_2),\; Ref(P,\sigma) \subseteq Ref(P_2,\sigma)$
Equality of the traces of P_1 *and* P *implies that there are no traces of* P_2 *that* P_1 *could do, but* P *could not do, thus,* $\forall \sigma \in Tr(P_2),\; Ref(P,\sigma) = Ref(P_1,\sigma)$ *and thus,* $\forall \sigma \in Tr(P_2),\; Ref(P_1,\sigma) \subseteq Ref(P_2,\sigma)$. *So,* $P_1 \; \mathbf{conf} \; P_2$ *as required.* $\square$

In addition, a proof of the following result can be found in [4].

Proposition 10
$\mathbf{cs} = C_{\mathbf{xcs},\mathbf{xcs}}$.

So, we have shown how equivalent instantiations of consistency can be given for all the following LOTOS relations: $\leq_{tr}$, **red**, **ext**, **xcs**, **conf** *and* **cs**. *This only leaves the equivalence relations; section 4.4 will show that these can be easily embraced.*

ODP Illustration 1 *In addition, we can illustrate these LOTOS instantiations of our theory in the context of our running ODP example.*

Trace Preorder. *The following holds,*

$$Eng \leq_{tr} Perm$$

Thus, by proposition 7(i) we know that,

$$Eng \; C_{\asymp_{\leq_{tr}},\leq_{tr}} \; Perm$$

which has the form of the binary consistency check we are interested in between Eng and Perm.

Conformance. *In addition,*

$$Eng \; \mathbf{conf} \; Perm$$

Thus, by proposition 9 we know that,

$$Eng \; C_{\mathbf{te},\mathbf{conf}} \; Perm$$

Reduction. *Now, if we put the last two cases together, we get,*

$$Eng \; \mathbf{red} \; Perm$$

which extends the previous properties and by proposition 7(ii),

$$Eng \; C_{\mathbf{te},\mathbf{red}} \; Perm$$

which is exactly what we are interested in.

Extension. *Also, we have that,*

$$Eng\, \mathbf{ext}\, Obl1$$

which by proposition 7(iii) gives us,

$$Eng\, C_{\mathbf{te},\mathbf{ext}}\, Obl1$$

which is another one of our constituent binary consistency checks.

4.3 Complete Consistency

For a set of descriptions a particular consistency check may always hold, i.e. any subset of descriptions will be consistent. This property is called *complete consistency* and is defined as:

Definition 16 Complete Consistency

A set of descriptions, ds, is completely consistent according to $dv_1, ..., dv_n$ *iff* $\forall X_1, ..., X_n \in ds,\ \overset{1..n}{C}(dv_i, X_i)$.

Note that this definition assumes that the consistency check is type correct for any n descriptions in the set. In the inter language setting, this will frequently fail to hold. Thus, complete consistency is a particularly useful concept in the intra language setting. In particular, if an FDT is known to be completely consistent there is no need to undertake consistency checking.

The following result is straightforward, it gives us a sufficient condition for complete consistency (remember the notation $t(ds, dv)$ denotes the set of terminal elements of ds according to dv).

Proposition 11

$t(ds, \overset{n}{\cap} dv_i) \neq \emptyset \implies \overset{1..n}{C}(dv_i, X_i)$ *for all* $X_1, ..., X_n \in ds$.

Proof

Clearly, $X \overset{n}{\cap} dv_i\ X' \implies X\ dv_i\ X'$, so the result follows immediately. □

LOTOS Illustration 2 *The following cases highlight complete consistency classes for basic LOTOS.*

(i) Consider $C(\mathbf{conf}, X)(\mathbf{ext}, X')$ *for any* $X, X' \in DES_{basicLOTOS}$. *Then the proces* ω*, which offers a deterministic choice of all possible actions at every point, defined as,*

ω := *choice* $a \in \mathcal{L}_X \cup \mathcal{L}_{X'}$ *[] a;* ω

satisfies $Tr(\omega) = (\mathcal{L}_X \cup \mathcal{L}_{X'})^*$ *and* $\forall \sigma \in Tr(\omega)^*,\ Ref(\omega, \sigma) = \{\emptyset\}$*, i.e. it performs all possible traces and refuses nothing. Thus,*

$\omega\ \mathbf{ext}\ X$ *for all* $X \in DES_{basicLOTOS}$

and since $\mathbf{ext} \cap \mathbf{conf} = \mathbf{ext}$*, proposition 11 is satisfied. So, LOTOS is completely consistent according to* **conf** *and* **ext**.

This complete consistency property of **conf** *and* **ext** *implies that in our ODP running example,*

$$Perm\, C_{\mathbf{conf},\mathbf{ext}}\, Obl1$$

holds automatically, with no further analysis required.

(ii) However, $C(\mathbf{red}, X)(\leq_{tr}, X')$ *for any* $X, X' \in DES_{basicLOTOS}$ *does not hold. In particular,* **red** *has no terminal element, i.e.* $t(\mathbf{red}, DES_{basicLOTOS}) = \emptyset$*. For such a terminal element to exist it must be a trace subset of any basic LOTOS description, which suggests it should be the basic LOTOS behaviour stop, but stop refuses everything. Furthermore, the descriptions* $X := a; stop$ *and* $X' := b; stop$ *serve as a counterexample that shows that basic LOTOS is not completely consistent for* $C(\mathbf{red}, X)(\leq_{tr}, X')$*. This is because* X *has no reductions other than itself (up to equivalence) while* X' *is only trace refined by itself and stop (up to equivalence).*

Thus, for example, it is not the case that we can automatically deduce that in our ODP running example,

$$Comp\, C_{\mathbf{red},\leq_{tr}}\, Perm$$

In fact, this does hold, but in order to demonstrate that it does hold we will have to work harder.

These are actually slightly degenerate examples of complete consistency, because in both cases one of the development relations implies the other, i.e. $\mathbf{ext} \subseteq \mathbf{conf}$ *and* $\mathbf{red} \subseteq \leq_{tr}$*. However, due to the common origins of the LOTOS refinement relations (i.e. trace/refusal semantics) such situations often arise for LOTOS.*

4.4 Balanced Consistency

Balanced consistency reflects the situation in which the specifications being checked are related to their common model by the same development relation; balanced consistency is written: $C_{dv}X_1X_2...X_n$ or $\overset{1..n}{C_{dv}}\, X_i$. The special case of binary balanced consistency, $C_{dv}X_1X_2$, is often written as $X_1\ C_{dv}\ X_2$. An example of a binary balanced consistency check is $Perm\, C_{\mathbf{red}}\, Comp$ from our running ODP example. We can characterise consistency in this restricted setting; the proof is trivial:-

Proposition 12
$C_{dv}X_1...X_n \iff lb(\{X_1, ..., X_n\}, dv) \neq \emptyset$.

Thus, in the balanced setting consistency checking degenerates to searching for lower bounds. Also, it should be clear that for balanced consistency lower bounds correspond to unifications, i.e. $\mathcal{U}_{dv}X_1...X_n = lb(\{X_1, ..., X_n\}, dv)$. In particular, the fact that the ordering of descriptions in balanced consistency is unimportant (which will be our next proposition) is reflected by the descriptions being interpreted as a set in lb.

4.4.1 Basic Properties

The following results are immediate:-

Proposition 13
(i) $C_{dv}X_1X_2...X_n = C_{dv}Y$ where Y is any possible permutation of $X_1...X_n$.
(ii) As a consequence of (i) C_{dv} is symmetric

This proposition states that the ordering of $X_1, ..., X_n$ in the argument list of C is not important in balanced consistency. The following results, which relate the characteristics of the development relation used to the induced balanced consistency, are also easily obtained:-

Proposition 14
(i) If dv is reflexive, then $X_1\ dv\ X_2 \implies X_1\ C_{dv}\ X_2$.
(ii) If dv is symmetric and transitive then $X_1\ C_{dv}\ X_2 \implies X_1\ dv\ X_2$.

Proof
(i) Assume $X_1\ dv\ X_2$; from reflexivity of dv we get X_1 is the required common development.
(ii) Assume $\exists X$ s.t. $X\ dv\ X_1\ \wedge\ X\ dv\ X_2$; then from symmetry $X_1\ dv\ X$ and from transitivity $X_1\ dv\ X_2$ as required. □

Corollary 3
If dv is an equivalence relation, then for all descriptions in ft, $dv = C_{dv}$.

LOTOS Illustration 3 *Corollary 3 can be used immediately to obtain the following results for basic LOTOS:-*

Proposition 15
(i) $C_{\mathbf{te}} = \mathbf{te}$; (ii) $C_{\sim} = \sim$; and (iii) $C_{\approx} = \approx$.

This result completes our relating of basic LOTOS development relations to consistency and along with propositions 7, 9 and 10, shows that all the main basic LOTOS development relations can be embraced by our interpretation of consistency.

In addition, this relationship between equivalence and consistency can be used to give us:

Proposition 16
$C_{\sim} \subset C_{\approx} \subset C_{\mathbf{te}} \subset C_{\mathbf{xcs}}$.

Proof
$C_{\sim} \subset C_{\approx} \subset C_{\mathbf{te}}$ come directly from propositions 15 and 2. In addition, from proposition 2 we can determine that $C_{\mathbf{te}} \subset \mathbf{cs}$ and since proposition 10 gives us $\mathbf{cs} = C_{\mathbf{xcs}}$ we are done. □

ODP Illustration 2 *However, as might be expected, equivalence based balanced consistency yields a check that is generally overly restrictive. For example, considering again our ODP viewpoints illustration, Eng is already related by the equivalence* **te***, the reason for this being that the engineering viewpoint specifies the implementation mechanisms of the system and is thus, observationally indistinguishable from the implementation itself. However, if we were to impose the same development relation on other viewpoints we would prevent any implementation freedom in the viewpoint specification process. For example, the consistency check, $Comp\, C_{\mathbf{te}}\, Eng$ certainly does not hold, because Comp contains a lot of non-determinism (and hence implementation freedom) which is not reflected in Eng.*

4.4.2 Complete Balanced Consistency

We would like to characterise complete consistency in the balanced setting. The following is very straightforward:

Proposition 17

Given $ds \subseteq DES_{ft} \ \wedge \ dv \in DEV_{ft},\ lb(ds, dv) \neq \emptyset \iff \forall X_1, ..., X_n \in ds,\ \overset{1..n}{C_{dv}}\, X_i$ *holds.*

i.e. if all subsets of ds have a lower bound then all specifications are consistent by dv.

As suggested by proposition 11 a sufficient condition for complete consistency is that a terminal element exists. Since for balanced consistency we only have one development relation this terminal element is a bottom element for the one development ordering.

Proposition 18

$t(ds, dv) \neq \emptyset \implies \forall X_1, .., X_n \in ds,\ C_{dv}(X_1, ..., X_n)$ *holds.*

LOTOS Illustration 4 *The following result is a simple instantiation of our general theory for the LOTOS development relations.*

Proposition 19

$\forall P_1,\ P_2 \in DES_{basicLOTOS}$, *(i)* $P_1\ C_{\leq_{tr}}\ P_2$; *(ii)* $P_1\ C_{\mathbf{ext}}\ P_2$; *and* *(iii)* $P_1\ C_{\mathbf{conf}}\ P_2$.

Proof

We have that, up to equivalence:

(i) $t(DES_{basicLOTOS}, \leq_{tr}) = \{stop\}$;

(ii) $t(DES_{basicLOTOS}, \mathbf{ext}) = \{\omega\}$ *(ω was introduced in LOTOS illustration 2); and*

(iii) $t(DES_{basicLOTOS}, \mathbf{conf}) = \{\omega\}$. □

Corollary 4

$C_{\leq_{tr}} = C_{\mathbf{ext}} = C_{\mathbf{conf}} = true$

where true is the universal relation over $DES_{basicLOTOS}$.

Thus, these instantiations of consistency are very weak and are unable to distinguish any specifications. In other words, when $\leq_{tr}$, **ext** *or* **conf** *is the chosen development relation, there is no need for a consistency check.*

We illustrate the second case, (ii), of proposition 19 with some examples in figure 6. The following properties hold:-

$$P \in \mathcal{U}_{\mathbf{ext}}(P_1, P_2), \quad Q \in \mathcal{U}_{\mathbf{ext}}(Q_1, Q_2), \quad Q' \notin \mathcal{U}_{\mathbf{ext}}(Q_1, Q_2), \quad R \in \mathcal{U}_{\mathbf{ext}}(R_1, R_2) \ \text{ and } \ R' \notin \mathcal{U}_{\mathbf{ext}}(R_1, R_2)$$

Notice that (b) shows that $\mathcal{U}_{\mathbf{ext}}$ *must not introduce new non-determinism, e.g. Q is a unification, but Q' is not as it may refuse either d or e after performing a and Q_1 cannot refuse d after a and Q_2 cannot refuse e after a. Additionally, (c) shows that unification may limit non-determinism. Specifically, R is a unification, but R' is not as it can refuse everything after the empty trace, while R_2 must offer either a or c.*

However, other instantiations of consistency are distinguishing:

Proposition 20

$C_{\mathbf{red}} \subset true$.

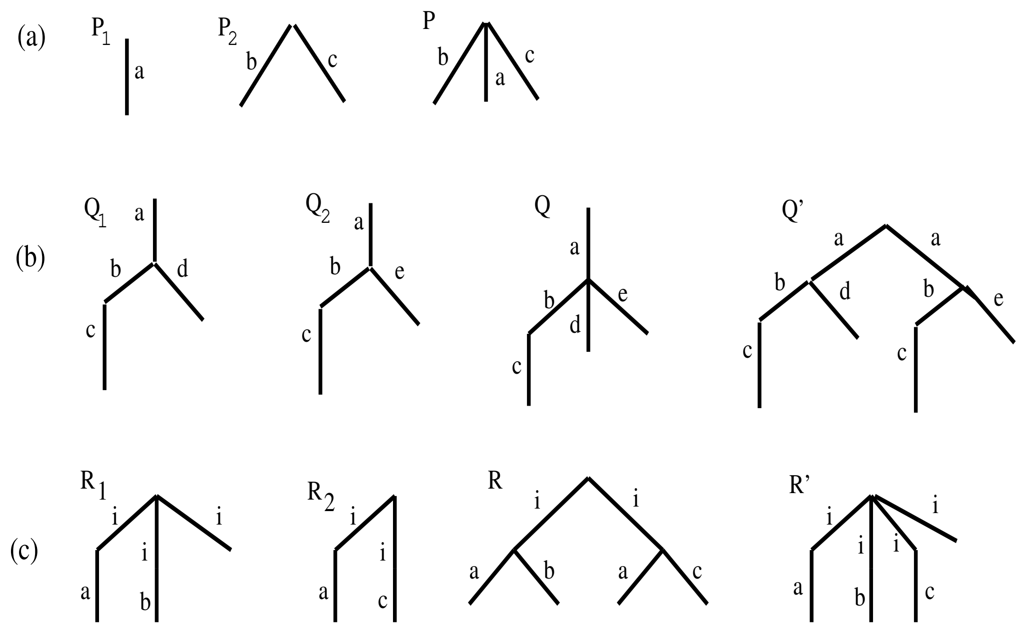

Figure 6: Unification by extension examples

Proof
We simply need to exhibit a pair of specifications that are not consistent. $P_1 := a; stop$ *and* $P_2 := b; stop$ *which were discussed in LOTOS illustration 2 are sufficient.* □

Thus, $C_{\mathbf{red}}$ *is strictly stronger than* $C_{\leq_{tr}}$, $C_{\mathbf{ext}}$ *and* $C_{\mathbf{conf}}$. *Consider the examples in figure 7. The following properties hold:-*

$$\mathcal{U}_{\mathbf{red}}(P_1, P_2) = \emptyset, \quad Q \in \mathcal{U}_{\mathbf{red}}(Q_1, Q_2) \quad \text{and } R_2 \in \mathcal{U}_{\mathbf{red}}(R_1, R_2).$$

ODP Illustration 3 *We can also illustrate these classes of balanced binary consistency in terms of our running ODP viewpoints example. If, for example, we consider the two viewpoint specifications Comp and Eng and relate them using first* $C_{\mathbf{conf}}$ *and then* $C_{\leq_{tr}}$ *(which are "parts" of the full binary consistency check that we are interested in between these viewpoints, which is* $C_{\mathbf{red,te}}$[6]*) then by proposition 19 (i) and (iii) we have automatically that,*

$Comp\, C_{\mathbf{conf}}\, Eng$ *and* $Comp\, C_{\leq_{tr}}\, Eng$

However, the use here of the term "parts" is somewhat loaded because as proposition 20 suggests even though for arbitrary P_1 *and* P_2 *we automatically have that,*

$P_1\, C_{\mathbf{conf}}\, P_2$ *and* $P_1\, C_{\leq_{tr}}\, P_2$

it does not follow that $P_1\, C_{\mathbf{red}}\, P_2$ *even though the relations underlying* **red** *are* **conf** *and* $\leq_{tr}$. *The problem is that the non-empty unification sets* $\mathcal{U}_{\mathbf{conf}}(P_1, P_2)$ *and* $\mathcal{U}_{\leq_{tr}}(P_1, P_2)$ *that enable* $P_1\, C_{\mathbf{conf}}\, P_2$ *respectively* $P_1\, C_{\leq_{tr}}\, P_2$ *to hold, may not intersect, i.e.* P_1 *and* P_2 *may have common implementations by* **conf** *and* $\leq_{tr}$ *respectively, but these implementations may not coincide.*

In fact, this very situation does arise with our ODP viewpoints example. As just stated $Comp\, C_{\mathbf{conf}}\, Eng$ *and* $Comp\, C_{\leq_{tr}}\, Eng$ *hold automatically, however,* $Comp\, C_{\mathbf{red}}\, Eng$ *actually fails to hold*[7]*, which we can justify as follows:*

[6] Remember, **red**=**conf**$\cap \leq_{tr}$ and **te**=**red**$\cap$ **red**$^{-1}$.

[7] Furthermore, checking *Eng* according to **red** is enough to ensure **te** since because *Eng* is completely deterministic any reduction of *Eng* will also be testing equivalent to *Eng*. Consequently our analysis here will show us exactly the relationship we are interested in for *Eng*.

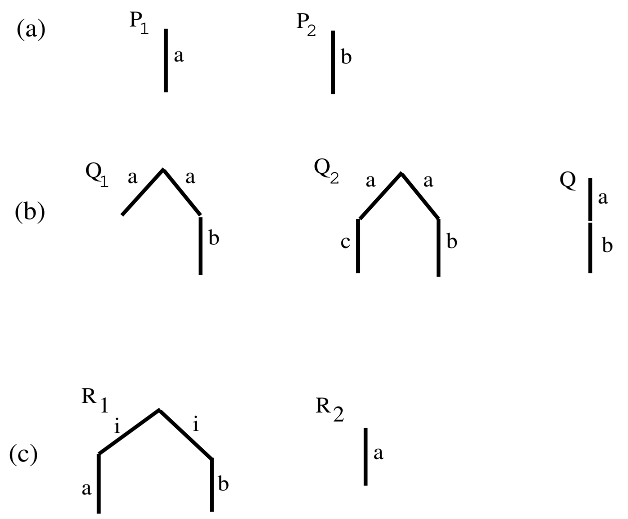

Figure 7: Unification by reduction, examples

- **Eng** *can perform the trace request_id request_id', say, i.e. it can perform two requests before performing a transmission. This possibility arises from the concurreny in Eng whereby the request handler and IO handler evolve concurrently and the request handler can receive a second request before the first request has been serviced by the IO handler. In fact, since Eng is completely deterministic the situation is somewhat stronger as Eng cannot refuse the second request. Thus, all reductions of Eng must be able to perform the trace request_id request_id'.*
- **Comp** *cannot perform the trace request_id request_id'. This is because each request is followed by a transmission before the next request is offered. Furthermore, since* **red** *does not allow new traces to be added the trace request_id request_id' cannot be performed by any reduction of Comp.*
- *Thus, any reduction of Eng must be able to perform request_id request_id', however no reduction of Comp can perform this trace!*

4.4.3 Bringing Together the LOTOS Balanced Consistency Relations

Because balanced consistency is relatively well behaved (compared to the other classes of consistency) we can give a complete characterisation of binary balanced consistency for basic LOTOS. This subsection performs this task by characterising the remaining basic LOTOS binary balanced consistency relation $C_{\mathbf{cs}}$ and summarising the complete set of basic LOTOS balanced consistency relations.

If we can show that $C_{\mathbf{cs}}$ is stronger than $C_{\mathbf{red}}$ then we will have an upper bound on the strength of $C_{\mathbf{cs}}$. We need a lemma first.

Lemma 1
$P_1 \; C_{\mathbf{cs}} \; P_2 \implies \forall P \in \mathcal{U}_{\mathbf{cs}} P_1 P_2, \; \forall \sigma \in Tr(P) \cap Tr(P_1) \cap Tr(P_2), \; Ref(P,\sigma) = Ref(P_1,\sigma) = Ref(P_2,\sigma).$

Proof
Assume $\sigma \in Tr(P) \cap Tr(P_1) \cap Tr(P_2)$ then (as a direct consequence of the definition of **cs**) from P **cs** P_1 we can get $Ref(P,\sigma) = Ref(P_1,\sigma)$ and from P **cs** P_2 we get $Ref(P,\sigma) = Ref(P_2,\sigma)$ and the result follows directly. $\square$

Proposition 21
$C_{\mathbf{red}} \not\subseteq C_{\mathbf{cs}}$

Proof
We show that P_1 and P_2 exist such that $P_1 \; C_{\mathbf{red}} \; P_2$, but $\neg(P_1 \; C_{\mathbf{cs}} \; P_2)$. Consider $P_1 := a; b; stop[]a; stop$ and $P_2 := a; b; stop$. Now $P_1 \; C_{\mathbf{red}} \; P_2$, because P_2 can act as the required common reduction.

We argue by contradiction that $\neg(P_1\ C_{\mathbf{cs}}\ P_2)$. So, assume P is such that P **cs** P_1 and P **cs** P_2. Firstly, P must be able to perform the trace a, because if $a \notin Tr(P)$ then $\{a\} \subseteq Ref(P,\epsilon)$, but since $\{a\} \not\subseteq Ref(P_2,\epsilon)$ this implies that $Ref(P,\epsilon) \not\subseteq Ref(P_2,\epsilon)$ and $\neg(P\ \mathbf{conf}\ P_2)$.

So, we assume $a \in Tr(P)$, hence $a \in Tr(P) \cap Tr(P_1) \cap Tr(P_2)$ and we can apply lemma 1 which implies that $Ref(P,a) = Ref(P_1,a) = Ref(P_2,a)$. But this cannot be the case as $\{b\} \subseteq Ref(P_1,a)$ and $\{b\} \not\subseteq Ref(P_2,a)$ so $Ref(P_1,a) \neq Ref(P_2,a)$; which gives us the required contradiction and implies that such a P does not exist. □

So, $C_{\mathbf{cs}}$ is not weaker than $C_{\mathbf{red}}$. Using the following small result we will be able to further clarify the relationship between $C_{\mathbf{cs}}$ and $C_{\mathbf{red}}$.

Lemma 2
P is deterministic (in the usual sense) $\Longrightarrow$ $(\forall \sigma \in Tr(P),\ a \in out(P,\sigma) \iff \neg\exists X \in Ref(P,\sigma)\ .\ a \in X)$.

Proof
Standard from theory of LOTOS. □

This result states that for a deterministic process an action cannot be both offered and refused. Thus, a fully deterministic process is characterised by its traces only.

Proposition 22
$C_{\mathbf{cs}} \subseteq C_{\mathbf{red}}$.

Proof
Assume $P_1\ C_{\mathbf{cs}}\ P_2$, i.e. $\exists P\ s.t.\ P\ \mathbf{cs}\ P_1\ \wedge\ P\ \mathbf{cs}\ P_2$. Now construct P' as the fully deterministic process characterised by:
$Tr(P') = Tr(P) \cap Tr(P_1) \cap Tr(P_2)$
Noting from this construction that $Tr(P') \subseteq Tr(P_1), Tr(P_2)$ and from lemma 1 that $\forall \sigma \in Tr(P'),\ Ref(P,\sigma) = Ref(P_1,\sigma) = Ref(P_2,\sigma)$. In order to show that P' is the required common reduction of P_1 and P_2 we need to show that $\forall \sigma \in Tr(P'),\ Ref(P',\sigma) \subseteq Ref(P_1,\sigma), Ref(P_2,\sigma)$. This is enough because any $\sigma' \in Tr(P_1) \cup Tr(P_2)\ s.t.\ \sigma' \notin Tr(P')$ will give $Ref(P',\sigma') = \emptyset$, which trivially gives us the required refusals relationship.

We argue by contradiction that $\forall \sigma \in Tr(P'),\ Ref(P',\sigma) \subseteq Ref(P_1,\sigma), Ref(P_2,\sigma)$. So, assume $\exists \sigma \in Tr(P')\ s.t.\ Ref(P',\sigma) \supset (Ref(P_1,\sigma) = Ref(P_2,\sigma) = Ref(P,\sigma))$. Thus, $\exists \{a\} \not\subseteq (Ref(P_1,\sigma) = Ref(P_2,\sigma) = Ref(P,\sigma))$, such that $\{a\} \in Ref(P',\sigma)$. From here we can use lemma 2 to get $a \notin out(P',\sigma)$, but it must also be the case that $a \in out(P_1,\sigma), out(P_2,\sigma), out(P,\sigma)$ and thus we have a contradiction as the trace $\sigma.a$ is in $Tr(P) \cap Tr(P_1) \cap Tr(P_2)$. So, it must be the case that $Ref(P',\sigma) \subseteq Ref(P_1,\sigma), Ref(P_2,\sigma)$ and P' **red** P_1 and P' **red** P_2 as required. □

Corollary 5
$C_{\mathbf{cs}} \subset C_{\mathbf{red}}$.

Proof
From propositions 21 and 22.

Thus, $C_{\mathbf{cs}}$ is strictly stronger than $C_{\mathbf{red}}$. In addition, we can show that $C_{\mathbf{cs}}$ is strictly weaker than $C_{\mathbf{xcs}}$, as follows:

Proposition 23
$C_{\mathbf{xcs}} \subset C_{\mathbf{cs}}$.

Proof
Firstly, proposition 10 gives us $\mathbf{cs}= C_{\mathbf{xcs}}$. Then we can argue as follows to give us $\mathbf{cs}\subset C_{\mathbf{cs}}$:-

Firstly, $P_1\ \mathbf{cs}\ P_2 \implies P_1\ C_{\mathbf{cs}}\ P_2$, follows immediately from the reflexivity of **cs**, i.e. either of P_1 or P_2 could act as the required common **cs**-development.

In addition, we can provide a counterexample to show that, $C_{\mathbf{cs}} \not\subseteq \mathbf{cs}$. Consider, $P_1 := i;a;stop[]b;c;stop$, $P_2 := i;a;stop[]b;stop$ and $P := i;a;stop$. Now, P **cs** P_1 and P **cs** P_2, but $\neg(P_1\ \mathbf{cs}\ P_2)$. This is because $\neg(P_2\ \mathbf{conf}\ P_1)$ as P_2 refuses c after the trace b, but P_1 cannot refuse c after the same trace. □

The relationship between the different interpretations of consistency are shown in figure 8. These instantiations present us with a number of possible interpretations of consistency in LOTOS. This situation reflects our view

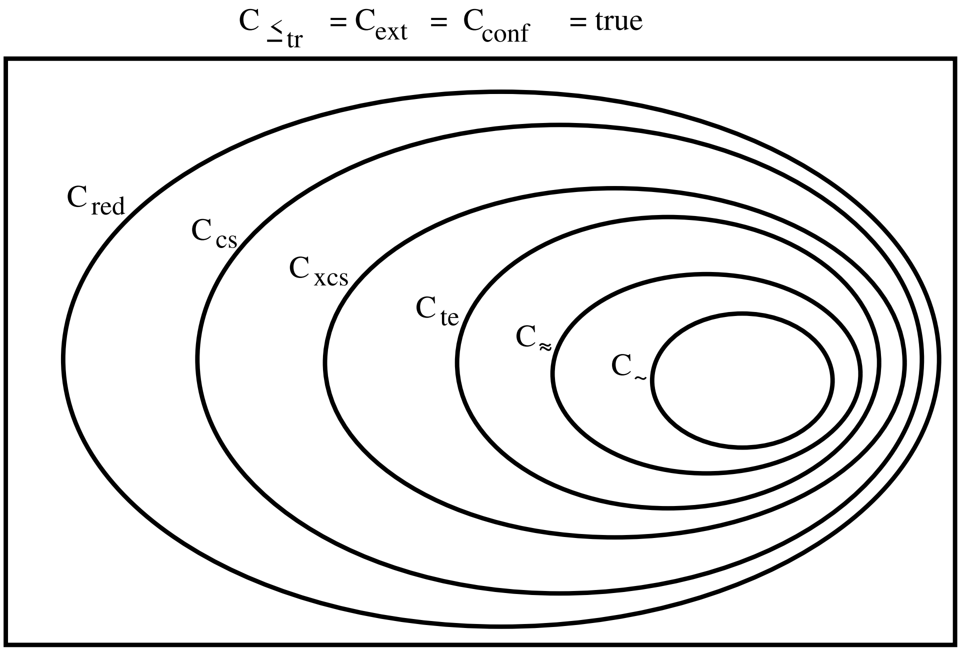

Figure 8: LOTOS Binary Consistency Relations

that consistency checking must be performed selectively, as was discussed in some depth in [15]. In particular, it is inappropriate to view consistency checking as a single mechanism which can be applied to any pair of specifications. For example, it would be inappropriate to check two specifications which express exactly corresponding functionality with $C_{\mathbf{ext}}$.

4.4.4 Bringing Together the ODP Example

We can now bring together the consistency checking relationships that we have highlighted concerning our ODP viewpoints example. We have the following pair-wise consistency relationships between viewpoints:-

- $Perm\, C_{\mathbf{red},\mathbf{ext}}\, Obl1$ holds. For example, *Obl1* is a unification.
- $Perm\, C_{\mathbf{red},\leq_{tr}}\, Obl2$ holds. For example, *Obl2* is a unification.
- $Perm\, C_{\mathbf{red}}\, Comp$ holds. For example, with $F' = F - \{\mathit{request}\}$, the process,

  $$\begin{array}{lcl} C & := & \textit{i; request?x:ID'; (choice } b{\in}F' \textit{ [] i; b!x; C)} \\ & & \textit{[] (choice } b{\in}F' \textit{ [] i; b!def; C)} \end{array}$$

  is a reduction of *Perm* and of *Comp*. Thus, the *Perm* viewpoint has specialized the behaviour of *Comp* to only offer audio and video transmission.
- $Perm\, C_{\mathbf{red},\mathbf{te}}\, Eng$ holds. For example, *Eng* is a unification.
- $Obl1\, C_{\mathbf{ext},\leq_{tr}}\, Obl2$ holds. For example, *Obl1* is a unification.
- $Obl1\, C_{\mathbf{ext},\mathbf{red}}\, Comp$ holds. For example, the process,

  $$D \;\; := \;\; \textit{i; request?x:ID'; (choice } b{\in}H \textit{ [] i; b!x; D)}$$

  is an extension of *Obl1* and a reduction of *Comp*. Thus, the obligation has specialised the computational viewpoint by preventing it from transmitting with the default identifier immediately.

- $Obl1\, C_{\mathbf{ext},\mathbf{te}}\, Eng$ holds. For example, *Eng* is a unification.
- $Obl2\, C_{\leq_{tr},\mathbf{red}}\, Comp$ holds. For example, the process,

$$E \quad := \quad i;\ request?x{:}ID';\ (choice\ b{\in}F'\ []\ i;\ b!x;\ E)$$

  is a trace refinement of *Obl2* and a reduction of *Comp*. Thus, *Obl2* specialises *Comp*, by preventing it from transmitting with the default identifier and to ensure it only transmits on video and audio links.
- $Obl2\, C_{\leq_{tr},\mathbf{te}}\, Eng$ does not hold for similar reasons to those that we discussed in section 4.4.2 for why $Comp\, C_{\mathbf{red}}\, Eng$ does not hold, i.e. *Eng* cannot refuse a second request after performing an initial request, while no trace refinement of *Obl2* can perform consecutive requests.
- $Comp\, C_{\mathbf{red},\mathbf{te}}\, Eng$ does not hold for the reason identified in section 4.4.2 and just highlighted again.

As previously discussed, the basic problem with the engineering specification is that it allows a second request to be made before the previous request has been matched to a transmission. The inconsistency this yields with regard to *Obl2* and *Comp* can be resolved by adding another synchronisation between the two components of the engineering specification (the same channel can be used for both):

$$\begin{aligned} NewEng \quad &:= \quad hide\ channel\ in\ RH\ |[channel]|\ IOH \\ RH \quad &:= \quad request?x{:}ID';\ channel!x;\ channel!x;\ RH \\ IOH \quad &:= \quad channel?x{:}ID';\ (transA!x;\ channel!x;\ IOH\ []\ transV!x;\ channel!x;\ IOH) \end{aligned}$$

with such a synchronisation in place the request handler will refuse the second request until the previous request has been matched to a transmission. This new engineering specification is consistent with all the other viewpoints, i.e.

$$Perm\, C_{\mathbf{red},\mathbf{te}}\, NewEng,\ Obl1\, C_{\mathbf{ext},\mathbf{te}}\, NewEng,\ Obl2\, C_{\leq_{tr},\mathbf{te}}\, NewEng,\ \text{and}\ Comp\, C_{\mathbf{red},\mathbf{te}}\, NewEng,$$

all hold, with in each case *NewEng* itself being an example unification.

The reader should notice that what has happened here is a nice example of what, in the introduction to this paper, we called ***fluid system development***, i.e. the viewpoints specifications have evolved independently and then a consistency check has revealed an inconsistency between the viewpoints which has prompted adaptation of a particular viewpoint, here the engineering viewpoint.

However, there is still one important remaining issue with this example - are the viewpoints "globally" consistent? Notice that we have only checked pairwise between viewpoints, but does such pairwise consistency ensure global consistency? This is one of the issues that we will consider in section 5.

4.5 Inter Language Consistency

The basic definition of consistency that we presented in section 4.1 enables descriptions in different formal techniques to be related and thus supports inter language consistency. In this circumstance the unification sought would be a description in a common notation, e.g. a semantic notation that can represent the formal techniques of both the original descriptions. An inter language consistency check (assuming type correctness) between descriptions in n formal techniques, $ft_1, ..., ft_n$, will typically have the following form:

$$\begin{aligned} &C(dv_1 \circ [\![\,]\!]_1, X_1)....(dv_n \circ [\![\,]\!]_n, X_n) \text{ where,} \\ &X_i \in DES_{ft_i}\ \ ;\ \ p_r([\![\,]\!]_i) \subseteq DES_{ft_i}\ \ ;\ \ p_l([\![\,]\!]_i) \subseteq DES_{ft}\ \ ;\ \text{ and }\ \ dv_i \in intraDEV_{ft}. \end{aligned}$$

Thus, ft is the common notation, i.e. a unification of $X_1, .., X_n$ would be in ft. Each description is related to the common model by a semantic map, $[\![\,]\!]_i$, which, in effect, translates into the common notation (this is the realisation of the ODP notion of translation, see figure 3) and then an intra language development relation, dv_i, is applied in the common notation.

Illustration 3 *[25] defines a translation of LOTOS into Z, which we will denote* [8]*:*

[8] We adopt the unconventional function typing notation, $f\ :\ S' \leftarrow S$, in order that functional relations reflect the order we have adopted for development relations, the order of which has been chosen to reflect the standard orientation of LOTOS relations

$$[\![\,]\!]_{Z<L} \;:\; DES_Z \leftarrow DES_{LOTOS}$$

A typical consistency check that we can perform with this semantic map is:

$$C(\sqsubseteq \circ [\![\,]\!]_{Z<L}, P)(\sqsubseteq, S) \quad \textit{for } \sqsubseteq \in intraDEV_Z,\ P \in DES_{LOTOS} \textit{ and } S \in DES_Z$$

Notice in this inter language consistency check the formal notation Z is used as the common notation.

The translation $[\![\,]\!]_{Z<L}$ probably seems an unlikely construction to some readers. So, we will say more about this map and its theoretical justification. The theoretical foundations for $[\![\,]\!]_{Z<L}$ are two semantics, $[\![\,]\!]_L \in SEM_{LOTOS}$ and $[\![\,]\!]_Z \in SEM_Z$, which are typed as follows:

$$[\![\,]\!]_L \;:\; DES_{ETS} \leftarrow DES_{LOTOS} \quad \textit{and} \quad [\![\,]\!]_Z \;:\; DES_{ETS} \leftarrow DES_Z$$

where ETS is an extended transition system semantic notation; the extension ensures that the transition system generated is finite state [53]. The semantic map $[\![\,]\!]_L$ is relatively standard, apart from the extension mechanism, for details see [53] [25]. However, the mapping $[\![\,]\!]_Z$ is more unusual. The basics of the mapping are as follows (details can be found in [25]):

- *Z operations become actions in the transition system.*
- *The order in which actions are offered is determined by analysing how Z operations become enabled according to pre and postconditions of operation schemas.*
- *Z data state is handled symbolically in the ETS, in particular, transitions have associated symbolic data effects.*
- *The initial state schema of Z specifications is mapped to the initial state of the ETS.*

$[\![\,]\!]_L$ and $[\![\,]\!]_Z$ are taken as given semantics in defining $[\![\,]\!]_{Z<L}$. The correctness of the translation is guaranteed because $[\![\,]\!]_{Z<L}$ satisfies:

$$\forall P \in DES_{LOTOS} \;.\; [\![\,P\,]\!]_L \approx_{ETS} [\![\,[\![\,P\,]\!]_{Z<L}\,]\!]_Z$$

where $\approx_{ETS}$ is weak bisimulation on Extended Transition Systems. Thus, translating any $LOTOS$ process into Z (using $[\![\,]\!]_{Z<L}$) and then taking the ETS semantics (using $[\![\,]\!]_Z$) is observationally equivalent to taking the ETS semantics (using $[\![\,]\!]_L$) of the LOTOS process. This is a strong justification for $[\![\,]\!]_{Z<L}$ as it ensures that (according to the given semantics) translation preserves a strong notion of behavioural equivalence.

Our interpretation of consistency prompts the question: what constitutes a reasonable cross language development relation. Specifically, we would actually like to know that the cross language development relation reflects, in some reasonable sense, a development relation from the source language. Although it is not correctly typed, conceptually, a specifier would like to make consistency checks such as:

$$C(\mathbf{red}, P)(\sqsubseteq, S)$$

i.e. the specifier wants to know that an implementation can be found which is a reduction of P and a Z refinement of S. We would like to replace this check with a type correct check such as

$$C(dv \circ [\![\,]\!], P)(\sqsubseteq, S)$$

where dv, in some sense, corresponds to **red**.

Thus, we would like to relate the development relations of a particular language to the cross language development relations that we use. In order to do this we introduce the notion of development relations in different notations *correlating* under certain conditions.

Definition 17 (Correlation between relations) *Given formal techniques $ft,\ ft' \in FT$, $dv \in DEV_{ft}$ and a semantic map $[\![\,]\!] : ft' \leftarrow ft$ (i.e. $[\![\,]\!]$ translates from ft to ft') then dv' correlates to dv, written $dv' \leftarrow dv$ iff $\forall X_1, X_2 \in DES_{ft}, X_1\ dv\ X_2 \iff [\![X_1]\!]\ dv'\ [\![X_2]\!]$.*

The left to right implication, $\forall X_1, X_2 \in DES_{ft}, X_1 \; dv \; X_2 \implies [\![X_1]\!] \; dv' \; [\![X_2]\!]$, ensures that any development in ft has a corresponding development in ft'. The right to left implication, $\forall X_1, X_2 \in DES_{ft}, X_1 \; dv \; X_2 \impliedby [\![X_1]\!] \; dv' \; [\![X_2]\!]$, ensures that ft' does not add new developments. Thus, it prevents unifications being found in ft' which do not correspond to developments in ft.

In fact, a body of work now exists on relating behavioural, e.g. LOTOS and state based, e.g. Z, development relations and this work has been summarised and extended in [12]. In particular, behavioural relations which correlate to state based relations (when state based specifications are interpreted behaviourally according to mappings like, $[\![\,]\!]_Z$) can be located. For example, [12] show that the most common interpretation of Z refinement (downward simulation - to give it its precise name) correlates to ready simulation testing over the induced labelled transition system. These results give an important link between the state based and behavioural worlds which make feasible inter-language consistency checking.

In addition, a substantial case study in consistency checking has been presented in [5]. This case study concerns the signalling system no. 7 protocol [51]. The protocol is described from multiple viewpoints and then the viewpoints are checked for consistency. Importantly, since both LOTOS and Z are used in these viewpoint specifications, inter lanuage consistency checks are employed in the style of those just discussed.

A similar translation from LOTOS to Object-Z has also been defined [21, 22]. This is a direct translation which is also structure preserving in that LOTOS syntactic operators are mapped directly to equivalent Object-Z syntactic operators. The common semantic model that verifies this translation is the standard (labelled transition) semantics of Object-Z into which the semantics of LOTOS is embedded.

4.6 Summary and Discussion

This section has highlighted a general interpretation of consistency, identified the basic properties of the definition and located a number of specific classes of consistency. Our interpretation of consistency, C, meets the requirements for a definition of consistency that we highlighted earlier, in the following ways:

- Different development relations can be instantiated which are appropriate both to different FDTs and to assessing different forms of consistency.
- Both intra and inter language consistency are incorporated.
- Consistency checking between an arbitrary number of descriptions can be supported and checked according to a list of development relations. Binary consistency is just a special case of this global consistency.
- Both balanced and unbalanced consistency are incorporated.

In its fully general form it is very difficult to characterise properties of our interpretation of consistency, it is too general. However, by restricting to particular classes of consistency, characterisations can be investigated. We have located the following classes: binary consistency, complete consistency, balanced consistency and inter language consistency.

Throughout we have illustrated our general notion of consistency using LOTOS. In particular, section 4.4.3 contained a complete characterisation of binary balanced consistency for basic LOTOS.

It should also be pointed out that elsewhere we have assessed the generality of our definition of consistency by showing that other interpretations of consistency can be embraced by our definition. In particular, [16] has shown that the three previously proposed alternatives for ODP consistency can be embraced by our interpretation. These three alternatives are consistency in terms of locating a common conformant implementation, consistency according to behavioural compatibility and consistency as freedom from logical contradiction. Thus, there is strong evidence that the interpretation of consistency described here is general enough to fully support consistency checking of ODP viewpoints.

5 Basic Strategies for Consistency Checking

So far in this paper we have viewed a group of descriptions $X_1, X_2, ..., X_n$ as consistent if their set of possible unifications $\overset{1..n}{\mathcal{U}}(dv_i, X_i)$ is non-empty. However unification sets can be very large and even infinite. Thus, if a system development trajectory is to be provided for viewpoint models it is important that the choice of possible unifications is reduced. In fact, we would like to select just one description from the set of unifications. This

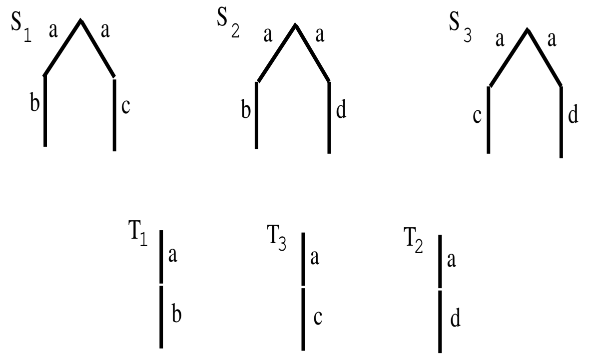

Figure 9: Pairwise Consistency Counterexample

would enable an incremental consistency checking strategy in which a group of viewpoints are unified and then this unification is further composed with another group of viewpoints.

The advantages of such incremental consistency checking strategies are that they do not force the involvement of all viewpoints in every consistency check. In particular, it may be possible to incrementally correct inconsistencies. In addition, such an approach will aid maintaining structure when unifying. One of the main problems with unification algorithms is that the generated unification is almost certain to be devoid of high level specification structure [49] (e.g. the LOTOS parallel composition operator, |[ ]|, would be expanded out). This is a big problem if the unification is to be further developed. It is very unlikely that a single unification of a large group of viewpoints will be able to reconcile the structure of all the views, however, an incremental focus of restructuring may be possible.

We must then consider how to obtain global consistency from a series of non-global (probably binary) consistency checks and unifications. This topic is what we consider now. Transitivity and reflexivity of development will be assumed in this work. These are restrictive assumptions that effectively rule out inter language consistency checking, but are required in order to develop a body of theory. In particular, since the work is investigating incremental consistency checking it seems reasonable to assume transitivity of development.

This section considers basic strategies for consistency checking. General formats for binary consistency checking are considered in section 5.1 and the central issue of least developed unification is discussed in 5.2. These basic strategies will be used in later sections when we consider the properties required in order to realise a binary consistency checking strategy.

5.1 Binary Consistency Checking Strategies

We would like to obtain global consistency through a series of binary consistency checks. Firstly, we must observe that we cannot assert consistency between three or more descriptions by performing a series of pairwise consistency checks, where pairwise consistency means binary consistency holds between all possible pairings of descriptions. As an illustration consider the three specifications, S_1, S_2 and S_3 shown in figure 9. These are balanced pairwise consistent by reduction, since, $T_1 \in \mathcal{U}_{\mathbf{red}}(S_1, S_2)$, $T_2 \in \mathcal{U}_{\mathbf{red}}(S_2, S_3)$ and $T_3 \in \mathcal{U}_{\mathbf{red}}(S_1, S_3)$; but, they are not globally consistent, i.e. $\neg\mathcal{C}_{\mathbf{red}}(S_1, S_2, S_3)$.

The problem is that pairwise consistency only requires the existence of a common development for each of the constituent binary checks. Thus, many binary consistency results may exist each of which focuses on a different common development. This is not sufficient to induce "global" consistency which requires the existence of a single common development.

We can illustrate this issue with our ODP example. In fact, the viewpoint specifications we have given, *Perm*, *Obl1*, *Obl2*, *Comp* and *NewEng* are both exhaustively pairwise consistent and globally consistent. The latter can be seen from the fact that *NewEng* is a unification of all the viewpoints. However, we could imagine a slightly different example where the problem would arise.

Imagine, for example, that in addition to the transmission media to be found in set H, which were,

$$H = \{\, transA\,,\ transV\,,\ transT_1\,, ...,\ transT_n\,\}$$

there was a further media type, say,

$$transX$$

which is not used in *Comp* and we have the two permissions:

PermA := (choice b∈ { request, transA, transX} [] i; b?x:ID; PermA) [] i; stop

and

PermB := (choice b∈ { request, transV, transX} [] i; b?x:ID; PermB) [] i; stop

then, $PermA\, C_{\mathbf{red}}\, PermB$, since,

P1 := (choice b∈ { request, transX} [] i; b?x:ID; P1) [] i; stop

is a unification. Also, $PermA\, C_{\mathbf{red}}\, Comp$ holds because,

$$\begin{array}{lll} P2 & := & i;\ request?x{:}ID';\ transA!x;\ P2 \\ & & [\,]\ i;\ transA!def;\ P2 \end{array}$$

is a unification. Finally, $PermB\, C_{\mathbf{red}}\, Comp$ holds because,

$$\begin{array}{lll} P3 & := & i;\ request?x{:}ID';\ transV!x;\ P3 \\ & & [\,]\ i;\ transV!def;\ P3 \end{array}$$

is a unification. However, $C_{\mathbf{red}}(PermA, PermB, Comp)$ does not hold, because the intersection of the sets of transmission types that they can perform is empty.

However, a combination of binary consistency checks and binary unification of the form shown in figure 10 should intuitively allow us to deduce global consistency, i.e. X_1 and X_2 are checked for consistency, then a unification of X_1 and X_2 is obtained, which is checked for consistency against X_3, then a unification of X_3 and the previous unification is performed. This process is continued through the n viewpoint descriptions. Thus, the base case is a binary consistency check and then repeated unification and binary consistency checks are performed against the next description. Of course, this is just one possible sequence of binary consistency checks. We would like to obtain full associativity results which support any appropriate incremental consistency checking strategy.

A more precise depiction of such an incremental consistency checking strategy is presented in figure 11 which highlights the $n = 4$ case. The binary unification function is denoted:

$$U \ : \ (DEV \times DES) \times (DEV \times DES) \rightarrow DES$$

i.e. two pairs (each comprising a development relation and a description) are taken and a description is returned.

So, each step in the algorithm considers a unification using the binary unification function U. The ith step is satisfied if a unification Y_i is generated by U which can be used to satisfy the $i + 1$st step. Importantly such an approach generalises corresponding balanced consistency checking strategies by taking the intersection of development relations; this ensures that the final unification (using transitivity of development) is a development (by appropriate development relations) of all the original descriptions.

However, we must be careful over the choice of unification. Specifically, an arbitrary description from the unification set will not always be satisfactory. We highlight such a situation in the following illustration.

Illustration 4 *Consider the three basic LOTOS specifications,*

$P_1 := a; b; stop[]a; c; stop$, $P_2 := a; b; stop[]a; c; stop[]a; d; stop$ *and* $P_3 := a; c; stop$

Further consider the consistency check $C_{\mathbf{red}} P_1 P_2 P_3$. *The three specifications are consistent by reduction since* P_3 *is a reduction of all three specifications. However, if we attempt a binary consistency checking algorithm and started with* P_1 *and* P_2 *we may choose as the unification of these two the process* $P := a; b; stop$, *and* $C_{\mathbf{red}} P P_3$ *does not hold.*

We can also demonstrate the problem in the context of our ODP viewpoints illustration. For example, the process stop is in the unification set of Perm and Obl2. However, stop cannot be related to any of the other viewpoints by **red** *(=*$\mathbf{red} \cap \leq_{tr}$*, i.e. the intersection of Perm and Obl2's development relations). Thus, if stop is taken as the unification of Perm and Obl2 then the incremental global consistency check will be compromised.*

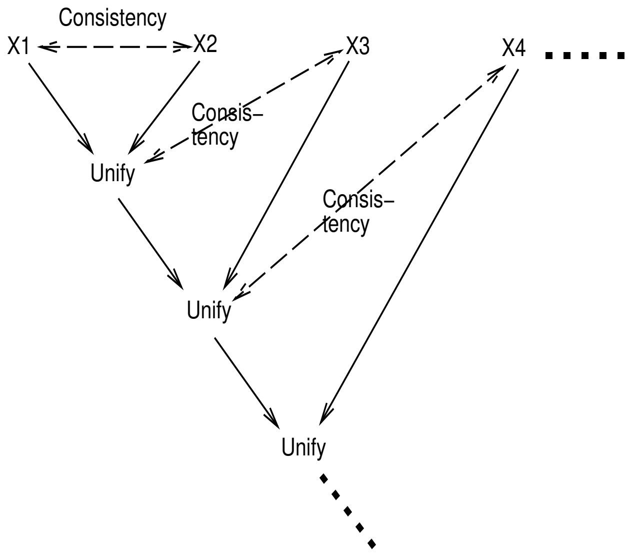

Figure 10: Binary Consistency Algorithm

In response to this observation we seek the unification that has been developed the least, i.e. the one that is most abstract and is, in terms of development, closest to the original descriptions. In the above example this will give the required result: P_1 itself is the least reduced unification, up to testing equivalence, of P_1 and P_2. The issue is that we could choose a unification of two descriptions that is too developed to be reconciled with a third description, while a less developed unification that could be reconciled, exists. We will consider the issue of least developed unifications next.

5.2 Least Developed Unifications

In traditional single threaded (waterfall) models of system development the issue of least development does not arise. This is because, assuming development is a preorder, each description is a least development of itself. Unfortunately, the situation is not so straightforward when we generalise to viewpoints and must reconcile the development trajectory of more than one description.

First our interpretation of least developed unification [9]. We assume $dv_i,\ 1 \leq i \leq n$, are preorders.

Definition 18 (Least Developed Unification)
*$X \in \overset{1..n}{\mathcal{U}}(dv_i, X_i)$ is a least developed unification iff $\forall X' \in \overset{1..n}{\mathcal{U}}(dv_i, X_i)\ .\ X'\ \overset{n}{\cap} dv_i\ X$,
where $\overset{n}{\cap} dv_i$ is a shorthand for $dv_1 \cap ... \cap dv_n$.*

This definition ensures that all unifications are a development of X. Notice the interpretation of development, that X and X' are related by $dv_1 \cap ... \cap dv_n$, i.e. the set of unifications is ordered by the intersection of the development relations used in unification. This is a natural interpretation since all descriptions in the unification set are developments of the least developed unification by all relevant development relations. Also notice that the least developed unification is unique (By a clarification of what we mean by uniqueness, here we are talking about uniqueness up to equivalence, where equivalence is interpreted as $\asymp_{(dv_1 \cap ... \cap dv_n)}$ for $dv_1, ..., dv_n$ the relevant development relations.).

Unfortunately, for inter language consistency, the least developed of the set of unifications is a problematic concept. Specifically, descriptions in the unification set, $\overset{1..n}{\mathcal{U}}(dv_i, X_i)$, are likely to be in a different notation from $X_1, ..., X_n$; thus it is unlikely that the unifications can be related in a type correct manner using $dv_1 \cap .. \cap dv_n$.

[9] We should point out that the terminology that we use here is slightly different to that which we have used in earlier work. For example, in [16] least unification has a more general meaning than it does here. We now believe that the definition given here leads to a more intuitive exposition of the concepts.

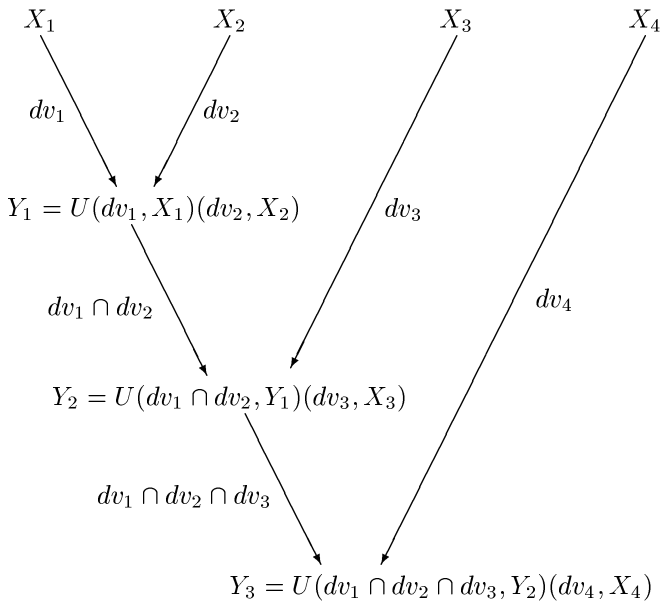

Figure 11: A Global Consistency Checking Algorithm

Thus, this definition and the remaining theory is only applicable to intra language consistency. Ongoing work is addressing generalisation of least developed unification to the inter language setting.

It should also be emphasized that there is nothing in the nature of development to ensure that a least developed unification exists and absence of a least developed unification is a big problem; elsewhere we have considered how to obtain global consistency from binary consistency checking in this situation [10]. This work has shown that as long as all infinite chains in the unification set are bounded incremental consistency checking can be obtained. However, this is only obtained by considering a set of candidate unifications at each stage in the consistency checking algorithm. For all practical purposes checking against, an admittedly finite, but possibly very large set of candidate unifications, is not feasible. As a reflection of this fact, we will not consider this class of consistency further here; we refer the interested reader to the theoretical results to be found in [10].

The next section seeks to clarify under what circumstances a least developed unification will exist in the general case of unbalanced consistency. Then section 7 will consider the same issue in the more restricted setting of balanced consistency. In both sections, particular LOTOS consistency checks are considered by way of illustration.

6 Least Developed Unification and Unbalanced Consistency

A least developed unification is a greatest element in the unification set ([10] considers this perspective in some depth). It is well known that greatest elements of partially ordered sets are unique up to equivalence. Thus, we can determine least developed unifications using the following function:

Notation 1
If it exists we denote the least developed unification as $lu(\overset{1..n}{\mathcal{U}}(dv_i, X_i), \overset{n}{\sqcap} dv_i)$*. This function returns* $\perp$ *if there is no least developed unification.*

It is worth pointing out again that we are considering uniqueness up to equivalence. Thus, $lu(S, dv)$ is the choice function from the relevant equivalence class, such a function existing by the axiom of choice. In order to simplify notation we will often write the least developed unification as $\overset{1..n}{lu}(dv_i, X_i)$ or $lu(dv_1, X_1)...(dv_n, X_n)$.

We will also use the following simple result.

Proposition 24
$Y = lu(dv, X)(dv', X') \ \wedge \ Y' = lu(dv, X)(dv', X')(dv'', X'') \implies Y' \ dv \cap dv' \ Y.$

Proof
Clearly, $Y' \in \mathcal{U}(dv, X)(dv', X')(dv'', X'')$, but we can use corollary 1 to get $Y' \in \mathcal{U}(dv, X)(dv', X')$ and by the definition of lu we have $Y' \ dv \cap dv' \ Y$, as required. □

We are now in a position to relate binary consistency strategies to global consistency. In order to express the associativity result we require we consider a function β which is derived from lu. The function returns a pair, with first element the intersection of the development relations considered and second element the least developed unification. Notice a bottom element is returned as least developed unification if either a least developed unification does not exist or one of the descriptions given as an argument is undefined.

Definition 19
$\beta(dv, X)(dv', X') = (dv \cap dv', Y)$
where
if $X = \perp \ \vee \ X' = \perp \ \vee \ lu(dv, X)(dv', X') = \perp$ *then* $Y = \perp$
otherwise $Y = lu(dv, X)(dv', X')$.

We will prove associativity of β by relating the two possible binary bracketings of β to $lu(dv, X)(dv', X')(dv'', X'')$.

Proposition 25
$r(\beta(dv, X)(\beta(dv', X')(dv'', X''))) \asymp_{dv \cap dv' \cap dv''} lu(dv, X)(dv', X')(dv'', X'')$ *and,*
$r(\beta(\beta(dv, X)(dv', X'))(dv'', X'')) \asymp_{dv \cap dv' \cap dv''} lu(dv, X)(dv', X')(dv'', X'')$
where, r is the right projection function, which yields the second element of a pair.

Proof
See [10]. □

Now if we define equality pairwise as,

$$(dv, X) = (dv', X') \text{ iff } dv = dv' \ \wedge \ X \asymp_{dv \cap dv'} X'$$

the following result is straightforward.

Corollary 6
$\beta(dv, X)(\beta(dv', X')(dv'', X'')) = \beta(\beta(dv, X)(dv', X'))(dv'', X'')$

Proof
Follows immediately from previous two results, propositions 25 and 25. □

This is a full associativity result which gives us that any bracketing of $\beta(dv_1, X_1), ..., (dv_n, X_n)$ is equal. Since β is just an alternative coding of lu that facilitates clarity of expression, we have full associativity of lu and that a consistency strategy using lu can be composed of any order of binary consistency checks. So, if least developed unifications exist, we can obtain global consistency from any appropriate series of binary consistency checks. This is an important result that arises from a well behaved class of unification.

The next question to ask is what conditions can we impose on development in order to obtain the existence of a least developed unification? The following property will certainly do.

Property 1 *An FDT, ft, satisfies property 1 iff,*

$$\forall X_1, ..., X_n \in DES_{ft} \ \wedge \ \forall dv_1, ..., dv_n \in DEV_{ft}, \ (\overset{1..n}{\mathcal{U}}(dv_i, X_i) \neq \emptyset \implies lu(\overset{1..n}{\mathcal{U}}(dv_i, X_i), \overset{n}{\cap} dv_i) \neq \perp).$$

This property ensures that any possible combination of descriptions and development relations in ft will generate a unification set with a greatest element.

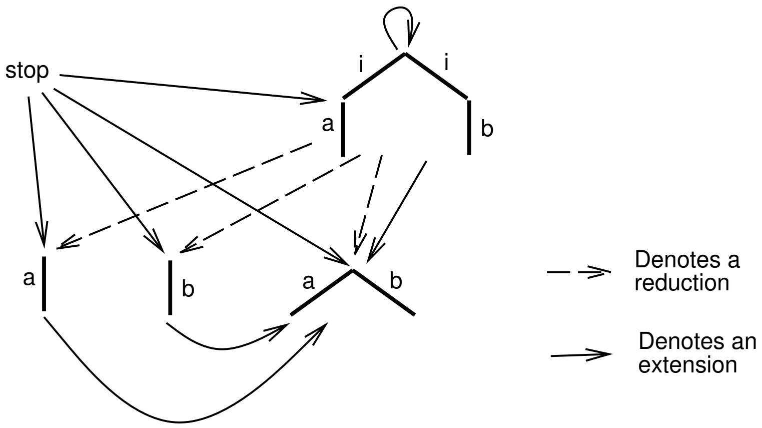

Figure 12: Unification Set for Unbalanced Consistency

LOTOS Illustration 5 *We will consider whether least developed unifications exist for binary combinations of the three main LOTOS preorders, i.e.* $C(\mathbf{ext}, P_1)(\mathbf{red}, P_2)$, $C(\mathbf{ext}, P_1)(\leq_{tr}, P_2)$ *and* $C(\leq_{tr}, P_1)(\mathbf{red}, P_2)$

Consistency by ext and red. *The following counterexample demonstrates that a least developed unification for this consistency check does not exist:*

$$P := stop \quad and \quad Q := i; a; stop[]i; b; stop$$

The unification set, $\mathcal{U}(\mathbf{ext}, P)(\mathbf{red}, Q)$ *is shown in figure 12 (identity arrows have not been included). All four of the unifications,* $a; stop$, $b; stop$, $a; stop[]b; stop$ *and* Q, *are minimally developed unifications*[10] *but none of them is less developed according to* $\mathbf{ext}\cap\mathbf{red}$ *than all the other three. Thus,* P *and* Q *have no least developed unification.*

Before we explain why a least developed unification can not always be found, we first note that for all P_1 *and* P_2 $\mathcal{U}(\mathbf{ext}, P_1)(\mathbf{red}, P_2)$ *will be empty (i.e* P_1 *and* P_2 *will not be consistent) unless the following condition holds.*

$$(*) \qquad Tr(P_1) \subseteq Tr(P_2)$$

If the traces of P_1 *are not a subset of the traces of* P_2 *then extension of* P_1 *and reduction of* P_2 *can not be reconciled.*

Thus, assuming $(*)$ *the unification that we require must at least satisfy:*

$$Tr(U_{er}(\mathbf{ext}, P_1)(\mathbf{red}, P_2)) \supseteq Tr(P_1) \quad \wedge \quad Tr(U_{er}(\mathbf{ext}, P_1)(\mathbf{red}, P_2)) \subseteq Tr(P_2)$$

where we have denoted the required binary unification function as U_{er}. *However, this leaves too much flexibility in the choice of unification. We could choose the traces of the selected unification to be equal to the traces of* P_1 *or the traces of* P_2 *or to be somewhere between the two. Any of these options would enable unification, but none would realise a least developed unification.*

Consistency by ext and trace preorder. *A least developed unification does not in general exist here either. An argument similar to that just made can be given.*

Consistency by trace preorder and red. *Consider the unification function* $U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)$ *characterised by the following trace refusal properties,*

$$Tr(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)) = Tr(P) \cap Tr(Q) \quad \wedge$$
$$\forall \sigma \in Tr(P) \cap Tr(Q),\ Ref(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q), \sigma) = Ref(Q, \sigma)$$

[10] A minimally developed unification, X, satisfies the following property: $\neg(\exists X' \in \overset{1..n}{\mathcal{U}}(dv_i, X_i)\ .\ X \overset{n}{\cap} dv_i\ X')$. The existence of minimally developed unifications does not imply the existence of least developed unifications. However, the other direction of implication does hold. Thus, least developed unification is a strictly stronger concept than minimally developed unification.

which, if it exists, can be used to derive a LOTOS process that is unique up to equivalence and is a least developed unification, proof of this fact is presented in the appendix, proposition 27. The issue of existence is actually crucial. Specifically, the above properties may not always characterise a "well formed" LOTOS process. For example, unification of the LOTOS processes $P_1 := a; stop$ and $P_2 := b; stop$ will require that,

$$Tr(U_{tr}(\leq_{tr}, P_1)(\mathbf{red}, P_2)) = \{\epsilon\} \ \wedge$$
$$Ref(U_{tr}(\leq_{tr}, P_1)(\mathbf{red}, P_2), \epsilon) = \{\emptyset, \{a\}\} \quad \textit{(assuming } \mathcal{L} = \{a, b\}\textit{)}$$

which implies that after the empty trace no actions are offered (as ϵ is the only trace) and b is not refused. Clearly, b not being offered implies it should be refused and no LOTOS process can realise these properties. [41] locates a set of conditions that characterise when a trace/refusal pair is well formed, in the sense that it can be realised as a LOTOS process. $U_{tr}(\leq_{tr}, P_1)(\mathbf{red}, P_2)$ will fail condition (f) on page 72 of [41]. It is beyond the scope of this paper to present these conditions here.

Importantly though, it can be shown that:

$$\forall P, Q \in DES_{LOTOS},\ U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q) \textit{ is well formed if and only if } C(\leq_{tr}, P)(\mathbf{red}, Q)$$

Thus, an approach to consistency checking LOTOS specifications is to unify specifications and then consider whether the unification is well formed. This is an alternative to the approach in [49] where conditions are highlighted which can be checked to show that specifications are consistent and a unification is only derived once it is known that one exists.

7 Strategies for Checking Balanced Consistency

The majority of work on consistency to be found in the literature has addressed restricted classes of consistency; to date, balanced consistency has almost exclusively been focused on. So, what in this restricted setting, enables us to obtain global consistency from binary consistency? We would like to locate a specialization of the existence of least developed unifications. As might be expected, the greatest lower bound gives us this specialization. The property that we require for balanced consistency checking to be performed incrementally is:

Property 2
$\forall \{X_1, ..., X_n\} \subseteq DES_{ft} \ \wedge \ \forall dv \in DEV_{ft},\ lb(\{X_1, ..., X_n\}, dv) \neq \emptyset \implies glb(\{X_1, ..., X_n\}, dv) \neq \emptyset.$

This property ensures that if a lower bound exists then a greatest lower bound can be found, i.e. the unification of $X_1, ..., X_n$ is non-empty implies a least developed unification exists. It is clear from the theory of least developed unifications we have presented and from set theory that taking greatest lower bounds is associative, i.e.

$$glb(\{glb(\{X_1, X_2\}, dv), X_3\}, dv) \asymp_{dv} glb(\{X_1, glb(\{X_2, X_3\}, dv)\}, dv)$$

and can thus be used to derive global consistency from binary consistency. In order to simplify notation here we have assumed that glb returns an arbitrary element from the equivalence class of its results. With these concepts we can identify what is the most well behaved class of development.

Definition 20 *(DES_{ft}, dv) is cocomplete iff $\forall S \subseteq DES_{ft},\ glb(S, dv) \neq \emptyset$.*

Cocompleteness is a dual concept to that of a complete partial order (see for example [44]) which considers the existence of least upper bounds as opposed to greatest lower bounds. If development is cocomplete for a particular FDT according to a development relation, then all specifications are balanced consistent and we can adopt any relevant incremental unification strategy. All descriptions are consistent since a lower bound exists for all collections of descriptions and incremental unification is well behaved since least developed unifications always exist.

LOTOS Illustration 6 *We will consider in turn the consistency checks: $C_{\mathbf{red}}$, $C_{\mathbf{ext}}$ and $C_{\leq_{tr}}$. The work presented here is closely related to that considered in [41].*

Balanced Consistency by red. *Consider the binary unification strategy $U_{\mathbf{red}}PQ$ characterised by the following trace/refusal properties:*

$$Tr(U_{\mathbf{red}}PQ) = Tr(P) \cap Tr(Q) \quad \wedge \quad \forall \sigma \in Tr(U_{\mathbf{red}}PQ),\ Ref(U_{\mathbf{red}}PQ, \sigma) = Ref(P, \sigma) \cap Ref(Q, \sigma)$$

In the appendix (proposition 28) we prove that $U_{\mathbf{red}}$ gives the greatest lower bound. If it exists, $U_{\mathbf{red}}$ is unique up to testing equivalence. In addition, in a similar way to U_{tr}, $U_{\mathbf{red}}$ does not always characterise a "well formed" LOTOS process. This reflects the fact that $C_{\mathbf{red}}$ is not completely consistent. For example, consider $P := a; stop$ and $Q := b; stop$, which we have already argued (in proposition 20) are not consistent by reduction. We obtain that $U_{\mathbf{red}}$ is characterised by,

$$Tr(U_{\mathbf{red}}PQ) = \{\epsilon\} \quad \wedge \quad Ref(U_{\mathbf{red}}PQ, \epsilon) = \{\emptyset\} \qquad \textit{(assuming that } \mathcal{L} = \{a, b\}\textit{)}$$

i.e. $U_{\mathbf{red}}PQ$ is a process that performs no traces and refuses nothing. Once again we obtain that,

$$\forall P, Q \in DES_{basicLOTOS},\ C_{\mathbf{red}}PQ \textit{ holds if and only if } U_{\mathbf{red}}PQ \textit{ is well formed.}$$

So, this binary unification function enables us to do incremental consistency checking for balanced consistency according to reduction. But, $C_{\mathbf{red}}$ is not cocomplete as balanced consistency according to reduction is not completely consistent.

Balanced Consistency by ext. *Consider the binary unification strategy, $U_{\mathbf{ext}}PQ$, characterised by the following trace/refusal properties:*

$$\begin{array}{l} Tr(U_{\mathbf{ext}}PQ) = Tr(P) \cup Tr(Q) \quad \wedge \\ \forall \sigma \in Tr(U_{\mathbf{ext}}PQ), \\ \qquad\qquad \sigma \in Tr(P) \cap Tr(Q) \Longrightarrow Ref(U_{\mathbf{ext}}PQ, \sigma) = Ref(P, \sigma) \cap Ref(Q, \sigma) \ \wedge \\ \qquad\qquad \sigma \in Tr(P) - Tr(Q) \Longrightarrow Ref(U_{\mathbf{ext}}PQ, \sigma) = Ref(P, \sigma) \ \wedge \\ \qquad\qquad \sigma \in Tr(Q) - Tr(P) \Longrightarrow Ref(U_{\mathbf{ext}}PQ, \sigma) = Ref(Q, \sigma) \end{array}$$

Proposition 29 in the appendix verifies that this unification function gives the greatest lower bound. Once again this construction characterises a LOTOS process that is unique up to testing equivalence. In addition, $U_{\mathbf{ext}}PQ$ can be shown to be well founded for all $P, Q \in DES_{basicLOTOS}$. Thus, $U_{\mathbf{ext}}$ gives a valid greatest lower bound for all pairs of specifications and thus balanced consistency by extension is cocomplete.

Balanced Consistency by Trace Preorder. *The binary unification strategy,*

$$U_{\leq_{tr}}PQ \ : \ Tr(P) \cap Tr(Q)$$

is unique up to $\asymp_{\leq_{tr}}$ and can easily be seen to generate the greatest lower bound of any two basic LOTOS processes P and Q. Thus, since $C_{\leq_{tr}}$ is completely consistent, we know that balanced consistency according to trace preorder is cocomplete.

8 Reflection on Consistency in LOTOS

One of our reasons for using LOTOS to illustrate consistency checking is that it offers a spectrum of development relations. This in particular enables us to illustrate unbalanced consistency within a single language. These illustrations give a perspective on the bounds of consistency checking for LOTOS.

In summary, all the balanced consistency instantiations turn out to be relatively well behaved. In particular, least developed unifications exist for all the following checks, $C_{\leq_{tr}}$, $C_{\mathbf{red}}$ and $C_{\mathbf{ext}}$. In contrast, the unbalanced consistency situations are not as well behaved: $C_{\leq_{tr},\mathbf{red}}$ yields a least developed unification, but $C_{\mathbf{ext},\mathbf{red}}$ and $C_{\leq_{tr},\mathbf{ext}}$ do not. This is not suprising as the relations, **ext** and **red** and **ext** and $\leq_{tr}$, are so very different.

This leaves us with a difficulty, how can we obtain global consistency from binary consistency when we wish to extend the functionality of only one of the original specifications. A possible approach to this is to adapt the original specifications using undefined behaviour and to do away with extension. This is an approach that has been used elsewhere [11] [40] in order to enable functionality extension in process algebra refinement methods.

To illustrate this approach, consider the following simple specification:

$$P \ := \ a; B \,[]\, b; B'$$

as it stands P will initially refuse any action other than a or b. In addition, any specification, Q say, that adds an alternative initial behaviour, e.g.,

$$Q := P\,[]\,c; B''$$

would fail to be a reduction of P, since Q would add traces to those of P. However, perhaps when we specify P, we do actually want to allow such addition of functionality. We can obtain this effect and stick with reduction by adding undefined behaviour to P. Consider the following behaviours:

$\Phi := \mathit{choice}\ x \in \mathit{Act} \setminus \{a,b\}\ []\ x; \Omega$
where,
$\Omega := (\mathit{choice}\ y \in \mathit{Act}\ []\ i; y; \Omega)\ []\ i; \mathit{stop}$

Now Ω is a completely undefined and unpredictable behaviour; at any state it may non-deterministically decide to do anything. In addition, Ω is at the top of the reduction preorder: anything is a reduction of it.

Now if we adapt P to P', as follows:

$$P' := i; P\ []\ \Phi$$

then a behaviour such as Q would indeed be a reduction of P'.

We can highlight a similar situation in the context of our ODP viewpoints example. In that example, *Obl1*, which was defined,

Obl1 := request?x:ID'; stop

is related according to extension. However, we can get the same effect as this if (assuming $Act = \pi(GG)$) we use the specification,

$$\begin{array}{lll} \mathit{NewObl1} & := & i;\ \mathit{request?x{:}ID'};\ \Omega \\ & & []\ (\mathit{choice}\ d \in \pi(GG\text{-}\{\mathit{request}\})\ []\ d;\ \Omega \end{array}$$

and relate it according to reduction.

The issue is that attempting to apply an action at a state in which it is not offered results in deadlock in LOTOS (and this prevents adding that action during refinement by reduction), however, we have made its result undefined (which can be refined by reduction). Such a use of undefined corresponds to the interpretation employed in pre and postcondition based refinement, such as in Z [47]. In such approaches applying an operation outside its precondition conceptually corresponds to attempting to perform an action at a state in which it is not offered. In pre and post-condition approaches such as Z applying an operation outside its precondition yields undefined rather than refusal. This is why refinement in Z enables functionality to be extended. However, as we have illustrated such an effect can be obtained in LOTOS by adding undefinedness explicitly.

[11, 12] considers mechanisms to add undefined behaviour to abstract specifications in a systematic manner. However, in the context of this paper this addition of undefined behaviour is interesting since it enables functionality extension to be obtained without using extension. Consequently, we can restrict ourselves to the combinations of preorder refinement: $C_{\leq_{tr},\mathbf{red}}$ and $C_{\mathbf{red}}$ which are more well behaved since they yield least developed unifications.

9 Related Work

A relatively substantial body of work on viewpoints related approaches to system development now exists. The majority of this work has considered partial specification in a particular specification notation. Issues such as unification and consistency checking arise in all these areas of investigation. Typical work on this topic is that by Wallis et al [2], [1], [3]; Jackson et al [37]; Boiten [6] and Derrick et al [23], [24], [13] for Z; and Leduc [41]; Khendek et al [38], Ichikawa et al [31] and Steen et al [49] for LOTOS. From amongst this body of language specific work Leduc's PhD work [41] has most influenced us. In fact, the trace/refusals theory presented here has grown out of Leduc's work.

An important body of research that is not language specific is the theory of institutions [28] and application of the theory to particular specification domains, in particular, by [26] to concurrent systems. However, it is valuable to relate the set theoretic constructions in this paper to categorical ones found in the theory of institutions:

- The ODP notion of correspondences between viewpoints plays a similar role to morphisms within a diagram in institutions, i.e. they identify how terms relate in different specifications.
- The cocone of a diagram is analogous to our notion of a unification.
- The colimit of a diagram is analogous to our notion of a least developed unification.
- The categorical and our notion of cocompleteness correspond.

Where the approaches differ is that composition in the institutions setting, e.g. in [26], typically consider composing components at a single level of system development, it is assumed that underlying models must be cocomplete, hence inconsistency is ruled out in the constraints imposed on the basic theory. The theory of institutions generalises logical frameworks and hence uses satisfaction, $\models$, as its core correctness relation, however, our framework is parameterised on the choice of development relation, which could be one of many relations. Our approach is prompted by the particular requirements of viewpoints in ODP, as indicated earlier in this paper.

There has now also been some work on specific mechanisms for checking consistency across languages; typical examples are the work of Zave et al [54] and Derrick et al [25]. The former of these considers a logical intermediary between languages and all notations are mapped to this intermediary. The latter approach has already been discussed.

10 Concluding Remarks

This paper has presented a general interpretation of consistency for multiple viewpoint models of system development and investigated possible consistency checking strategies. The main original contribution of the paper is the generality of the theory investigated. We have motivated the need for a general interpretation of consistency with reference to the requirements of viewpoints modelling in Open Distributed Processing. Our interpretation of consistency embraces intra and inter language consistency, balanced and unbalanced consistency and both binary and global consistency.

We have identified the properties of each of the classes of consistency; that we have considered and we have classified how global consistency can be derived from a series of binary consistency checks. This topic has been investigated in the past, but only in the context of a restricted class of consistency and this is the first paper to investigate consistency checking strategies for as general an interpretation of consistency as ours. The main difference between our theory and earlier work is that we handle unbalanced consistency.

The requirement that a least developed unification always exists was highlighted. If such a least developed unification does not always exist then incremental consistency checking is realistically impossible, although, we have considered the theoretical consequences of such a unification not existing elsewhere. We considered the existence of least developed unifications in both the general case and for balanced consistency. In the latter setting, the concept of a unification reduces to a lower bound and least developed unification to greatest lower bound.

Throughput we have illustrated the different varieties of consistency using LOTOS. These illustrations characterise the forms of consistency check that can arise with basic LOTOS. In particular, we have given a complete classification of the LOTOS balanced consistency relations, see figure 8, and we have highlighted which combinations of LOTOS refinement relations yield a least developed unification. For unbalanced consistency it was shown that $C(\mathbf{ext}, P_1)(\mathbf{red}, P_2)$ and $C(\mathbf{ext}, P_1)(\leq_{tr}, P_2)$ do not yield a least developed unification, while a least developed unification for $C(\leq_{tr}, P_1)(\mathbf{red}, P_2)$ can always be found. For balanced consistency $C_{\mathbf{red}}$, $C_{\mathbf{ext}}$ and $C_{\leq_{tr}}$ all have greatest lower bounds and thus least developed unifications, but only $C_{\mathbf{ext}}$ and $C_{\leq_{tr}}$ are cocomplete.

Acknowledgements. We would like to thank the reviewers of this paper for a number of useful comments.

References

[1] M. Ainsworth, A. H. Cruickshank, L. J. Groves, and P. J. L. Wallis. Formal specification via viewpoints. In J Hosking, editor, *Proc. 13th New Zealand Computer Conference*, pages 218–237, Auckland, New Zealand, 18th–20th August 1993. New Zealand Computer Society.

[2] M. Ainsworth, A. H. Cruickshank, L. J. Groves, and P. J. L. Wallis. Viewpoint specification and Z. *Information and Software Technology*, 36(1):43–51, February 1994.

[3] M. Ainsworth and P. J. L. Wallis. Co-refinement. In D Till, editor, *Proc. 6th Refinement Workshop*, City University, London, 5th–7th January 1994. Springer-Verlag.

[4] E. Boiten, H. Bowman, J. Derrick, and M. Steen. Cross viewpoint consistency in Open Distributed Processing (intra language consistency). Technical Report 8-95, Computing Laboratory, University of Kent at Canterbury, 1995.

[5] E. Boiten, H. Bowman, J. Derrick, and M. Steen. Viewpoint consistency in Z and LOTOS: A case study. In J. Fitzgerald, C. B. Jones, and P. Lucas, editors, *Formal Methods Europe (FME '97)*, LNCS 1313, pages 644–664, Graz, Austria, September 1997. Springer-Verlag.

[6] E. Boiten, J. Derrick, H. Bowman, and M. Steen. Consistency and refinement for partial specification in Z. In M.-C. Gaudel and J. Woodcock, editors, *FME'96: Industrial Benefit of Formal Methods, Third International Symposium of Formal Methods Europe*, volume 1051 of *Lecture Notes in Computer Science*, pages 287–306. Springer-Verlag, March 1996.

[7] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14(1):25–59, 1988.

[8] G. Booch. *Object-oriented Analysis and Design.* The Benjamin/ Cummings Publishing Company, Inc, 1994.

[9] G. Booch, J. Rumbaugh, and I. Jacobson. *Unified Modeling Language.* Rational Software Corporation, January 1997.

[10] H. Bowman, E.A. Boiten, J. Derrick, and M.W.A. Steen. Strategies for consistency checking based on unification. *Science of Computer Programming*, 33:261–298, March 1999.

[11] H. Bowman, C. Briscoe-Smith, J. Derrick, and B. Strulo. On behavioural subtyping in LOTOS. In H. Bowman and J. Derrick, editors, *FMOODS'97, 2nd IFIP Conference on Formal Methods for Open Object Based Distributed Systems*, pages 335–351. Chapman and Hall, July 1997.

[12] H. Bowman and J. Derrick. A junction between state based and behavioural specification. In A.F̃antechi P.C̃iancarini and R. Gorrieri, editors, *FMOODS'99, 3rd IFIP Conference on Formal Methods for Open Object Based Distributed Systems*, pages 213–239. Kluwer, February 1999.

[13] H. Bowman, J. Derrick, P. Linington, and M.W.A. Steen. FDTs for ODP. *Computer Standards and Interfaces*, 17:457–479, September 1995.

[14] H. Bowman, J. Derrick, P.F. Linington, and M.W.A. Steen. Cross viewpoint consistency in Open Distributed Processing. *IEE Software Engineering Journal, Special Issue on Viewpoints, Editors: A. Finkelstein and I. Sommerville*, 11(1):44–57, January 1996.

[15] H. Bowman, J. Derrick, and M.W.A. Steen. Some results on cross viewpoint consistency checking. In K. Raymond and L. Armstrong, editors, *IFIP TC6 International Conference on Open Distributed Processing*, pages 399–412, Brisbane, Australia, February 1995. Chapman and Hall.

[16] H. Bowman, E.A.Boiten, J. Derrick, and M.W.A. Steen. Viewpoint consistency in ODP, a general interpretation. In *First IFIP International workshop on Formal Methods for Open Object-based Distributed Systems*, pages 189–204, Paris, March 1996. Chapman & Hall.

[17] E. Brinksma. What is the method in formal methods. In K.R. Parker and G.A. Rose, editors, *FORTE'91, Formal Description Techniques, IV*, pages 33–50, Sydney, Australia, 1992. North-Holland.

[18] E. Brinksma and G. Scollo. Formal notions of implementation and conformance in LOTOS. Technical Report INF-86-13, Dept of Informatics, Twente University of Technology, 1986.

[19] E. Brinksma, G. Scollo, and C. Steenbergen. Process specification, their implementation and their tests. In B. Sarikaya and G. V. Bochmann, editors, *Protocol Specification, Testing and Verification, VI*, pages 349–360, Montreal, Canada, June 1986. North-Holland.

[20] G. Cowen, J. Derrick, M. Gill, G. Girling (editor), A. Herbert, P. F. Linington, D. Rayner, F. Schulz, and R. Soley. *Prost Report of the Study on Testing for Open Distributed Processing.* APM Ltd, 1993.

[21] J. Derrick, E.A. Boiten, H. Bowman, and M. Steen. Translating LOTOS to Object-Z. In D.J. Duke and A.S. Evans, editors, *Northern Formal Methods Workshop*, volume 2nd BCS-FACS Northern Formal Methods Workshop of *Workshops in Computing.* Springer-Verlag, July 1997.

[22] J. Derrick, E.A. Boiten, H. Bowman, and M. Steen. Viewpoints and consistency: translating LOTOS to Object-Z. *Computer Standards and Interfaces*, 1999. to appear.

[23] J. Derrick, H. Bowman, and M. Steen. Maintaining cross viewpoint consistency using Z. In K. Raymond and L. Armstrong, editors, *IFIP TC6 International Conference on Open Distributed Processing*, pages 413–424, Brisbane, Australia, February 1995. Chapman and Hall.

[24] J. Derrick, H. Bowman, and M. Steen. Viewpoints and Objects. In *Ninth Annual Z User Workshop, Lecture Notes in Computer Science 967*, Limerick, September 1995. Springer-Verlag.

[25] J. Derrick, E.A.Boiten, H. Bowman, and M. Steen. Supporting ODP - translating LOTOS to Z. In *First IFIP International workshop on Formal Methods for Open Object-based Distributed Systems*, pages 399–406, Paris, March 1996. Chapman & Hall.

[26] J. Fiadeiro and T. Maibaum. Temporal theories as modularisation units for concurrent system specification. *Formal Aspects of Computing*, 4:239–272, 1992.

[27] 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.

[28] J. Goguen and R. Burstall. Introducing institutions. In E. Clarke and D. Kozen, editors, *Proceedings Logics of Programming Workshop*, volume 164 of *Lecture Notes in Computer Science*, pages 221–256. Springer-Verlag, 1984.

[29] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, 1988.

[30] C. A. R. Hoare. *Communicating Sequential Processes.* Prentice Hall, 1985.

[31] H. Ichikawa, K. Yamanaka, and J. Kato. Incremental Specification in LOTOS. In L. Logrippo, R. L. Probert, and H. Ural, editors, *Protocol Specification, Testing and Verification X*, pages 183–196, Ottawa, Canada, 1990.

[32] ISO. Information Processing Systems – Open Systems Interconnection – Basic Reference Model, 1984. IS 7498.

[33] ISO. Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour, 1989. IS 8807.

[34] ISO. LOTOS description of the Session Protocol, 1989. ISO/IEC TR9572.

[35] ISO. LOTOS description of the Session Service, 1989. ISO/IEC TR9571.

[36] ITU Recommendation X.901-904 — ISO/IEC 10746 1-4. *Open Distributed Processing - Reference Model - Parts 1-4*, July 1995.

[37] D. Jackson. Structuring Z specifications with views. *ACM Transactions on Software Engineering and Methodology*, 4(4):365–389, October 1995.

[38] F. Khendek and G. von Bochmann. Merging specification behaviours. Technical Report 856, Departement d'informatique et de recherche operationnelle, Universite de Montreal, 1993.

[39] G. Kiczales, J. Lamping, A. Mendhekar, C. Maeda, C. Lopes, J.-M. Loingtier, and J. Irwin. Aspect-oriented programming. Technical Report SPL97-008 P9710042, PARC, February 1997.

[40] K.G. Larsen, B. Steffen, and C. Weise. A constraint oriented proof methodology based on modal transition systems. Technical Report RS-94-47, University of Aarhus, 1994.

[41] G. Leduc. *On the Role of Implementation Relations in the Design of Distributed Systems using LOTOS.* PhD thesis, University of Liège, Liège, Belgium, June 1991.

[42] G. Leduc. A framework based on implementation relations for implementing LOTOS specifications. *Computer Networks and ISDN Systems*, 25:23–41, 1992.

[43] P. F. Linington. RM-ODP: The Architecture. In K. Raymond and L. Armstrong, editors, *IFIP TC6 International Conference on Open Distributed Processing*, pages 15–33, Brisbane, Australia, February 1995. Chapman and Hall.

[44] J. Loeckx and K. Sieber. *The Foundations of Program Verification.* Wiley, 1984.

[45] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[46] D.L. Parnas. Language-free mathematical methods for software design (invited paper). In *Ninth Annual Z User Workshop*, Limerick, September 1995. Springer-Verlag.

[47] B. Potter, J. Sinclair, and D. Till. *An introduction to formal specification and Z.* Prentice Hall, 1991.

[48] I. Sommerville. *Software Engineering.* Addison-Wesley, 1989.

[49] M. Steen, H. Bowman, and J. Derrick. Composition of LOTOS specifications. In P. Dembinski and M. Sredniawa, editors, *Protocol Specification, Testing and Verification*, Warsaw, Poland, 1995. Chapman & Hall.

[50] M.W.A. Steen. *Consistency and Composition of Process Specifications.* PhD thesis, University of Kent at Canterbury, Canterbury, Kent, UK, 1998.

[51] J. Woodcock and J. Davies. *Using Z: Specification, Refinement, and Proof.* Prentice Hall, 1996.

[52] R. Worden. Fermenting and distilling. In J.P. Bowen and J.A. Hall, editors, *ZUM'94, Z User Workshop*, pages 1–6, Cambridge, United Kingdom, June 1994.

[53] J-P. Wu and S. Chanson. Translation from LOTOS and Estelle specifications to extended transition system and its verification. In S. T. Voung, editor, *Formal Description Techniques, II*, pages 533–549, Vancouver, Canada, December 1989. North-Holland.

[54] P. Zave and M. Jackson. Conjunction as composition. *ACM Transactions on Software Engineering and Methodology*, 2(4):379–411, October 1993.

Appendix

Results for Section 3.2.7

Proposition 26
(i) $\leq_{tr}$, $\mathbf{red}$ *and* $\mathbf{ext}$ *are preorders.*
(ii) $\mathbf{te}$, $\sim$, *and* $\approx$ *are equivalences.*
(iii) $\sim\subset\approx\subset \mathbf{te} \subset \mathbf{cs}$
(iv) **conf** *is reflexive, but neither symmetric nor transitive.*
(v) **cs** *is (a) reflexive and (b) symmetric, but (c) not transitive.*
(vi) **xcs** *is a preorder, it is (a) reflexive, (b) not symmetric and (c) transitive.*

Proof
(i), (ii) and (iv) are all standard results from the theory of LOTOS and process algebra in general, see for instance [41], [30] and [45]. However, (iii), (v) and (vi) require some justification.

Proof of (iii). $\sim\subset\approx\subset \mathbf{te}$ are standard process algebra results. $\mathbf{te} \subset \mathbf{cs}$ requires some justification. Firstly, it is straightforward to see that $\mathbf{te} \subseteq \mathbf{cs}$. In addition, we can provide the two processes $P := a; stop[]i; b; stop$ and

$Q := i; b; stop$ as counterexamples to justify that **cs** $\not\subseteq$ **te**, since P **cs** Q, but $\neg(P$ **te** $Q)$ as the trace sets of the two processes are not equal.

Proof of (v). This holds for the following reasons:-

(v.a) This is a consequence of **conf** being reflexive.

(v.b) This is immediate from the definition of **cs**.

(v.c) The following counterexample justifies this. Let $P := b; stop[]i; a; stop$; $Q := i; a; stop$ and $R := b; c; stop[]i; a; stop$; then P **cs** Q, Q **cs** R, but $\neg(P$ **cs** $R)$. This is because $\neg(P$ **conf** $R)$ as P refuses c after the trace b, but R cannot refuse c after the same trace.

Proof of (vi). This holds for the following reasons:-

(vi.a) Take $P \in DES_{LOTOS}$, then P **ext** P and P **conf** P (by reflexivity of extension and conformance) so P **xcs** P as required.

(vi.b) Consider the processes $P := b; stop[]i; a; stop$ and $Q := a; stop$. Now P **xcs** Q since P **cs** Q and $Tr(P) \supseteq Tr(Q)$, but, $\neg(Q$ **xcs** $P)$ because $\neg(Tr(Q) \supseteq Tr(P))$.

(vi.c) Assume P **xcs** Q and Q **xcs** R, we need P **xcs** R. Now P **xcs** Q and Q **xcs** R imply P **ext** Q and Q **ext** R which implies P **ext** R (by transitivity of **ext**). So, all that remains is to show that R **conf** P. But, since $Tr(R) \subseteq Tr(P)$ and $\forall \sigma \in Tr(R),\ Ref(P,\sigma) = Ref(Q,\sigma)\ \wedge\ Ref(Q,\sigma) = Ref(R,\sigma)$ [11]. We can derive that, $\forall \sigma \in Tr(R),\ Ref(P,\sigma) = Ref(R,\sigma)$. In addition, $\forall \sigma \in Tr(P) - Tr(R)$, $Ref(R,\sigma) = \emptyset$ which trivially implies $Ref(P,\sigma) \supseteq Ref(R,\sigma)$, as required.

$\square$

Results for Section 6

Proposition 27
$\forall P, Q \in DES_{basicLOTOS}$, *s.t.* $C(\leq_{tr}, P)(\mathbf{red}, Q)$ *holds,* $U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)$ *is the least developed unification.*

Proof
First we show that $U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)$ is a unification and then we show that it is the greatest such unification.

Unification. $U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)\ \leq_{tr}\ P$ follows immediately, since $Tr(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)) = Tr(P) \cap Tr(Q) \subseteq Tr(P)$. In addition, $U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)$ **red** Q since,

$$Tr(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)) = Tr(P) \cap Tr(Q) \subseteq Tr(Q) \quad \wedge$$
$$\forall \sigma \in Tr(Q),$$
$$(\sigma \in Tr(P) \implies Ref(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q), \sigma) = Ref(Q, \sigma)) \quad \wedge$$
$$(\sigma \notin Tr(P) \implies Ref(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q), \sigma) = \emptyset \subseteq Ref(Q, \sigma))$$

Greatest Unification. Take $R \in \mathcal{U}(\leq_{tr}, P)(\mathbf{red}, Q)$, we need to show that $R\ (\leq_{tr} \cap \mathbf{red})\ U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)$. However, $(\leq_{tr} \cap \mathbf{red}) = \mathbf{red}$, so all we actually need to show is that R **red** $U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)$. Since R is a unification it must satisfy:

$$Tr(R) \subseteq Tr(P), Tr(Q) \quad \wedge \quad \forall \sigma \in Tr(Q),\ Ref(R, \sigma) \subseteq Ref(Q, \sigma)$$

So, we can immediately obtain:

(i) $Tr(R) \subseteq Tr(P) \cap Tr(Q) = Tr(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)) \quad \wedge$

(ii) $\sigma \in Tr(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)) \Longrightarrow \sigma \in Tr(Q)$, so, $\forall \sigma \in Tr(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)),\ Ref(R, \sigma) \subseteq Ref(Q, \sigma) = Ref(U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q), \sigma)$.

which gives R **red** $U_{tr}(\leq_{tr}, P)(\mathbf{red}, Q)$ as required. $\square$

[11] Notice, P **xcs** $Q \implies \forall \sigma \in Tr(P),\ Ref(Q, \sigma) \subseteq Ref(P, \sigma)\ \wedge\ \forall \sigma \in Tr(Q),\ Ref(P, \sigma) \subseteq Ref(Q, \sigma)$, but also $Tr(Q) \subseteq Tr(P)$. So, $\forall \sigma \in Tr(Q),\ Ref(Q, \sigma) = Ref(P, \sigma)$.

Results for Section 7

Proposition 28
$\forall P, Q \in DES_{basicLOTOS}$, s.t. $C_{\mathbf{red}}PQ$ holds, $U_{\mathbf{red}}PQ$ is the greatest lower bound of P and Q.

Proof
In the usual way we prove this in two halves: first we show that $U_{\mathbf{red}}$ is a lower bound and then we show that it is the greatest lower bound.

Lower Bound. We need to show that $U_{\mathbf{red}}PQ$ **red** P, Q. This follows since, $Tr(P) \cap Tr(Q) \subseteq Tr(P), Tr(Q)$ which is the required trace subsetting property and $\forall \sigma \in (Tr(P) \cup Tr(Q)) - (Tr(P) \cap Tr(Q))$, $Ref(U_{\mathbf{red}}PQ, \sigma) = \emptyset \subseteq Ref(Q, \sigma), Ref(P, \sigma)$ and $\forall \sigma \in Tr(P) \cap Tr(Q), Ref(U_{\mathbf{red}}PQ, \sigma) = Ref(P, \sigma) \cap Ref(Q, \sigma) \subseteq Ref(Q, \sigma), Ref(P, \sigma)$, which is the required refusals property.

Greatest Lower Bound. Assume R such that R **red** P and R **red** Q, we need to show that R **red** $U_{\mathbf{red}}PQ$. First we consider the traces. From R **red** P and R **red** Q we obtain that $Tr(R) \subseteq Tr(P), Tr(Q)$ which implies $Tr(R) \subseteq Tr(P) \cap Tr(Q) = Tr(U_{\mathbf{red}}PQ)$, which is the required trace subsetting property. Now we consider refusals. Firstly, note that $\forall \sigma \in Tr(R)$, $Ref(R, \sigma) \subseteq Ref(P, \sigma)$ and $Ref(R, \sigma) \subseteq Ref(Q, \sigma)$, which implies that $Ref(R, \sigma) \subseteq Ref(P, \sigma) \cap Ref(Q, \sigma) = Ref(U_{\mathbf{red}}PQ, \sigma)$. In addition, $\forall \sigma \in (Tr(U_{\mathbf{red}}PQ) - Tr(R))$, $Ref(R, \sigma) = \emptyset \subseteq Ref(U_{\mathbf{red}}PQ, \sigma)$. This gives us the required refusals property and we are done. □

Proposition 29
$\forall P, Q \in DES_{basicLOTOS}$, $U_{\mathbf{ext}}PQ$ is the greatest lower bound of P and Q.

Proof
Once again we prove this in two parts.

Lower Bound. We need to show that $U_{\mathbf{ext}}PQ$ **ext** P, Q. The trace property is easily obtained since $Tr(U_{\mathbf{ext}}PQ) = Tr(P) \cup Tr(Q) \supseteq Tr(P), Tr(Q)$. In addition, refusals are correctly related because $\forall \sigma \in Tr(P)$, $(\sigma \in Tr(Q) \Longrightarrow Ref(U_{\mathbf{ext}}PQ, \sigma) = Ref(P, \sigma) \cap Ref(Q, \sigma) \subseteq Ref(P, \sigma)) \wedge (\sigma \notin Tr(Q) \Longrightarrow Ref(U_{\mathbf{ext}}PQ, \sigma) = Ref(P, \sigma))$, and we can argue similarly about the refusals of Q. Thus, $U_{\mathbf{ext}}PQ$ **ext** P, Q, as required.

Greatest Lower Bound. Assume R **ext** P, Q, we need to show that R **ext** $U_{\mathbf{ext}}PQ$. Once again the trace property that we require is straightforward: $Tr(R) \supseteq Tr(P) \cup Tr(Q) = Tr(U_{\mathbf{ext}}PQ)$. The refusals property requires a little more work. We obtain that $\forall \sigma \in Tr(U_{\mathbf{ext}}PQ)$, $(\sigma \in Tr(P) \cap Tr(Q) \Longrightarrow Ref(R, \sigma) \subseteq Ref(P, \sigma) \cap Ref(Q, \sigma) = Ref(U_{\mathbf{ext}}PQ, \sigma)) \wedge (\sigma \in Tr(P) - Tr(Q) \Longrightarrow Ref(R, \sigma) \subseteq Ref(P, \sigma) = Ref(U_{\mathbf{ext}}PQ, \sigma)) \wedge (\sigma \in Tr(Q) - Tr(P) \Longrightarrow Ref(R, \sigma) \subseteq Ref(Q, \sigma) = Ref(U_{\mathbf{ext}}PQ, \sigma))$ which is sufficient to show that R **ext** $U_{\mathbf{ext}}PQ$, as required. □