Comparing LOTOS and Z refinement relations

John Derrick, Howard Bowman, Eerke Boiten and Maarten Steen
Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK.
(Email: jd1@ukc.ac.uk.)

Abstract

This paper compares the LOTOS and Z refinement relations. The motivation for such a comparison is the use of multiple viewpoints for specifying complex systems defined by the reference model of the Open Distributed Processing (ODP) standardization initiative.

The ODP architectural semantics describes the application of formal description techniques (FDTs) to the specification of ODP systems. Of the available FDTs, Z is likely to be used for at least the information, and possibly other, viewpoints, whilst LOTOS is a strong candidate for use in the computational viewpoint. Mechanisms are clearly needed to support the parallel development, and integration of, viewpoints using these FDTs. We compare the LOTOS bisimulation relations and the reduction relations to the Z refinement relation showing that failure-traces refinement corresponds closely to refinement in Z.

Key words: Open Distributed Processing; Z; LOTOS; Refinement.

1 Introduction

The aim of this paper is to support the use of FDTs within distributed system design by providing a comparison of the LOTOS and Z refinement relations.

Open Distributed Processing (ODP) [11] is a joint standardisation activity of the ISO and ITU. A reference model has been defined which describes an architecture for building *open* distributed systems. Central to this architecture is a *viewpoints* model. This enables distributed systems to be described from a number of different perspectives. There are five viewpoints: *enterprise*, *information*, *computational*, *engineering* and *technology*. Requirements and specifications of an ODP system can be made from any of these viewpoints.

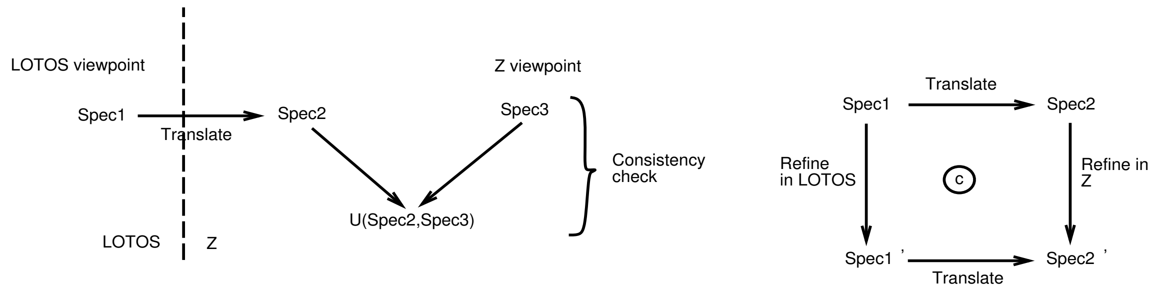
[0]This work was partially funded by British Telecom Labs., Martlesham, Ipswich, U.K. and partially by the U.K. Engineering and Physical Sciences Research Council under grant number GR/K13035.

The ODP reference model (RM-ODP) recognises the need for formalism, with Part 4 of the RM-ODP defining an architectural semantics which describes the application of formal description techniques (FDTs) to the specification of ODP systems. Of the available FDTs, Z is likely to be used for at least the information, and possibly other, viewpoints (the ODP Trader specification is being written using Z for the information viewpoint), whilst LOTOS is a strong candidate for use in the computational and engineering viewpoint.

One of the consequences of adopting a multiple viewpoint approach to specification is that descriptions of the same or related entities can appear in different viewpoints and must co-exist. If viewpoints are to be developed in parallel, how do these developments compare? To answer this we need to know the relationship between refinement relations in differing languages. Here we compare Z and LOTOS.

Inherent in such any viewpoint modelling is the need to check the *consistency* of viewpoints and to show that the different specifications do not impose *contradictory requirements*. Similar consistency properties arise outside ODP, see for example [9]. Previous work has clarified the nature of viewpoints modelling and consistency [4]. A collection of viewpoints is consistent if and only if a common refinement can be found, i.e. a specification that refines all the original viewpoints (each with respect to a particular refinement relation). Of course the choice of refinement relation to apply to each of the different viewpoints is critical.

We have shown how consistency checking may be performed within a single FDT, [1, 3, 6, 16], and how we can translate between FDTs in [7]. The strategy we envisage to check the consistency of one ODP viewpoint written in Z with another written in LOTOS is as follows. First translate the LOTOS specification to an observationally equivalent one in Z, then use the mechanisms defined in [1, 3, 6] to check the consistency of the two viewpoints now both expressed in Z. These mechanisms attempt to find a common Z refinement of the two viewpoints - if one exists the viewpoints are consistent.



An obvious question that now arises is, what is the relationship between the Z refinement relation and the LOTOS refinement relations. Ideally, we would like to find a LOTOS relation which is equivalent to refinement in Z. This is not always possible, and in order to be able to develop the original LOTOS and Z viewpoints further with any confidence, we need to ensure that we use appropriate LOTOS and Z refinement relations, i.e. ones which make the diagram on the right commute. That is, if we refine $Spec1$ to $Spec1'$ in LOTOS, their translations are refinements in Z ($Spec2'$ is a Z refinement in $Spec2$). Thus if after refining the original LOTOS viewpoint we can still find a common refinement, we know that it is a common refinement of the original viewpoints. This will ensure that we can still check for consistency after we have refined the original LOTOS viewpoint, since any subsequent positive consistency check implies the original viewpoints were consistent. This enables us to refine the original LOTOS viewpoint with confidence. Thus, as such, this work builds on a LOTOS to Z translation defined in [7], and this is briefly reviewed in section 2. Section 3 considers basic LOTOS without internal actions, section 4 then considers adjustments needed to the Z refinement relation when considering internal actions. Section 5 discusses full LOTOS, and section 6 concludes the paper.

2 Definitions

In [7] we defined a translation from full LOTOS to Z, which we review briefly here. The essential idea behind the translation is to turn LOTOS processes into ZEST objects, and hence if necessary into Z. Because ODP is object-based, there is a need to provide object-oriented capabilities in FDTs used within ODP. ZEST [5] is an extension to Z to support specification in an object-oriented style, developed by British Telecom specifically to support distributed system specification.

ZEST does not increase the expressive power of Z, and a flattening to Z is provided. What ZEST provides is structuring at a suitable level of abstraction by associating individual operations with one state schema. A *class* is a state schema together with its associated operations and attributes. A class is a template for objects, each object of the class conforms to that class. In many ways ZEST is similar to Object-Z [8], although the latter does not provide a flattening to Z.

Given a LOTOS specification the ADT component is translated directly into the Z type system. For the behaviour expression of the LOTOS specification, we first derive an intermediate semantic model (called an ETS) from the LOTOS, and use this to generate the Z specification. This will involve translating each LOTOS action into a ZEST operation schema with explicit pre- and post-conditions to preserve the temporal ordering.

For example, the LOTOS process $in?x : nat;\ out!(x+2);\ stop$ will be translated into a ZEST object which contains operation schemas with names *in* and *out*. The operation schemas have appropriate inputs and outputs to perform the value passing defined in the LOTOS process. Each operation schema includes a predicate (defined over a state variable) to ensure that it is applicable in accordance with the temporal behaviour of the LOTOS specification.

Example 1 *The LOTOS process*

$$
\begin{array}{l}
\textbf{process}\ \mathrm{compare}[in, out](min, max : int) : \textbf{noexit} := \\
\quad in?x : int; \\
\quad ([min < x < max] \rightarrow out!x;\ \mathrm{compare}[in, out](min, max) \\
\quad [\,][x \leq min] \rightarrow out!min;\ \mathrm{compare}[in, out](x, max) \\
\quad [\,][x \geq max] \rightarrow out!max;\ \mathrm{compare}[in, out](min, x)) \\
\textbf{endproc}
\end{array}
$$

will translate into the ZEST object:

$$
\begin{array}{l}
\textit{compare} \\
\quad s : \{s_0, s_1, s_2, s_3, s_4\} \\
\quad x, min, max : int \\
\\
\quad \textit{INIT} \\
\quad \Delta(s) \\
\quad \hline
\quad s = s_0 \\
\\
\quad \textit{in} \\
\quad \Delta(s), \Delta(x) \\
\quad ch? : int \\
\quad \hline
\quad ch? = x' \wedge (s = s_0 \wedge s' = s_1) \\
\\
\quad \textit{out} \\
\quad \Delta(s) \\
\quad ch! : int \\
\quad \hline
\quad (ch! = x \wedge (s = s_1 \wedge s' = s_2) \wedge (min < x < max)) \vee \\
\quad (ch! = min \wedge (s = s_1 \wedge s' = s_3) \wedge (x \leq min)) \vee \\
\quad (ch! = max \wedge (s = s_1 \wedge s' = s_4) \wedge (x \geq max)) \vee \\
\\
\quad i \\
\quad \Delta(s), \Delta(min, max) \\
\quad \hline
\quad (s = s_2 \wedge s' = s_0) \vee (s = s_3 \wedge s' = s_0 \wedge min' = x) \vee (s = s_4 \wedge s' = s_0 \wedge max' = x)
\end{array}
$$

The temporal ordering is controlled by predicates over the state variable s, and the values $s_0, \ldots$ correspond directly to behaviour expressions at the nodes in an LTS of the process. Recursion is dealt with by using an internal action, which is translated as an internal Z operation with special name i (this is a convention we adopt to deal with internal operations). Input and output are controlled by channels $ch?$ and $ch!$. The key points about the translation for this paper are that LOTOS actions become Z operations, and that these use a state variable (usually denoted s or t) to control the temporal ordering of the operations.

A Z specification describes the state space together with a collection of operations. The Z refinement relation [15], defined between two Z specifications, allows both the state space and the individual operations to be refined in a uniform manner. We employ the convention (adopted in OO versions of Z such as ZEST and Object-Z) that an operation is locked to the environment outside its pre-condition.

Operation refinement is the process of recasting each abstract operation AOp into a concrete operation COp, such that the following holds. The pre-condition of COp may be weaker than the pre-condition of AOp, and COp may have a stronger post-condition than AOp. That is, COp must be applicable (offered to the environment) whenever AOp is, and if AOp is applicable, then every state which COp might produce must be one of those which AOp might produce. We shall see that this refinement corresponds closely to the bisimulation relations in LOTOS.

Data refinement extends operation refinement by allowing the state space of the concrete operations to be different from the state space of the abstract operations (both in terms of size of state space and the types of items defined within the state), and this offers a substantial departure from LOTOS (where no general framework exists to verify the equivalence of LOTOS specifications containing ADT definitions and behaviour expressions). Consider an abstract specification with state space $Astate$, operation AOp, and a refined specification with state space $Cstate$ and operation COp. These operations have input $x? : X$ and output $y! : Y$. A refinement is defined in terms of an abstraction schema or retrieve relation, called Abs, which relates abstract and concrete states. It has the same signature as $Astate \wedge Cstate$, and its property holds if the concrete state is one of those which represent the abstract state [15]. The retrieve relation does not need to be total nor functional.

There are two main conditions which must be satisfied by a Z refinement. The first condition is the applicability condition. The second condition, correctness, ensures that the state after the concrete operation represents one of those abstract states that could be reached by the abstract operation. Formally they state (where the quantification is over all state elements, inputs and outputs):

$$\forall\, Astate;\ Cstate;\ x? : X \bullet \mathrm{pre}\, AOp \wedge Abs \Rightarrow \mathrm{pre}\, COp$$
$$\forall\, Astate;\ Cstate;\ Cstate';\ x? : X;\ y! : Y \bullet \mathrm{pre}\, AOp \wedge Abs \wedge COp \Rightarrow (\exists\, Astate' \bullet Abs' \wedge AOp)$$

Example 2 *As an example of translation and Z refinement, consider the LOTOS processes given by* $Q := a;\ (b;\ c;\ stop[]b;\ d;\ stop)$ *and* $P := (a;\ (b;\ c;\ stop[]b;\ d;\ stop))[]a;\ b;\ c;\ stop$. P *and* Q *translate into the ZEST objects (and hence into Z by the obvious flattening):*

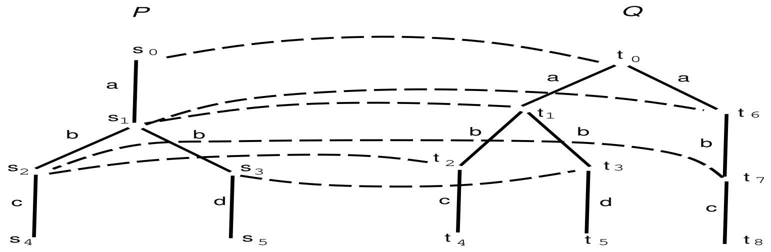
$$
\begin{array}{|l}
\hline
\textit{Q} \\
\begin{array}{|l}
\hline
s : \{s_0, s_1, s_2, s_3, s_4, s_5\} \\
\hline
\end{array} \\
\begin{array}{|l}
\hline
\textit{INIT} \\
s = s_0 \\
\hline
\end{array} \\
a \mathrel{\widehat=} \left[\, \Delta(s) \;\middle|\; s = s_0 \wedge s' = s_1 \,\right] \\
b \mathrel{\widehat=} \left[\, \Delta(s) \;\middle|\; s = s_1 \wedge s' \in \{s_2, s_3\} \,\right] \\
c \mathrel{\widehat=} \left[\, \Delta(s) \;\middle|\; s = s_2 \wedge s' = s_4 \,\right] \\
d \mathrel{\widehat=} \left[\, \Delta(s) \;\middle|\; s = s_3 \wedge s' = s_5 \,\right] \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{P} \\
\begin{array}{|l}
\hline
t : \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\} \\
\hline
\end{array} \\
\begin{array}{|l}
\hline
\textit{INIT} \\
t = t_0 \\
\hline
\end{array} \\
a \mathrel{\widehat=} \left[\, \Delta(t) \;\middle|\; t = t_0 \wedge t' \in \{t_1, t_6\} \,\right] \\
\begin{array}{|l}
\hline
b \\
\Delta(t) \\
\hline
(t = t_1 \wedge t' \in \{t_2, t_3\}) \vee (t = t_6 \wedge t' = t_7) \\
\hline
\end{array} \\
\begin{array}{|l}
\hline
c \\
\Delta(t) \\
\hline
(t = t_2 \wedge t' = t_4) \vee (t = t_7 \wedge t' = t_8) \\
\hline
\end{array} \\
d \mathrel{\widehat=} \left[\, \Delta(t) \;\middle|\; t = t_3 \wedge t' = t_5 \,\right] \\
\hline
\end{array}
$$

Then P is a Z refinement of Q, this is verified against the following retrieve relation:

$$
\begin{array}{|l}
\hline
\textit{Ret} \\
s : \{s_0, s_1, s_2, s_3, s_4, s_5\}, t : \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\} \\
\hline
(s = s_0 \wedge t = t_0) \vee (s = s_1 \wedge t \in \{t_1, t_6\}) \vee (s = s_2 \wedge t \in \{t_2, t_7\}) \vee \\
(s = s_3 \wedge t = t_3) \vee (s = s_4 \wedge t \in \{t_4, t_8\}) \vee (s = s_5 \wedge t = t_5) \\
\hline
\end{array}
$$

which is depicted as dotted lines in the LTS below. Note that in LOTOS, P and Q are testing equivalent, but not bisimular.



The retrieve relation, whose existence is required for a refinement to hold in Z, relates abstract to concrete states in a manner similar to the simulation relation used in bisimulations. We begin our comparison by considering simulation relations in LOTOS and compare them to the Z refinement relation.

Notation: Standard notational usage is different between (and even within) the two languages, and can be confusing. $\sqsubseteq_Z$ is the Z refinement relation; red, ext, $\leq_{2/3}$ and $\leq_{FT}$ are LOTOS refinement relations. The order that the preorders are written differs. So, whereas $Q \sqsubseteq_Z P$ and $Q \leq_{2/3} P$ mean that P is a refinement of Q, A red B, A ext B and $A \leq_{FT} B$ mean that A is a refinement of B. We preserve historical usage, but throughout the paper use P for the refinement and Q for the abstract specification.

3 Basic LOTOS without internal actions

In this section we will consider basic LOTOS without internal actions. Our aim here is to characterise Z refinement in terms of equivalent LOTOS relations. We will first consider simulation based relations, then we will compare reduction relations and other trace-based notions of development.

3.1 Simulation relations

Assuming the usual LOTOS notations, we can define simulation and bisimulation in LOTOS as follows:

Definition 1 *1. A simulation is a binary relation $\mathcal{R}$ such that whenever $P \mathcal{R} Q$ then: $P \xrightarrow{a} P'$ implies $\exists\, Q' : Q \xrightarrow{a} Q'$ and $P' \mathcal{R} Q'$. Two processes are simulation equivalent iff there exists a simulation $\mathcal{R}$ with $P \mathcal{R} Q$ and a simulation $\mathcal{R}'$ with $Q \mathcal{R}' P$.*

2. A binary relation $\mathcal{R}$ is a bisimulation iff both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.

Let us first show that simulation is too weak a relation to imply refinement in Z. By this we mean that there exist LOTOS processes P and Q such that P is a simulation of Q, however, when translated into Z, P is not a refinement of Q.

Example 3 *Let $Q := a;\ b;\ stop$ and $P := (a;\ stop[\,]a;\ b;\ stop)$. Then P is a simulation of Q. However, when translated into Z, P is not a refinement of Q, i.e. no retrieve relation will allow the conditions of refinement to be met, where the translations of Q and P into Z are:*

Q

$s : \{s_0, s_1, s_2\}$

INIT
$$\Delta(s)$$
$$s = s_0$$

a
$$\Delta(s)$$
$$s = s_0 \wedge s' = s_1$$

b
$$\Delta(s)$$
$$s = s_1 \wedge s' = s_2$$

P

$t : \{t_0, t_1, t_2, t_3\}$

INIT
$$\Delta(t)$$
$$t = t_0$$

a
$$\Delta(t)$$
$$t = t_0 \wedge t' \in \{t_1, t_3\}$$

b
$$\Delta(t)$$
$$t = t_1 \wedge t' = t_2$$

However, the situation with bisimulation is more positive. We can prove the following theorem.

Theorem 1 *Bisimulation equivalence implies (mutual) Z refinement. That is if P and Q are bisimular processes then when translated into Z, Q and P are Z-refinements of each other.*

Proof

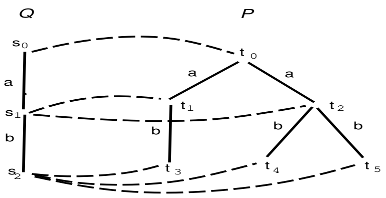
Let $\mathcal{R}$ be the bisimulation relation between P and Q. $\mathcal{R}$ will thus relate behaviour expressions in the derivations of P and Q. We will construct a retrieve relation between the Z translations, based upon $\mathcal{R}$ in the obvious fashion. Let s and t be the state variables of the translation of P and Q into Z respectively. Each derivation of P or Q will correspond to a state of s or t in the state space of the translation of P and Q. The retrieve relation *Ret* has as predicate a series of disjunctions, each of the form $(s = s_i \wedge t = t_j)$ where the derivations corresponding to s_i and t_j are in the bisimulation relation. That is, for each $(S, T) \in \mathcal{R}$ the retrieve relation has as one of its disjunctions $(s = s_i \wedge t = t_j)$ where s_i and t_j are the states corresponding to S and T respectively.

To distinguish Z operations, let us denote the operation a in specification P by Pa et cetera. To show $\mathrm{pre}\, POp \wedge Ret \vdash \mathrm{pre}\, QOp$ for all operations in P, let us suppose that $\mathrm{pre}\, Pa \wedge Ret$ holds for some operation a. Then Pa is applicable at a state which corresponds to behaviour expression C_1 where $(C_1, C_2) \in \mathcal{R}$ for some C_2. Then $C_1 \xrightarrow{a} C_1'$ since the Z operation is applicable iff the LOTOS action could be offered. By definition of bisimulation, there exists a C_2' such that $C_2 \xrightarrow{a} C_2'$ and this implies that Qa is applicable in the translation of Q at the state which corresponds to C_2, i.e. $\mathrm{pre}\, Qa$ holds.

To show that $\mathrm{pre}\, POp \wedge Ret \wedge QOp \vdash \exists\, Pstate' \bullet Ret' \wedge POp$, suppose that $\mathrm{pre}\, Pa$ is true in state C_1 and that Qa is applied at a state C_2 where $(C_1, C_2) \in \mathcal{R}$. Then $C_2 \xrightarrow{a} C_2'$ and thus there exists a C_1' such that $C_1 \xrightarrow{a} C_1'$ with $(C_1', C_2') \in \mathcal{R}$. Hence if $Pstate'$ is the Z state that corresponds to C_1', a disjunction corresponding to (C_1', C_2') will be in the predicate of the retrieve relation, hence $\exists\, Pstate' \bullet Ret' \wedge Pa$ will hold.

Thus Q is a refinement of P, and by symmetry P is a refinement of Q. □

Example 4 *As an example, consider the bisimular processes $P := a;\ b;\ stop[](a;\ (b;\ stop[]b;\ stop))$ and $Q := a;\ b;\ stop$ with the obvious bisimulation relation (depicted in the diagram):*



The translations into Z are refinements of each other, where the retrieve relation is given by

$$\begin{array}{|l}
\hline \textit{Ret} \\
s : \{s_0, s_1, s_2\}, t : \{t_0, t_1, t_2, t_3, t_4, t_5\} \\
\hline
(s = s_0 \wedge t = t_0) \vee (s = s_1 \wedge t \in \{t_1, t_2\}) \vee (s = s_2 \wedge t \in \{t_3, t_4, t_5\}) \\
\hline
\end{array}$$

and the states correspond to the behaviour expressions at the nodes in the LTS as shown above. □

However, bisimulation is stronger than necessary in terms of implying Z refinement, as can be seen from processes P and Q in section 2. P and Q were not bisimular, but after translation into Z, P was a Z refinement of Q. We now consider 2/3-bisimulation [10, 14] introduced by Larsen, which lies between simulation and bisimulation equivalences in terms of strength. 2/3-bisimulation induces an equivalence called 2/3-bisimulation equivalence (bisimulation is itself an equivalence).

Definition 2 *A 2/3-bisimulation is a binary relation $\mathcal{R}$ such that whenever $P\mathcal{R}Q$ then:*

1. $P \xrightarrow{a} P'$ *implies* $\exists\, Q' : Q \xrightarrow{a} Q'$ *and* $P'\mathcal{R}Q'$
2. $Q \xrightarrow{a} \;\Rightarrow P \xrightarrow{a}$

in which case we write $Q \leq_{2/3} P$. Two processes are 2/3-bisimulation equivalent iff there exists a 2/3-bisimulation $\mathcal{R}$ with $P\mathcal{R}Q$ and a 2/3-bisimulation $\mathcal{R}'$ with $Q\mathcal{R}'P$.

Theorem 2 *2/3-bisimulation implies Z refinement. That is if $Q \leq_{2/3} P$, then when translated into Z, P is a Z-refinement of Q.*

Proof
The proof is similar to the proof for bisimulation. The second condition in the definition of 2/3-bisimulation gives us the weakening of pre-conditions allowed in Z refinement, whilst the first and second conditions together allow us to prove correctness. □

The example $Q := a;\ \mathit{stop}$ and $P := a;\ \mathit{stop}[]b;\ \mathit{stop}$ where $Q \sqsubseteq_Z P$ but $Q \not\leq_{2/3} P$ shows we can weaken bisimulation further, and we are led to the following definition, which we call 1/3-bisimulation for processes.

Definition 3 *A 1/3-bisimulation is a binary relation $\mathcal{R}$ such that whenever $P\mathcal{R}Q$ then:*

1. $Q \xrightarrow{a} \wedge P \xrightarrow{a} P'$ *implies* $\exists\, Q' : Q \xrightarrow{a} Q'$ *and* $P'\mathcal{R}Q'$
2. $Q \xrightarrow{a} \;\Rightarrow P \xrightarrow{a}$

in which case we write $Q \leq_{1/3} P$. Two processes are 1/3-bisimulation equivalent iff there exists a 1/3-bisimulation $\mathcal{R}$ with $P\mathcal{R}Q$ and a 1/3-bisimulation $\mathcal{R}'$ with $Q\mathcal{R}'P$.

This allows us to completely characterise the simulation based relations on LOTOS with respect to Z refinement in terms of their relative strengths (in increasing order), where Z refinement corresponds to 1/3-bisimulation for processes (the proof is obvious):

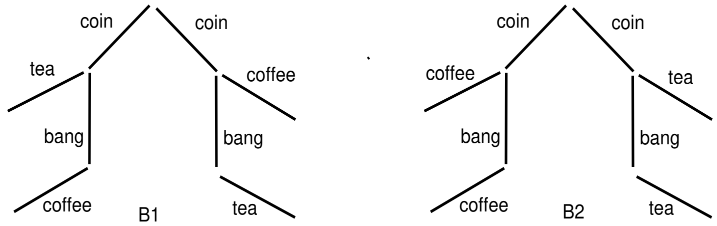
$$\text{simulation} < \text{1/3-bisimulation} \equiv \text{Z-refinement} < \text{2/3-bisimulation} < \text{bisimulation}$$

3.2 Trace based development relations

Another view of development in LOTOS is given by relations based upon sets of traces and refusals. *Conf* is a non-transitive conformance relation, whilst **red** and **ext** are relations based upon the reduction and extension of sets of traces, the latter two relations induce an equivalence called testing equivalence. Assuming the standard notation for refusal sets etc:

Definition 4
Reduction - P **red** Q, *iff* $\forall\, \sigma \in Tr(Q),\ Ref(P,\sigma) \subseteq Ref(Q,\sigma)$ *and* $Tr(P) \subseteq Tr(Q)$;
Extension - P **ext** Q, *iff* $\forall\, \sigma \in Tr(Q),\ Ref(P,\sigma) \subseteq Ref(Q,\sigma)$ *and* $Tr(Q) \subseteq Tr(P)$;
Testing equivalence - $te = \mathbf{red} \cap \mathbf{red}^{-1} = \mathbf{ext} \cap \mathbf{ext}^{-1}$

In this subsection we will consider the relationship between these and related concepts and the Z refinement relation. Intuitively, if P_1**red** P_2, then P_1 has fewer traces than P_2, but even in an environment whose traces are restricted to those of P_1, P_1 deadlocks less often. Hence, reduction has been taken as a natural concept of refinement for LOTOS, particularly given the domain of application. However, as argued in [13, 12] reduction sometimes identifies too many processes, and a more subtle notion of refinement is needed. For example, consider the following two coffee/tea machines:



As Langerak argues, with machine B_1 you always get what you want. After inserting a coin, if you want coffee and the coffee button doesn't work, bang the machine then coffee will be available. However, with machine B_2 you might have to settle for tea even if you wanted coffee.

The above two processes are testing equivalent, however, their translations into Z are not refinements in either direction. Hence, neither **red**, **ext** nor **te** imply Z refinement. Interestingly, they fail to be refinements in Z for the same reasons that Langerak argues they should not be acceptable as reductions in LOTOS.

As a solution to these issues, a new reduction relation is proposed in [13, 12] called failure trace reduction which is based upon a subtler notion of testing than that used for reduction. This relation distinguishes the correct number of processes in order to imply their translations are refinements in Z. We prove this now. First, definitions of Langeraks refinement relation.

Definition 5 *Let P be a process, and $t = a_1 \ldots a_n$ be a trace of P, i.e.* $P \xrightarrow{i^{k0}} P_0 \xrightarrow{a_1 i^{k1}} \ldots P_{n-1} \xrightarrow{a_n i^{kn}} P_n$ *for* $k_i \geq 0$. *Then* $f = A_0 a_1 A_1 a_2 \ldots a_n A_n$ *is a failure trace of P whenever* $A_i \subseteq \mathcal{L}$ *or* $A_i = \epsilon$ *(the neutral element wrt concatenation) and:*

- *if* $P_i \xrightarrow{i}$ *then* $A_i = \epsilon$*;*
- *if* $P_i \stackrel{i}{\not\longrightarrow}$ *then* $A_i = \epsilon$ *or* $A_i \subseteq (\mathcal{L} - out(P_i))$*, where* $out(P) = \{g \in \mathcal{L} \mid P \xrightarrow{g}\}$*.*

The set of all failure traces of P is denoted by FT(P).

Definition 6 *Let* P_1 *and* P_2 *be processes. Then*

$$(P_1 \leq_{FT} P_2) \textit{ iff } (FT(P_1) \subseteq FT(P_2))$$

Theorem 3 *Let* $P \leq_{FT} Q$*, i.e.* P *is a failure-traces refinement of* Q*. Then the Z translation of* P *is a Z-refinement of* Q*.*

Proof

Since $P \leq_{FT} Q$, $FT(P) \subseteq FT(Q)$. Let s and t be the state variable in translations of Q and P respectively. We construct a retrieve relation between the state of Q and the state of P, by linking up states that correspond to failure traces of P. In fact we consider *maximal* failure traces of P, i.e. failure traces that cannot be extended in terms of their trace or their refusal sets. For each subtrace τ_i of such a maximal trace $\sigma \in FT(P)$, let s_i and t_i be corresponding states in Q and P. The retrieve relation will consist of a disjunction of predicates of the form $(s = s_i \wedge t = t_i)$, i.e. given by the correspondences due to all subtraces of all maximal elements of $FT(P)$. The example below shows how the correspondence works, we take maximal elements of $FT(P)$, consider their prefexes, and link these to states in Q where the trace element is the same, but the failures are possibly increased.

With this retrieve relation we will prove the following for all operations Op:

- $\text{pre}\, QOp \wedge Ret \vdash \text{pre}\, POp$
- $\text{pre}\, QOp \wedge Ret \wedge POp \vdash \exists\, Qstate' \bullet Ret' \wedge QOp$

1. Let Op be an operation in Q. Suppose that $\text{pre}\, QOp \wedge Ret$ holds, i.e. QOp is applicable at state $s = s_i$ with $(s = s_i \wedge t = t_i)$ in Ret. Now by definition of Ret, this state corresponds to a failure trace $\sigma_i = \rho A \in FT(P) \subseteq FT(Q)$, where ρ has the form $A_0 a_1 A_1 a_2 \ldots a_n$. Since QOp is applicable at s_i, this means that QOp cannot be refused at s_i, so that $\rho\{QOp\} \notin FT(Q)$, which implies that $\rho\{QOp\} \notin FT(P)$, thus Op is applicable in P at any state corresponding to σ_i, i.e. $\text{pre}\, POp$ holds at state t_i. Hence, $\text{pre}\, QOp \wedge Ret \vdash \text{pre}\, POp$.

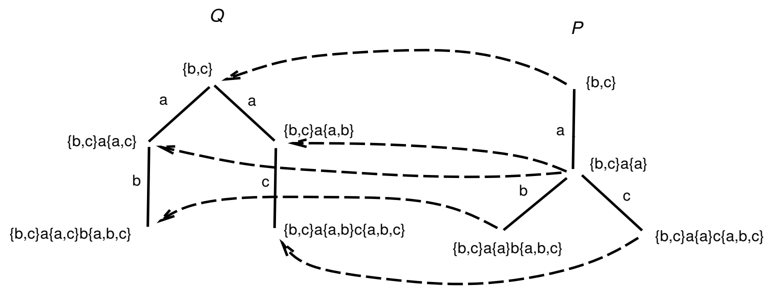
2. Let $\text{pre}\, QOp \wedge Ret$ hold, i.e. QOp is applicable at state $s = s_i$ with $(s = s_i \wedge t = t_i)$ in Ret, and we apply POp. Now s_i and t_i correspond to a $\sigma_i \in FT(P)$, and since POp is applicable, $\sigma_i OpA \in FT(P) \subseteq FT(Q)$ for some refusal set A. Let s_j and t_j correspond to this failure trace. Then $(s' = s_j \wedge t' = t_j)$ is one of the disjunctions of Ret' and s_j is a possible after state of applying QOp, hence the correctness condition holds. $\square$

As a corollary to this theorem we have $\leq_{FT} \subseteq \leq_{1/3}$.

Example 5 *As an illustration of the proof, consider the processes* $P := a;\ (b;\ stop[]c;\ stop)$ *and* $Q := a;\ b;\ stop[]a;\ c;\ stop$*.*

Then $P \leq_{FT} Q$. The maximal failure traces of P are $\{b,c\}a\{a\}b\{a,b,c\}$ and $\{b,c\}a\{a\}c\{a,b,c\}$, and those of Q are $\{b,c\}a\{a,b\}c\{a,b,c\}$ and $\{b,c\}a\{a,c\}b\{a,b,c\}$. The retrieve relation will link up states between P and Q corresponding to the prefixes of the maximal failure traces of P, i.e. $\{b,c\}$, $\{b,c\}a\{a\}$, $\{b,c\}a\{a\}b\{a,b,c\}$ and $\{b,c\}a\{a\}c\{a,b,c\}$.

The retrieve relation will then be given as in the following diagram. The prefixes of the maximal traces are marked on the nodes, and the dotted lines give the inclusions that define the retrieve relation.



Note that the failure-traces refinement defined by Langerak was restricted to finite processes to avoid issues connected with divergence. However, the results carry over to the infinite case since we are restricting ourselves to observable actions here. In the next section we relax this condition and consider how to deal with internal actions when refining in Z.

4 Basic LOTOS with internal actions

In the presence of internal (or unobservable) actions, how do process algebra refinements compare with Z? In LOTOS development the internal action is either treated in the same manner as other actions, and this gives rise to strong bisimulations etc, or it is treated as unobservable and development relations consider processes according to external observations. Classically Z does not have a notion of internal operation, although latterly a number of proposals have been made to model internal operations, particularly in the context of distributed systems. In the LOTOS to Z translation we have mapped the internal action to a Z operation schema with distinguished name i. This is consistent, however, we now have internal operations explicitly in the Z specification and we must consider how to treat refinement of internal operations in Z. There are three options which we will consider in turn:

1. As in strong bisimulation, treat an internal schema like any other, i.e. it must be refined in the same manner as external schemas;

2. Update the refinement relation to deal with internal operation schemas from the point of view of an external observer;

3. Translate out any internal operations, moving the non-deterministic behaviour into the observable schemas before refining the specification. We do not consider this option here. However, the mechanism involves taking a LOTOS specification P, translating it to Z, then eliminating the internal schemas by placing any non-determinism in the observable operations.

Option one corresponds to the strong notions of bisimulation, and the results of section 3.1 carry over directly, e.g. strong 1/3-bisimulation characterises Z refinement.

The second option involves incorporating the notion of internal schema into the Z refinement relation. To do so we consider the standpoint of an *external* observer. Such an external observer will require that a retrieve relation is still defined between the state spaces of the abstract and concrete specifications and that each observable operation AOp is recast as a concrete operation COp. The applicability and correctness criteria are then replaced by their weak counterparts where we allow internal operations in the pre- and post-conditions.

Thus $\mathrm{pre}\, AOp$ is replaced by the condition that AOp is applicable after a number of internal operations. This is described by saying $\exists$ internal operations $i_1, \ldots, i_k$ such that $\mathrm{pre}(i_1 \mathbin{;} \ldots \mathbin{;} i_k \mathbin{;} AOp)$ where $\mathbin{;}$ is schema composition in the Z schema calculus [15]. We abbreviate this to $\mathrm{pre}(i^k \mathbin{;} AOp)$ or $\mathrm{pre}_w(AOp)$.

Similarly for the correctness criteria, in place of the application of an operation Op we allow a finite number of internal operations before and after (i.e. $i^k \mathbin{;} Op \mathbin{;} i^l$), denoting this by Op_w when Op is an observable operation. When Op is an internal operation Op_w denotes i^k (for $k \geq 0$), where i^0 is $\Xi state$. This ensures that we can match up an occurrence of an internal operation in the abstract specification by zero (using $\Xi state$) or more (using i^k) internal actions in the concrete specification. Thus for *weak* refinement we require the following for all operations Op:

- $\mathrm{pre}_w\, AOp \land Ret \vdash \mathrm{pre}_w\, COp$
- $\mathrm{pre}_w\, AOp \land Ret \land COp_w \vdash \exists\, Astate' \bullet Ret' \land AOp_w$

We can then prove analogous results for the weak bisimulation relations and this weak Z refinement by replacing strong derivation $\xrightarrow{a}$ by weak derivation $\xRightarrow{a}$, and noting that in a specification P, $C \xRightarrow{a}$ iff $\exists\, k$ such that $\mathrm{pre}(i^k \mathbin{;} Pa)$ is true at a state corresponding to behaviour expression C in the Z translation. Furthermore, the results in section 3.2 carry over, and in particular we can prove:

Theorem 4 *Let $P \leq_{FT} Q$, i.e. P is a failure-traces refinement of Q, where P and Q are finite basic LOTOS processes (possibly containing internal actions). Then the Z translation of P is a weak Z-refinement of Q.*

Proof

With the similar set up as before, we can prove weak applicability and correctness. The proof is similar to the previous proof. However, to deal with unstable states in P we amend the retrieve relation to include addition links as follows. In a state in P whose failures are a subset of failures of a state in Q, then all states that are immediately previously unstable in P are also related to states in Q. $\Box$

5 Full LOTOS

In this section we consider full LOTOS, i.e. with consideration of value passing and communication. We find that the refinement relations between LOTOS and Z diverge due to the different way in which input and output is treated in the two languages. By input and output in LOTOS we mean the use of variable and value declarations on action denotations. As noted in [2], variable declarations can be regarded as input and value declarations as output. However, input and output are treated fundamentally the same way in the semantics of LOTOS, so the relations **red** and **ext** etc place the same restrictions on them, whereas in Z input and output play fundamentally different roles in the refinement relation.

The fact that variable and value declarations are represented in the same way in the semantics is illustrated by the equivalence of $a?x : t;\ B(x)$ with **choice** $x : t[]a!x;\ B(x)$. So, for example, $a?x : [3];\ B$ is equivalent to $a!3;\ B$. This means that at the semantic level a derivation $\xrightarrow{a<x>}$ could be due to either a value or variable declaration. Because the LOTOS refinement relations are defined at the semantic level in terms of derivations (for either traces or simulations), value and variable declarations will be treated in the same way. In particular, in a LOTOS refinement either no change is allowed to the input or output, or both the input and output can be weakened. In contrast to this, specifications in Z involving input cannot be changed into equivalent specifications involving output, the distinction between input and output is preserved in the semantics.

Example 6 *Extension allows weakening of both the input and output, but reduction allows no change to either value or variable declarations.*

Consider the processes

$$P := a?x : [2..4];\ B(x) \qquad Q := g!2;\ B \qquad P' := a?x : [2..5];\ B(x) \qquad Q' := g!2;\ B[]g!3;\ B$$

Then P' is an extension of P and Q' is an extension of Q. Thus extension allows weakening of both the input and output (this is analogous to the weakening of the pre- and post-conditions in a Z specification).

However, P' is not a reduction of P, nor is $Q' \mathbf{red} Q$, nor would the behaviour $a?x : [2..3];\ B(x)$ be a reduction of P. Hence, reduction allows no change to either value or variable declarations. $\Box$

In contrast to LOTOS, but in common with most state based languages, Z treats input and output differently. A Z refinement can weaken pre-conditions (e.g. input) and strengthen post-conditions (e.g. output). So, for example, the Z specification corresponding to P would be refined by that corresponding to P', (i.e. the pre-condition can be weakened). Conversely, the post-condition can be strengthened in a Z refinement, so Q is a Z refinement of Q'. Note that in Z, this is viewed as reduction of non-determinism, Q' either outputs a 2 or 3 and then behaves like B, and the refinement reduces the non-determinism of the output.

Thus to summarise, LOTOS reduction allows no change of input or output (as defined above), and in this respect is stronger than Z refinement. These arguments extend to failure traces refinement, and thus for full LOTOS *failure traces refinement implies Z refinement under translation.*

Extension allows weakening of both input and output, and hence in general is too weak to imply Z refinement under translation.

The simulation relations are similar. Since weak and strong bisimulation relations require any observable derivation in one specification to be matched by the same derivation in the other, no change in the value and variable declarations is permitted by these relations. Conversely, both 1/3− and 2/3-bisimulation allow input and outputs to be weakened, and thus they *do not imply Z refinement for full LOTOS.* For example, Q' is a 1/3-bisimulation of Q, but the translation of Q' is not a Z refinement of the translation of Q.

6 Conclusions

We have related a number of refinement relations in Z and LOTOS. Considering basic LOTOS without internal actions we completely characterised the simulation based relations by showing:

$$\text{simulation} < 1/3\text{-bisimulation} \equiv \text{Z-refinement} < 2/3\text{-bisimulation} < \text{bisimulation}$$

We also found that neither **red**, **ext** or **te** imply Z refinement. However, we found that if P is a failure-traces refinement of Q, then the Z translation of P is a Z-refinement of Q, i.e. failure-traces refinement implies Z refinement. Classically Z does not have a notion of internal operation, and we extended refinement in Z to treat internal operations in a manner similar to their treatment in LOTOS. Then, with internal actions, the above results all hold using this weak Z refinement.

Finally we considered full LOTOS, and found that both failure traces refinement and bisimulation imply Z refinement under translation. However, this argument does not extend to the other relations, for example 2/3-bisimulation does not imply Z refinement for full LOTOS.

References

[1] 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.

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

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

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

[5] E. Cusack and G. H. B. Rafsanjani. ZEST. In S. Stepney, R. Barden, and D. Cooper, editors, *Object Orientation in Z*, Workshops in Computing, pages 113–126. Springer-Verlag, 1992.

[6] J. Derrick, H. Bowman, and M. Steen. Viewpoints and Objects. In J. P. Bowen and M. G. Hinchey, editors, *Ninth Annual Z User Workshop*, LNCS 967, pages 449–468, Limerick, September 1995. Springer-Verlag.

[7] 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*, Paris, March 1996. Chapman & Hall. To appear.

[8] R. Duke, G. Rose, and G. Smith. Object-Z: A specification language advocated for the description of standards. *Computer Standards and Interfaces*, 17:511–533, September 1995.

[9] A. Fantechi, S. Gnesi, and C. Laneve. Two standards means problems : A case study on formal protocol descriptions. *Computer Standards and Interfaces*, 9:11–19, 1989.

[10] A. Fantechi, S. Gnesi, and G. Ristori. Compositional logic semantics and LOTOS. In L. Logrippo, R. L. Probert, and H. Ural, editors, *Protocol Specification, Testing and Verification, X*, pages 365–378, Ottawa, Canada, June 1990. North-Holland.

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

[12] R. Langerak. A testing theory for LOTOS using deadlock detection. In *Protocol Specification Testing and Verification IX*, pages 87–98. North-Holland, 1989.

[13] R. Langerak. *Transformations and Semantics for LOTOS*. PhD thesis, University of Twente, The Netherlands, 1992.

[14] K. Larsen and A. Skou. Bisimulation through probabilistic testing. In *Sixteenth Annual ACM Symposium on Principles of Programming Languages*. ACM Press, 1989.

[15] J. M. Spivey. *The Z notation: A reference manual.* Prentice Hall, 1989.

[16] M. W. A. Steen, H. Bowman, and J. Derrick. Composition of LOTOS specifications. In P. Dembinski and M. Sredniawa, editors, *Protocol Specification, Testing and Verification, XV*, pages 73–88, Warsaw, Poland, 1995. Chapman & Hall.