29

Supporting ODP - Translating LOTOS to Z

J. Derrick, E.A. Boiten, H. Bowman and M.W.A. Steen [1]

Computing Laboratory, University of Kent, Canterbury, CT2 7NF, UK
Phone: + 44 1227 827570, Fax: + 44 1227 762811,
Email: {J.Derrick,E.A.Boiten,H.Bowman,mwas}@ukc.ac.uk.

Abstract

This paper describes a translation of full LOTOS into Z. A common semantic model is defined and the translation is proved correct with respect to the semantics.

The motivation for such a translation is the use of multiple viewpoints for specifying complex systems defined by the reference model of the Open Distributed Processing (ODP) standardization initiative.

Keywords: Open Distributed Processing; Z; LOTOS; Consistency.

1 INTRODUCTION

The aim of this paper is to support the use of FDTs within distributed system design by providing a translation between full LOTOS and Z.

An important example of open object-based distributed systems is the Open Distributed Processing (ODP) Reference Model. The ODP standardization initiative is a natural progression from OSI, broadening the target of standardization from the point of interconnection to the end-to-end system behaviour. One of the cornerstones of this framework is a model of multiple viewpoints which enables different participants each to observe a system from a suitable perspective and at a suitable level of abstraction.

Formal description techniques (FDTs) are likely to be used for 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 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. *Consistency* of specifications across viewpoints thus becomes a central issue. We have shown how consistency checking may be performed within a single FDT, [BDLS95, DBS95a, DBS95b, SBD95], however, the real challenge lies in checking for consistency across language boundaries, and this requires translation between FDTs.

[1]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.

E. Najm et al. (eds.), *Formal Methods for Open Object-based Distributed Systems*

The work described here makes a first step towards a solution, by defining a translation of full LOTOS into Z using a common semantic model. Section 2 explains the model. Section 3 then provides a semantics for Z in this model. Section 4 then defines the LOTOS to Z translation.

2 EXTENDED TRANSITION SYSTEMS

In [WC89] extended transition systems (ETS) are used to define a semantics for full LOTOS, and we will use them as our common semantic model. An extended transition system provides a semantic model for the data in addition to the control behaviour of a system. Given a signature Σ, and a set of variables V, the set of terms over Σ and V is denoted $T_\Sigma(V)$ (we assume it includes all boolean terms).

Definition 1 *An extended transition system is a 6-tuple* $ETS = \langle S, E, A, R, s_0, f_0 \rangle$ *where* S *is a set of states of the ETS;* $E \subseteq S \times Id$ *is a finite set of extensions on ETS, and* Id *a finite set of identifiers;* A *is a set of actions on ETS (see below);* R *is a set of transition relations on ETS (see below);* s_0 *is the initial state of the system;* f_0 *is the initial assignment of the variables.*

Definition 2 *Let* G *be a set of gates over which an extended transition system can communicate. Actions are elements of* G *with a finite list of attributes: either a value or variable declaration of the form* $!e$*, or a variable declaration of the form* $?v : t$*. Let* I *be a set of internal (unobservable) actions. Elements of* I *are denoted* i*. The set of actions of an ETS is the set*

$$A = \{g?v_1 : t_1 \ldots ?v_m : t_m \; !e_1 \ldots !e_n \mid g \in G \cup I, e_i \in T_\Sigma(V), v_j \in V\}$$

The function name(a) returns the gate name in action a *(either observable or internal).*

Definition 3 *Each element of the set of transition relations* R *is a 5-tuple* $r = \langle a, s, s', p, f \rangle$ *where* a *is an enabling action;* $s, s' \in S$ *are states of the ETS (not necessarily distinct);* $p \in T_\Sigma(V)$ *is an enabling predicate associated with* r*;* $f : V \rightarrow T_\Sigma(V)$ *is an action function associated with* r*.*

The intuitive meaning of a transition relations r is that if the ETS is in state s and the enabling action a is offered, then the enabling predicate is evaluated on the current assignment of variables. When p is true, the ETS will go into the new state s' and the variables are updated by the action function f.

A LOTOS specification of a system defines the temporal relationships among the interactions that constitute the externally observable behaviour of the system [BB88]. A specification consists of two parts: the *behaviour expression* describes the process behaviour and its interaction with the environment whilst the *abstract data type* (ADT) describes the data structures and value expressions.

The translation from LOTOS to ETS given in [WC89] is based on the standard transition derivation system defined in [ISO89] extended to cover data representation and value

passing in full LOTOS. The algorithm generates an extended transition system with a finite set of transition relations.

The transition rules work bottom-up beginning with the LOTOS terminals. A translation algorithm is then developed using the transition rules (full details are given in [WC89]). Chanson also defines a weak bisimulation for extended transition systems. This will be used as the equivalence in our common semantic model.

3 AN ETS SEMANTICS FOR Z

The Z specification language [Spi89] has gained acceptance as one of the viewpoint specification languages for ODP, particularly for the information viewpoint. Because ODP is object-based, there is a need to provide object-oriented capabilities in FDTs used within ODP. ZEST [CR92] 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 has a state which conforms to the class state schema, and is subject to state transitions which conform to the class operations. In many ways ZEST is similar to Object-Z [DRS95], although the latter does not provide a flattening to Z.

The standard semantics for Z is denotational [Spi88]. Consideration of object-oriented issues, however, leads naturally to viewing objects as processes and hence to an *observational* view of the semantics of the specification. Z state changes occur by application of Z operation schemas, thus an observational view regards invocation of a Z operation as a transition in a labelled transition system (LTS).

We will provide an ETS for each ZEST specification, in such a way that a LOTOS specification and its ZEST translation are observationally equivalent in the ETS semantics. The semantics of a ZEST specification is defined to be the ETS of the top level object. We assume that all inheritance has been expanded out in the given ZEST class. The set of variables in the ETS consists of all state variables defined together with all inputs and outputs declared in the operation schemas. The ETS of a ZEST object is derived from considering the application of the last operation schema defined in the object to the ETS derived from the object excluding that schema.

The base ETS

To start, the ETS of an object with no operations is defined. Consider the ZEST object:

P

Attributes
State
Initial_State

the ETS of this is given by $ETS = \langle \{s_0\}, \varnothing, \varnothing, \varnothing, s_0, f_0 \rangle$ where f_0 is the assignment of *Initial_State*, ie the predicate. (The LOTOS translation always produces such an assignment.)

The inductive case

To calculate the effect of operations on the transition system, suppose that the ZEST object P' has an associated ETS of $ETS' = \langle S', E', A', R', s'_0, f'_0 \rangle$. Then we calculate the ETS of the object P (where A is an operation schema) in terms of ETS' in the following fashion, where P' and P are:

P'
Attributes
State
Initial_State
$Operation_1$
⋮
$Operation_n$

P
Attributes
State
Initial_State
$Operation_1$
⋮
$Operation_n$
A

Consider each $s' \in S'$ in turn. Given such an $s' \in S'$, we evaluate pre A at that state (ie on the current assignment of the variables). If A is not applicable, no new relation is added to R' and the ETS is not extended. If A is applicable at s', then a new transition is added to R' and the ETS is extended. We calculate the transitions as follows.

Calculating a transition from an operation schema

An operation schema A given by

A
$\Delta(state\_vars)$
Declarations
OpPred

maps to a transition $r = \langle a, s', s, p, f \rangle$ where

1. $a = A?x_1? : t_1 \ldots ?x_n? : t_n \; !y_1! : u_1 \ldots !y_m! : u_m$ for declarations $x_1? : t_1, \ldots, x_n? : t_n, y_1! : u_1, \ldots, y_m! : u_m$ within A;
2. p is the precondition of OpPred at the current assignment of variables;
3. f gives the effect on state and output variables of performing operations A; and

4. If the effect of f on s' produces an assignment of variables that corresponds to a state $s'' \in S'$, then $s = s''$. If not (or it is undecidable), then a new state, s is added $(S = S' \cup \{s\})$.

For all states added which are not in S', the effect of the object has to be calculated on those states because an existing operation may be applicable at the new state. Therefore all the operations $Op_1, \ldots, Op_n, A$ are applied to these new states to extend the ETS further.

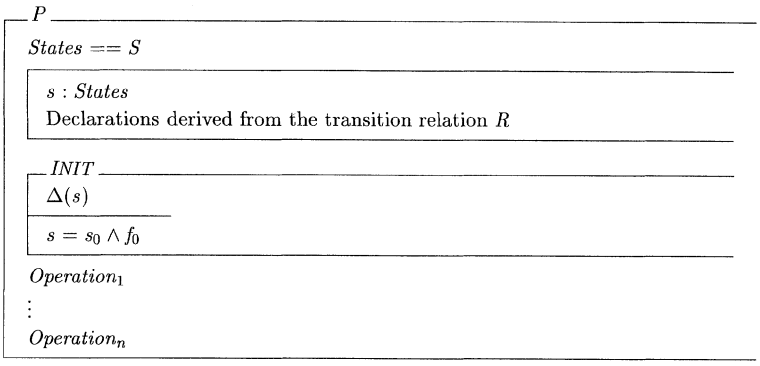
The result of this process is an ETS containing a (not necessarily finite) set of transition relations R. The final ETS consists of the updated set of states and transitions, together with $E = E'$, $A = A' \cup \{a\}$, $s_0 = s'_0$, $f_0 = f'_0$.

4 TRANSLATION FROM FULL LOTOS TO Z

The essential idea behind the translation is to turn LOTOS processes into ZEST objects, and hence if necessary into Z. The ADT component of a LOTOS specification is translated directly into the Z type system. For the behaviour expression of a LOTOS specification, we first derive the 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 (derived from the ETS) to ensure that it is applicable in accordance with the temporal behaviour of the LOTOS specification. Because a finite ETS is generated from any LOTOS specification (see [WC89]), a ZEST specification can be generated which fully describes the LOTOS correctly.

Let $ETS = \langle S, E, A, R, s_0, f_0 \rangle$ be the unique finite extended transition system associated with the LOTOS behaviour expression P. The translation $T(P)$ of the behaviour expression P will be the ZEST object given by:



Operation Schemas

The operation schemas contained within the ZEST object are derived from the finite set of transition relations generated from the LOTOS specification. For each $r \in R$ we generate a (partial) operation schema, and when all relations in R have been considered we merge together operation schemas which have the same name in a manner we describe below.

Let $r = \langle a, s_1, s_2, p, f \rangle \in R$ with $g = name(a)$. Then r will define a template schema of the form:

g

$\Delta(s)$
Declarations derived from a

(transition condition derived from s_1, s_2)$\wedge$
(pre-constraint derived from p)$\wedge$
(post-condition derived from f)

The constituent parts of this are:

1. Transition condition: The transition predicate will be $(s = s_1 \wedge s' = s_2)$.
2. Declarations: An action of the form $g?x_1 : t_1 \ldots ?x_n : t_n !E_1 \ldots !E_m$ is translated to the declaration

g

$\Delta(s), \Delta(x_1, \ldots, x_n)$
$t_1 ch_1? : t_1, \ldots, t_n ch_n? : t_n$
$t_{n+1} ch_{n+1}! : t_{n+1}, \ldots, t_{n+m} ch_{n+m}! : t_{n+m}$

$\ldots$

where $t_{n+i} = type(E_i)$, and the appearance of t_j in a declaration $t_j ch_j?$ or $t_j ch_j!$ is its syntactic representation as a string of characters. This is needed for technical reasons.

In addition, the state schema is amended to include the declarations: $x_1 : t_1, \ldots, x_n : t_n$.

3. Pre-constraint: The pre-constraint is derived from the input/output of an action together with the predicate p. For an action of the form above, the pre-constraint is:

$$(x_1' = t_1 ch_1? \wedge \ldots \wedge x_n' = t_n ch_n?) \wedge (t_{n+1} ch_{n+1}! = E_1 \wedge \ldots \wedge t_{n+m} ch_{n+m}! = E_m) \wedge p[t_1 ch_1?/x_1] \ldots [t_n ch_n?/x_n]$$

where $p[u/v]$ denotes substitution in the standard fashion. A further relabelling is also applied to p and the expressions E_i: for any variable, x say, which is bound when considering the schema alone (ie its binding occurrence occurs at the gate under consideration), any other subsequent occurrence of x in that action are replaced by x'. Furthermore, for any free variable, say y, that appears in the expressions E_i we conjoin $(y = y')$ to the predicate p. An example will make this clear:

(a) $g?x : t_1!(x + 2)$ will become:

$$
\begin{array}{|l}
\hline
g \\
\Delta(s), \Delta(x) \\
t_1 ch_1? : t_1 \\
t_1 ch_2! : t_1 \\
\hline
(x' = t_1 ch_1? \land t_1 ch_2! = (x' + 2)) \\
\ldots \\
\hline
\end{array}
$$

where here the relabelling has been applied to the expression $E_1 = (x + 2)$.

4. Post-condition: By construction, the action function f in the transition relation r will consist of a finite number of assignments of the form $v \leftarrow E$. These are re-written as $v' = E$. Binding occurrences of a variable are relabelled as in the predicate p described above.

Merging Schemas together

Given two partial operation schemas with the same name, built from two different transition relations, we combine them by merging the declarations in the usual fashion (there can be no clashes by construction) and taking the disjunction of the predicates.

For example, given the behaviour $input?x : t;\ a?y : u;\ input!(x + 2)!y;\ stop$, we generate two partial schemas describing the operation *input*:

$$
\begin{array}{|l}
\hline
input \\
\Delta(s), \Delta(x) \\
tch_1? : t \\
\hline
(x' = tch_1? \land s = s_0 \land s' = s_1) \\
\hline
\end{array}
\qquad
\begin{array}{|l}
\hline
input \\
\Delta(s) \\
tch_1! : t, uch_2! : u \\
\hline
(tch_1! = (x + 2) \land uch_2! = y \land s = s_2 \land s' = s_3) \\
\land (x' = x) \land (y' = y) \\
\hline
\end{array}
$$

the combined schema will be:

$$
\begin{array}{|l}
\hline
input \\
\Delta(s), \Delta(x) \\
tch_1? : t \\
tch_1! : t, uch_2! : u \\
\hline
(x' = tch_1? \land s = s_0 \land s' = s_1) \lor \\
((tch_1! = (x + 2) \land uch_2! = y \land s = s_2 \land s' = s_3) \land (x' = x) \land (y' = y)) \\
\hline
\end{array}
$$

To derive a ZEST translation from a LOTOS specification, we apply the translation algorithm to derive a unique finite ETS from the LOTOS specification, then apply the above translation rule to derive the ZEST object.

The translation defined here can be verified against the ETS semantics, i.e. a LOTOS specification and its Z translation will be observationally equivalent in the ETS semantics.

5 CONCLUSIONS

The work described here aims to provide a first step in defining a translation between LOTOS and Z. The translation mechanism was defined, together with a common semantic

framework that verifies the translation algorithm.

Extended transition systems provided the common semantic framework and the relationship between the ETS semantics for LOTOS and the standard LTS semantics needs to be explored. However, although we have used an ETS semantics for LOTOS, any LTS semantics for LOTOS that could be embedded in a finite ETS will produce a translation to Z correct with respect to that semantics.

References

[BB88] T. Bolognesi and E. Brinksma. Introduction to the ISO Specification Language LOTOS. *Computer Networks and ISDN Systems*, 14(1):25–59, 1988.

[BDLS95] H. Bowman, J. Derrick, P. Linington, and M. Steen. FDTs for ODP. *Computer Standards and Interfaces*, 17:457–479, September 1995.

[CR92] 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.

[DBS95a] 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.

[DBS95b] 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.

[DRS95] 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.

[ISO89] ISO. Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour, 1989. IS 8807.

[SBD95] 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.

[Spi88] J. M. Spivey. *Understanding Z: A specification language and its formal semantics.* Cambridge University Press, 1988.

[Spi89] J. M. Spivey. *The Z notation: A reference manual.* Prentice Hall, 1989.

[WC89] 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.