Defect Report concerning ISO International Standard 8807 and Proposal for a Correct Flattening of LOTOS Parameterized Types

Hubert Garavel* Mihaela Sighireanu
INRIA Rhône-Alpes
VERIMAG — Miniparc-ZIRST
rue Lavoisier
38330 MONTBONNOT ST MARTIN
FRANCE

July, 17, 1995

Abstract

We report a defect in the definition of LOTOS (ISO/IEC standard IS-8807). We show that the standard semantics for parameterized and actualized types — although it is consistent in itself — is not satisfactory in case of multiple instantiations of the same generic type, and leads to strange behaviours in existing LOTOS tools.

We indicate how to avoid this problem when writing formal specifications in LOTOS, and we also propose a modification of the flattening mechanism defined in IS-8807 in order to solve this problem.

1 Proposed changes in the International Standard 8807

We propose two modifications of [ISO88], Section 7.3.4.3 (page 42).

1.1 Modification 1

In the following text:

*This work has been supported in part by the European Commission, under project ISC-CAN-65 "EUCALYPTUS-2: A European/Canadian LOTOS Protocol Tool Set".

$gs(< s, t >)$

$= < s_i, T_j >$
if $< s, t > \in FS_1$
and $s = s'_i \quad (1 \leq i \leq p)$
where $1 \leq j \leq n$ and $< s_i, T_j > \in S_2$

$= < s, T_j >$
if $< s, t > \in FS_1$
and $s \neq s'_i \quad (1 \leq i \leq p)$
where $1 \leq j \leq n$ and $< s, T_j > \in S_2$

$= < s_i, T >$
if $< s, t > \in S_1 - FS_1$
and $s = s'_i \quad (1 \leq i \leq p)$

$= < s, t >$
if $< s, t > \in S_1 - FS_1$
and $s \neq s'_i \quad (1 \leq i \leq p)$

$=$ undefined otherwise

replace the piece of text contained in the frame box by the following piece of text:

$= < s, T >$
if $t = T_0$
and $< s, T_0 > \in S_1 - FS_1$
and $s \neq s'_i \quad (1 \leq i \leq p)$

$= < s, t >$
if $t \neq T_0$
and $< s, t > \in S_1 - FS_1$
and $s \neq s'_i \quad (1 \leq i \leq p)$

1.2 Modification 2

In the following text:

$gop(<< op, t >, args, res, pos >)$

$= << op_i, T_j >, gs(args), gs(res), pos >$
if $<< op, t >, args, res, pos > \in FOP_1$
and $op = op'_i \quad (1 \leq i \leq q)$
where $1 \leq j \leq n$ and $<< op_i, T_j >, gs(args), gs(res), pos > \in OP_2$

$= << op, T_j >, gs(args), gs(res), pos >$
if $<< op, t >, args, res, pos > \in FOP_1$
and $op \neq op'_i \quad (1 \leq i \leq q)$
where $1 \leq j \leq n$ and $<< op, T_j >, gs(args), gs(res), pos > \in OP_2$

$= << op_i, T >, gs(args), gs(res), pos >$
if $<< op, t >, args, res, pos > \in OP_1 - FOP_1$
and $op = op'_i \quad (1 \leq i \leq q)$

$= << op, t >, args, res, pos >$
if $<< op, t >, args, res, pos > \in OP_1 - FOP_1$
and $gs(args) = args$ and $gs(res) = res$
and $op \neq op'_i \quad (1 \leq i \leq q)$

$$
\begin{aligned}
&= \ll op, T >, gs(args), gs(res), pos > \\
&\qquad\qquad \text{if} \ll op, t >, args, res, pos > \in OP_1 - FOP_1 \\
&\qquad\qquad \text{and } gs(args) \neq args \text{ and } gs(res) \neq res \\
&\qquad\qquad \text{and } op \neq op'_i \quad (1 \leq i \leq q) \\
&= \text{undefined otherwise}
\end{aligned}
$$

replace the piece of text contained in the frame box by the following piece of text:

$$
\boxed{
\begin{aligned}
&= \ll op, T >, args, res, pos > \\
&\qquad\qquad \text{if } t = T_0 \\
&\qquad\qquad \text{and} \ll op, t >, args, res, pos > \in OP_1 - FOP_1 \\
&\qquad\qquad \text{and } gs(args) = args \text{ and } gs(res) = res \\
&\qquad\qquad \text{and } op \neq op'_i \quad (1 \leq i \leq q) \\
&= \ll op, t >, args, res, pos > \\
&\qquad\qquad \text{if } t \neq T_0 \\
&\qquad\qquad \text{and} \ll op, t >, args, res, pos > \in OP_1 - FOP_1 \\
&\qquad\qquad \text{and } gs(args) = args \text{ and } gs(res) = res
\end{aligned}
}
$$

2 Rationale of the proposed changes

In this section, we explain the motivations underlying the proposed changes. We first recall the basic notations and concepts used in [ISO88]. Then we give two examples of the problems raised by the existing semantics (one example for sorts and the other for operations).

Then we suggest a way to avoid this undesirable problem, still remaining in the framework of LOTOS standard semantics. Finally, we intuitively justify the proposed changes in the semantics.

2.1 Notations and concepts

The following definitions, taken from [ISO88], will also be used in the sequel of the present document.

A signature is a tuple $< S, OP >$, where S is a set of sorts and OP is a set of operations. In a signature, a given sort is represented by a 2-tuple:

$$< \text{sort\_identifier}, \text{definition\_type} >$$

and a given operation is represented by a 4-tuple:

$$\ll \text{operation\_identifier}, \text{definition\_type} >, args, res, pos >$$

where:

- *args* is the tuple of the sorts the arguments,
- *res* is the sort of the result,
- *pos* have *infix* or *prefix* values.

An algebraic specification is a 3-tuple $SPEC = < S, OP, E >$, where $< S, OP >$ is a signature, and E is a set of conditional equations.

A data-presentation $pres = < S', OP', E' >$ is an algebraic specification $< S, OP, E >$ whose sorts, operations and equations are labelled by the identifier of the type in which they are defined.

A parameterized data-presentation $ppres = < fpres, tpres >$ is a pair of data-presentations consisting of a *formal* data-presentation *fpres* and a *target* data-presentation *tpres*, where the formal data-presentation is included component-wise in the target data-presentation.

2.2 An example of the problem arising with the sorts

We consider the example of a generic list type, which is actualized twice to obtain a list of booleans and a list of naturals.

```
type GENERIC_LIST is
  formalsorts ITEM
  sorts LIST
  opns
     NIL    :  -> LIST
     CONS   :  ITEM, LIST -> LIST
endtype

type LIST_BOOL is GENERIC_LIST actualizedby BOOLEAN using
    sortnames BOOL for ITEM
endtype

type LIST_NAT is GENERIC_LIST actualizedby NATURAL using
    sortnames NAT for ITEM
endtype
```

According to Section 7.3.4.3 of [ISO88], the parameterized presentations obtained after applying the "flattening" algorithm to this example are:

- for the type GENERIC_LIST:

$$\begin{array}{rcl} ppres & = & <<< \mathrm{ITEM}, \mathrm{GENERIC\_LIST} >, \emptyset, \emptyset >, \\ & & < \{< \mathrm{ITEM}, \mathrm{GENERIC\_LIST} >, < \mathrm{LIST}, \mathrm{GENERIC\_LIST} >\}, \\ & & \{<< \mathrm{NIL}, \mathrm{GENERIC\_LIST} >, \emptyset, \mathrm{LIST}, \mathit{prefix} >, \\ & & << \mathrm{CONS}, \mathrm{GENERIC\_LIST} >, < \mathrm{ITEM}, \mathrm{LIST} >, \mathrm{LIST}, \mathit{prefix} >\}, \emptyset > \end{array}$$

- for the type LIST_BOOL:

$$\begin{array}{rcl} ppres & = & ppres_{\mathrm{BOOLEAN}} \\ & \cup & <<>, \\ & & << \mathrm{LIST}, \mathrm{GENERIC\_LIST} >, \\ & & \{<< \mathrm{NIL}, \mathrm{GENERIC\_LIST} >, \emptyset, \mathrm{LIST}, \mathit{prefix} >, \\ & & << \mathrm{CONS}, \mathrm{LIST\_BOOL} >, < \mathrm{BOOL}, \mathrm{LIST} >, \mathrm{LIST}, \mathit{prefix} >\}, \emptyset > \end{array}$$

- for the type LIST_NAT:

$$\begin{array}{rcl} ppres & = & ppres_{\mathrm{NATURAL}} \\ & \cup & <<>, \\ & & << \mathrm{LIST}, \mathrm{GENERIC\_LIST} >, \\ & & \{<< \mathrm{NIL}, \mathrm{GENERIC\_LIST} >, \emptyset, \mathrm{LIST}, \mathit{prefix} >, \\ & & << \mathrm{CONS}, \mathrm{LIST\_NAT} >, < \mathrm{NAT}, \mathrm{LIST} >, \mathrm{LIST}, \mathit{prefix} >\}, \emptyset > \end{array}$$

Two problems arise:

1. There is a "semantic clash" between different objects. In the presentation, it is not possible to distinguish between the sort LIST defined in type GENERIC_LIST, the sort LIST obtained by actualization in type LIST_BOOL, and the sort LIST obtained by actualization in type LIST_NAT. Indeed, according to [ISO88] semantics, these three objects have the same name LIST and the same definition type GENERIC_LIST.

2. Due to this clash, there are four different operations returning a result of sort LIST:

$$
\begin{array}{l}
< \mathrm{NIL}, \mathrm{GENERIC\_LIST} > \\
< \mathrm{CONS}, \mathrm{GENERIC\_LIST} > \\
< \mathrm{CONS}, \mathrm{LIST\_BOOL} > \\
< \mathrm{CONS}, \mathrm{LIST\_NAT} >
\end{array}
$$

   This leads existing LOTOS tools dealing with parameterized types (e.g., TOPO [ndM88]) to implement these three different sorts as a single one.

2.3 Another example of the problem arising with the operations

A symmetric problem exists for operations, as illustrated by the following example:

```
type NEW_GENERIC_LIST is GENERIC_LIST, BOOLEAN
  opns
     CAR  :  LIST -> ITEM
     OK   :  LIST -> BOOL
  eqns
    forall I: ITEM, L: LIST
    ofsort BOOL
    CAR (L) eq I => OK (L) = true;
    CAR (L) ne I => OK (L) = false;
endtype

type NEW_LIST_BOOL is NEW_GENERIC_LIST actualizedby BOOLEAN using
    sortnames BOOL for ITEM
endtype

type NEW_LIST_NAT is NEW_GENERIC_LIST actualizedby NATURAL using
    sortnames NAT for ITEM
endtype
```

According to [ISO88], the parameterized presentations for this three types are:

- for the type NEW_GENERIC_LIST:

$$
\begin{array}{rcl}
ppres & = & ppres_{\mathrm{GENERIC\_LIST}} \\
 & \cup & ppres_{\mathrm{BOOLEAN}} \\
 & \cup & <<>, \\
 & & < \emptyset, \{<< \mathrm{CAR}, \mathrm{NEW\_GENERIC\_LIST} >, < \mathrm{LIST} >, \mathrm{ITEM}, \mathit{prefix} >, \\
 & & << \mathrm{OK}, \mathrm{NEW\_GENERIC\_LIST} >, < \mathrm{LIST} >, \mathrm{BOOL}, \mathit{prefix} >\}, \\
 & & \{ \textbf{forall}\ \ \mathrm{I: ITEM, L: LIST} \\
 & & \quad \mathrm{CAR\ (L)\ eq\ I} => \mathrm{OK\ (L)} = \mathrm{true}; \\
 & & \quad \mathrm{CAR\ (L)\ ne\ I} => \mathrm{OK\ (L)} = \mathrm{false};\} >>
\end{array}
$$

- for the type NEW_LIST_BOOL:

$$
\begin{array}{lcl}
ppres & = & ppres_{\mathrm{BOOLEAN}} \\
 & \cup & <<>, \\
 & & < \emptyset, \{<< \mathrm{CAR}, \mathrm{NEW\_LIST\_BOOL} >, < \mathrm{LIST} >, \mathrm{BOOL}, \mathit{prefix} >, \\
 & & << \mathrm{OK}, \mathrm{NEW\_GENERIC\_LIST} >, < \mathrm{LIST} >, \mathrm{BOOL}, \mathit{prefix} >\}, \\
 & & \{\ \textbf{forall}\ \ \text{I: BOOL, L: LIST} \\
 & & \quad \text{CAR (L) eq I => OK (L) = true;} \\
 & & \quad \text{CAR (L) ne I => OK (L) = false;}\} >>
\end{array}
$$

- for the type NEW_LIST_NAT:

$$
\begin{array}{lcl}
ppres & = & ppres_{\mathrm{BOOLEAN}} \\
 & \cup & ppres_{\mathrm{NATURAL}} \\
 & \cup & <<>, \\
 & & < \emptyset, \{<< \mathrm{CAR}, \mathrm{NEW\_LIST\_NAT} >, < \mathrm{LIST} >, \mathrm{NAT}, \mathit{prefix} >, \\
 & & << \mathrm{OK}, \mathrm{NEW\_GENERIC\_LIST} >, < \mathrm{LIST} >, \mathrm{BOOL}, \mathit{prefix} >\}, \\
 & & \{\ \textbf{forall}\ \ \text{I: NAT, L: LIST} \\
 & & \quad \text{CAR (L) eq I => OK (L) = true;} \\
 & & \quad \text{CAR (L) ne I => OK (L) = false;}\} >>
\end{array}
$$

The static semantics rules for actualization allow to distinguish between the operation CAR defined in type NEW_GENERIC_LIST, the operation CAR defined in type NEW_LIST_BOOL and the operation CAR defined in type NEW_LIST_NAT. This is due to the fact that the profile of the operation CAR defined in type NEW_GENERIC_LIST contains the formal sort ITEM.

Unfortunately, this is not the case for operation OK, the profile of which does not contain the sort ITEM. There is a semantic clash, since only a single operation OK, defined in type NEW_GENERIC_LIST, can be distinguished.

Moreover, there are six equations attached to this unique operation OK ! These equations are the following:

```
forall  I: ITEM, L: LIST
   CAR (L) eq I => OK (L) = true;
   CAR (L) ne I => OK (L) = false;
forall  I: BOOL, L: LIST
   CAR (L) eq I => OK (L) = true;
   CAR (L) ne I => OK (L) = false;
forall  I: NAT, L: LIST
   CAR (L) eq I => OK (L) = true;
   CAR (L) ne I => OK (L) = false;
```

2.4 A way to avoid these problems when using standard LOTOS

The problems described in Sections 2.2 and 2.3 happen as soon as a parameterized type is actualized twice without renaming its local sorts and operations with the "**for**" clause.

Thus, the problem can be avoided if, in all actualizations, "**for**" clauses are systematically used to rename all sorts and operations defined in the parameterized type.

For example, a correct version of the example given in Section 2.2 would be:

```
type GENERIC_LIST is
  (* unchanged *)
endtype
type LIST_BOOL is GENERIC_LIST actualizedby BOOLEAN using
    sortnames BOOL for ITEM
              BLIST for LIST   (* this was added *)
endtype

type LIST_NAT is GENERIC_LIST actualizedby NATURAL using
    sortnames NAT for ITEM
              NLIST for LIST   (* this was added *)
endtype
```

By doing so, there is no clash between sorts LIST, BLIST, and NLIST, since each of them is given a unique identifier.

Similarly, a correct version of the example given in Section 2.3 would be:

```
type NEW_GENERIC_LIST is
  (* unchanged *)
endtype

type NEW_LIST_BOOL is NEW_GENERIC_LIST actualizedby BOOLEAN using
    sortnames BOOL for ITEM
    opnnames BOK for OK   (* this was added *)
endtype

type NEW_LIST_NAT is NEW_GENERIC_LIST actualizedby NATURAL using
    sortnames NAT for ITEM
    opnnames NOK for OK   (* this was added *)
endtype
```

However, if the user forgets to use the "**for**" clause, the static semantic rules of [ISO88] might lead to semantic clashes, without any error or warning messages.

2.5 Justification of the proposed changes

The changes proposed in Section 1 have an intuitive justification.

The underlying idea is the following one: every time that a clause "X **for** Y" is not explicitly given for a sort Y or an operation Y local to the parameterized type, the modified semantics implicitly adds a clause of the form "Y **for** Y".

By example, the actualization of parameterized type GENERIC_LIST with booleans (2.2 (p. 4)):

```
type LIST_BOOL is GENERIC_LIST actualizedby BOOLEAN using
    sortnames BOOL for ITEM
endtype
```

is strictly equivalent, under our modified semantics, to:

```
type LIST_BOOL is GENERIC_LIST actualizedby BOOLEAN using
    sortnames
        BOOL for ITEM
        LIST for LIST
endtype
```

Similarly, the actualization of parameterized type NEW_GENERIC_LIST with booleans (2.3 (p. 5)):

```
type NEW_LIST_BOOL is NEW_GENERIC_LIST actualizedby BOOLEAN using
    sortnames BOOL for ITEM
endtype
```

is equivalent to:

```
type NEW_LIST_BOOL is NEW_GENERIC_LIST actualizedby BOOLEAN using
    sortnames
        BOOL for ITEM
        CAR for CAR
        OK for OK
endtype
```

Practically, adding implicitly a "Y **for** Y" clause has the effect of declaring a new object (sort or operation) Y in the actualized type.

This new object Y does not clash with the object Y already defined in the parameterized type, because they do not have the same definition type. Thus, the semantic clashes exhibited in Section 2.2 and 2.3 are avoided.

It is worth noticing that our modified semantics is close to the one originally proposed in the Draft International Standard for LOTOS [ISO87]. For instance, in the case of type LIST_BOOL, the definition of the LIST sort obtained after actualization is LIST_BOOL (as in the DIS-8807 semantics) instead of GENERIC_LIST (as in IS-8807 semantics).

However, our modified semantics is more advanced than the DIS-8807 semantics, because it accept the renaming of non-formal sorts and operations during actualization (a feature introduced in IS-8807).

Conclusions

We have exhibited a problem in IS-8807 related to the actualization of abstract data types.

We have shown that this problem leads existing tools (such as TOPO) to generate meaningless code.

We have proposed a specification discipline to avoid this problem, still keeping IS-8807 unchanged.

We have also proposed a simple and intuitively appealing change in LOTOS static semantics to solve this problem.

We believe that it is easy to adapt existing tools in order to incorporate our modified semantics.

References

[ISO87] ISO. LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. Draft International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, July 1987.

[ISO88] ISO. LOTOS — A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour. International Standard 8807, International Organization for Standardization — Information Processing Systems — Open Systems Interconnection, Genève, September 1988.

[ndM88] J. A. Ma nas and T. de Miguel. From LOTOS to C. In Kenneth J. Turner, editor, *Proceedings of the 1st International Conference on Formal Description Techniques FORTE'88 (Stirling, Scotland)*, pages 79–84, Amsterdam, September 1988. North-Holland.