A generalization of the E-LOTOS trapping operator: towards reconciliation with LOTOS

M. Kapus-Kolar [1]

Jožef Stefan Institute, POB 3000, SI-1001 Ljubljana, Slovenia

Abstract

LOTOS and its enhanced version E-LOTOS are standard languages for formal specification of concurrent and reactive systems. For better backward compatibility of E-LOTOS with LOTOS, we introduce into E-LOTOS specification and trapping of non-urgent and potentially decisive successful termination of processes.

Key words: formal methods, LOTOS, E-LOTOS

1. Introduction

LOTOS [4,1] is a standard language for formal specification of concurrent and reactive systems. A LOTOS specification represents a process by its external behaviour, i.e. by its visible events and their ordering. Various process composition operators are provided for structured specification of systems.

Recently, LOTOS has been enhanced into E-LOTOS [5,9], a language for specification of real-time systems. As E-LOTOS is much richer and more user friendly than LOTOS, many LOTOS users will want to switch to E-LOTOS and associated tools, expecting that translation of a LOTOS specification to E-LOTOS is a trivial matter, so that they can still use their old LOTOS specifications and their usual specification styles.

[5] in its Annex B indeed provides guidelines for such translation. However, as hinted in the Introduction of [5], the basic E-LOTOS operational semantics is in some details not compatible with that of LOTOS. That is most inconvenient, because it is an incompatibility in the core of the languages. To solve the problem, we propose a slight extension of E-LOTOS to allow, like in LOTOS, decisive successful termination of processes. The idea is explained in the simplest possible setting where actions carry no data and every time step is a **tick**.

2. Successful Termination and Sequential Control Transfer in LOTOS

In LOTOS, a step S of a process B is always an action A, that might be an internal action **i**, a gate action G or a successful termination δ. An **i** can be specified directly or result from action hiding. A δ is specified by "**exit**".

Any action is allowed to have alternative steps, i.e. to be decisive for a particular process. That

[1] Email: monika.kapus-kolar@ijs.si

also applies to δ. In a "$B_1 [] B_2$" (choice), an initial δ of B_1 or B_2 resolves the choice between B_1 and B_2, as any other initial action would.

$$\frac{B_1 \xrightarrow{\delta} B_1'}{B_1 [] B_2 \xrightarrow{\delta} B_1'} \qquad \frac{B_2 \xrightarrow{\delta} B_2'}{B_1 [] B_2 \xrightarrow{\delta} B_2'}$$

In a "$B_1 [> B_2$" (disabling), an initial δ of B_2 disables B_1, as any other start of B_2 would. On the other hand, a δ in B_1 discards the disabling process B_2.

$$\frac{B_1 \xrightarrow{\delta} B_1'}{B_1 [> B_2 \xrightarrow{\delta} B_1'} \qquad \frac{B_2 \xrightarrow{\delta} B_2'}{B_1 [> B_2 \xrightarrow{\delta} B_2'}$$

In a "$B_1 |[G_1 \ldots G_n]| B_2$" (parallel composition), the concurrent processes B_1 and B_2 are synchronized not only on $G_1 \ldots G_n$, but by definition also on δ, i.e. successful termination of the composite process can only be a common action of B_1 and B_2.

$$\frac{B_1 \xrightarrow{\delta} B_1', B_2 \xrightarrow{\delta} B_2'}{B_1 |[G_1 \ldots G_n]| B_2 \xrightarrow{\delta} B_1' |[G_1 \ldots G_n]| B_2'}$$

There are two kinds of sequential control transfer. In an "$\mathbf{i}; B_2$" or "$G; B_2$" (action prefix), control is transferred to B_2 upon the initial action. In a "$B_1 \gg B_2$" (enabling), control is transferred from B_1 to B_2 upon δ in B_1, where the action is hidden from the environment.

$$\mathbf{i}; B_2 \xrightarrow{\mathbf{i}} B_2 \qquad G; B_2 \xrightarrow{G} B_2 \qquad \frac{B_1 \xrightarrow{\delta}}{B_1 \gg B_2 \xrightarrow{\mathbf{i}} B_2}$$

3. Successful Termination and Sequential Control Transfer in E-LOTOS

E-LOTOS is a timed language, i.e. an S might also be a time step **tick**. A process with no step can be specified by "**block**".

There are urgent and non-urgent actions. An urgent action A of a process B cannot have **tick** as its alternative, i.e. B is not allowed to idle when A is pending. An **i** or a δ is by definition urgent, while a G is by definition non-urgent. An urgent action might also be a signal X. A self-standing δ can be specified by "**null**".

$$\mathbf{null} \xrightarrow{\delta} \mathbf{block}$$

Parallel processes are in principle not allowed to synchronize on urgent actions, because prompt execution of an urgent action could otherwise be prevented by lack of co-operation of the concurrent processes. Still, parallel processes are, like in LOTOS, by definition synchronized on δ, i.e. processes are allowed to delay an otherwise urgent δ as long as absolutely necessary.

The LOTOS action prefix and enabling operators have been unified into a sequential composition operator. In a "$B_1; B_2$", B_2 becomes enabled when B_1 *becomes ready* for δ. In other words, B_2 is started as the handler for δ *trapped* in B_1. Remember that in LOTOS, B_2 becomes enabled when B_1 *executes* δ. To extend the concept of trapping to action prefix, E-LOTOS defines that in an "$\mathbf{i}; B_2$", "$G; B_2$" or "**signal** $X; B_2$", B_2 is enabled upon execution of the initial action because an **i**, G or X by definition enables δ. In other words, an "**i**", "G" or "**signal** X" in E-LOTOS specifies a successfully terminating process.

$$\mathbf{i} \xrightarrow{\mathbf{i}} \mathbf{null} \qquad G \xrightarrow{G} \mathbf{null} \qquad \mathbf{signal}\ X \xrightarrow{X} \mathbf{null}$$

$$\frac{B_1 \xrightarrow{\delta}, B_2 \xrightarrow{S} B_2'}{B_1; B_2 \xrightarrow{S} B_2'}$$

Action trapping for E-LOTOS has originally been proposed for exception handling [2], but immediately recognized as widely applicable. In the current version of E-LOTOS, trapping is allowed for δ (sequential composition) and for signals (they typically indicate an exception).

An abstract syntax for trapping actions in a $\mathcal{T}$ (a set of trappable actions T) in a process B and the rule for transfer of control to a $B_{T'}$, the handler of a T' in B, are as follows:

$$\frac{B \xrightarrow{T'}, B_{T'} \xrightarrow{S} B_{T'}'}{\mathbf{trap}\ \{(T\ \mathbf{is}\ B_T) | (T \in \mathcal{T})\}\ \mathbf{in}\ B\ \mathbf{endtrap} \xrightarrow{S} B_{T'}'} [T' \in \mathcal{T}]$$

If the trapped T' has an alternative in B, the current E-LOTOS rules for trapping might result in nondeterministic ageing of the composite process, violating the declared time determinacy of E-LOTOS. E-LOTOS solves the problem by ensuring that trappable actions never have alternatives, that implying their urgency.

Because δ might be decisive in LOTOS, the above policy introduces several incompatibilities

between LOTOS and E-LOTOS. Within the context of a "$B_1 [] B_2$", an initial δ of B_1 or B_2 is no longer executable. Actually, it is according to [5] even illegal to specify an initially terminated alternative, but we take a more flexible view that specifying a non-executable action can do no harm. Similarly, within the context of a "$B_1 [> B_2$", an initial δ of B_2 is no longer executable, while a δ in B_1 is prefixed by an additional **i** representing the discarding of B_2.

$$\frac{B_1 \stackrel{\delta}{\rightarrow}}{B_1 [> B_2 \stackrel{\mathbf{i}}{\rightarrow} \mathbf{null}}$$

One can easily imagine that a designer has written a LOTOS specification of the form "$(\mathbf{exit} [] B_1) \gg \ldots \gg (\mathbf{exit} [] B_n)$", representing a series of omittable processes B_1 to B_n. A naturally expected translation to E-LOTOS would be "$(\mathbf{null} [] B_1); \ldots; (\mathbf{null} [] B_n)$", but that, if not illegal, would be equivalent to "$B_1; \ldots; B_n$". A quite appropriate solution would be to replace every **exit** by **i**.

However, we now give an example for which there seems to be no such easy way out. Take a LOTOS process "$(B_{11} [> B_{12}) |[G]| (B_{21} [> B_{22})$", where there is no G in B_{11} and B_{21}, while B_{12} and B_{22} always co-operate on it, though G is not their first action. The two parallel processes may terminate without activating B_{12} and B_{22}, but only if they collectively decide so, by synchronizing upon successful termination of B_{11} and B_{21}. Even if substituting **null** for every **exit** yields a legal E-LOTOS specification, the resulting process might deadlock. For example, the first process might individually discard B_{12} after B_{22} has already started and waits for synchronization with B_{12} on G. That is possible because in E-LOTOS, the discarding occurs upon an **i**. Obviously, the prohibition of decisive δ is a serious problem for transition from LOTOS to E-LOTOS.

4. LOTOS Successful Termination as Non-urgent Successful Termination in E-LOTOS

As the first step towards more natural inclusion of LOTOS into E-LOTOS with respect to δ, we introduce the LOTOS **exit** as a special E-LOTOS behaviour. Let **exit** in E-LOTOS specify a non-urgent δ, in contrast to the urgent δ specified by **null**.

$\mathbf{exit} \stackrel{\mathbf{tick}}{\rightarrow} \mathbf{exit} \qquad \mathbf{exit} \stackrel{\delta}{\rightarrow} \mathbf{block}$

Since an E-LOTOS δ is always without alternatives, we define that a LOTOS δ is a δ with alternatives. It is important that within a pure LOTOS-style specification, a LOTOS δ can never lose its **tick** alternative without gaining another alternative. Likewise, an E-LOTOS δ can never gain an alternative within a pure E-LOTOS-style specification. When including a LOTOS-style B into an E-LOTOS specification, or an E-LOTOS-style B into a LOTOS-style specification, one can avoid confusion by first adapting its δ, i.e. instead of including B, one should better include "$B; \mathbf{null}$" or "$B; \mathbf{exit}$", respectively.

A LOTOS δ is supposed to resolve choice:

$$\frac{B_1 \stackrel{\delta}{\rightarrow} B_1', \exists S \neq \delta : B_1 \stackrel{S}{\rightarrow}}{B_1 [] B_2 \stackrel{\delta}{\rightarrow} B_1'} \qquad \frac{B_2 \stackrel{\delta}{\rightarrow} B_2', \exists S \neq \delta : B_2 \stackrel{S}{\rightarrow}}{B_1 [] B_2 \stackrel{\delta}{\rightarrow} B_2'}$$

Such a δ is also supposed to be decisive within a "$B_1 [X > B_2$", the E-LOTOS generalization of "$B_1 [> B_2$". A "$B_1 [X > B_2$" specifies process B_1 repeatedly suspended upon the start of B_2 and resumed whenever B_2 upon X returns into its initial state.

$$\frac{B_1 \stackrel{\delta}{\rightarrow} B_1', \exists S \neq \delta : B_1 \stackrel{S}{\rightarrow}}{B_1 [X > B_2 \stackrel{\delta}{\rightarrow} B_1'} \qquad \frac{B_2 \stackrel{\delta}{\rightarrow} B_2', \exists S \neq \delta : B_2 \stackrel{S}{\rightarrow}}{B_1 [X > B_2 \stackrel{\delta}{\rightarrow} B_2'}$$

We must also rewrite the original E-LOTOS rule for E-LOTOS δ in B_1:

$$\frac{B_1 \stackrel{\delta}{\rightarrow}, \forall S \neq \delta : B_1 \stackrel{S}{\not\rightarrow}}{B_1 [X > B_2 \stackrel{\mathbf{i}}{\rightarrow} \mathbf{null}}$$

5. Trapping of Decisive Actions in E-LOTOS

Any δ, even if it is decisive, is primarily intended for trapping. As the E-LOTOS trapping operator cannot properly trap decisive actions, we propose its generalization. Here let us note that in E-LOTOS, a trap for δ in a B is originally specified by an "**exit is** B_δ". That should be understood as also applying to δ specified in B by **null**.

The first rule defines normal execution of the non-trapped actions of B:

$$\frac{B \xrightarrow{A} B'}{\textbf{trap } \{(T \textbf{ is } B_T)|(T \in \mathcal{T})\} \textbf{ in } B \textbf{ endtrap} \xrightarrow{A} \textbf{trap } \{(T \textbf{ is } B_T)|(T \in \mathcal{T})\} \textbf{ in } B' \textbf{ endtrap}} \quad [A \notin \mathcal{T}]$$

The second rule defines transfer of control because a T' with no alternative in B has been enabled (the E-LOTOS-type transfer).

$$\frac{B \xrightarrow{T'}, \forall S \neq T' : B \not\xrightarrow{S}, B_{T'} \xrightarrow{S'} B'_{T'}}{\textbf{trap } \{(T \textbf{ is } B_T)|(T \in \mathcal{T})\} \textbf{ in } B \textbf{ endtrap} \xrightarrow{S'} B'_{T'}} \quad [T' \in \mathcal{T}]$$

The third rule defines transfer of control upon hidden execution of a T' with an alternative in B (the LOTOS-type transfer).

$$\frac{B \xrightarrow{T'}, \exists S \neq T' : B \xrightarrow{S}}{\textbf{trap } \{(T \textbf{ is } B_T)|(T \in \mathcal{T})\} \textbf{ in } B \textbf{ endtrap} \xrightarrow{\mathbf{i}} B_{T'}} \quad [T' \in \mathcal{T}]$$

The last rule defines a time step in B. It can occur only if there is no T' available for trapping. For suppose that B individually can execute both **tick** or such a T'. Then the T' has an alternative, and as such transfers control to $B_{T'}$ according to the third rule, upon being executed as an **i**, which is by definition not allowed to have a **tick** alternative in the composite process. In other words, trapping is urgent even for a non-urgent T', like it is for δ in ET-LOTOS [7], which is known to be a conservative extension of LOTOS.

$$\frac{B \xrightarrow{\mathbf{tick}} B', \forall T' \in \mathcal{T} : B \not\xrightarrow{T'}}{\textbf{trap } \{(T \textbf{ is } B_T)|(T \in \mathcal{T})\} \textbf{ in } B \textbf{ endtrap} \xrightarrow{\mathbf{tick}} \textbf{trap } \{(T \textbf{ is } B_T)|(T \in \mathcal{T})\} \textbf{ in } B' \textbf{ endtrap}}$$

In a LOTOS-style specification, the only form of trapping will be sequential composition. In any "$A; B_2$", control will be transferred to B_2 according to the original E-LOTOS rule, which is also the original LOTOS rule for action prefix, because the only semantics for A remains the original E-LOTOS semantics, i.e. the result of A will be **null**. In any "$B_1; B_2$" representing a "$B_1 \gg B_2$", the δ of B_1 will be recognized as a LOTOS δ and consequently, control transferred to B_2 according to the original LOTOS rule.

For an E-LOTOS-style specification, the third rule for trapping never applies, while the first and the last rule act as the original E-LOTOS rules for the progress of B.

6. Discussion and Conclusions

We have enhanced basic E-LOTOS with non-urgent and potentially decisive δ and with trapping of such δ. That suffices for precise semantic inclusion of basic LOTOS into basic E-LOTOS, the only necessary syntactic enhancement being co-existence of semantically distinct **exit** and **null**. One could, however, decide to take full advantage of the generalized trapping operator, which is now potentially capable to properly trap any visible action, even an interaction G of multiple processes.

With proper inclusion of basic LOTOS as our exclusive goal, one of our aims has been to let a "$B_1; B_2$" representing a "$B_1 \gg B_2$" transfer control from B_1 to B_2 only upon an explicit **i**. However, in a timed setting, such an **i** seems a bad idea after all. For take for example a "$(B_1 [] \textbf{exit}); B_2$". One would expect that the **exit** stands for skipping, i.e. that the intended behaviour is that of "$(B_1; B_2) [] B_2$". If both alternatives in such a choice can idle, the decision may be delayed. But "$(B_1 [] \textbf{exit}); B_2$" is actually equivalent to "$(B_1; B_2) [] (\mathbf{i}; B_2)$", where the **i** makes the choice unexpectedly urgent.

[3] indicates that it would be possible to define sequential transfer of control, even transfer into multiple alternative directions (as in the case of an operator trapping more than one action), without representing transfer by **i** and without sacrificing time determinacy. So instead of seeking precise compatibility with LOTOS, we should perhaps de-

cide to redesign E-LOTOS more thoroughly, with LOTOS NT [3,8] as a model. In particular, with action urgency no longer being a prerequisite for their trappability, we could reconsider the possibility of entirely dropping urgency of visible actions, particularly the concept of signals (as suggested in [3]), thereby automatically restoring in E-LOTOS the key LOTOS concept of action observability [6].

For conclusion, let us once more emphasize that the proposed enhancement of E-LOTOS is simple, but crucial for semantic backward compatibility with LOTOS. Better compatibility would speed up migration of LOTOS users to E-LOTOS.

References

[1] T. Bolognesi and E. Brinksma, Introduction to the ISO specification language LOTOS, *Computer Networks and ISDN Systems* **14** (1987) 25–59.

[2] H. Garavel and M. Sighireanu, On the introduction of exceptions in LOTOS, in: R. Gotzhein and J. Bredereke (eds.), *Formal Description Techniques IX* (Chapman & Hall, 1996) 469–484.

[3] H. Garavel and M. Sighireanu, Towards a second generation of formal description techniques - rationale for the design of E-LOTOS, in: J.-F. Groote, B. Luttik, and J. van Wamel (eds.), *Proc. of the 3rd International Workshop on Formal Methods for Industrial Critical Systems FMICS'98*, Amsterdam, The Netherlands, May 1998, 187–230.

[4] ISO, *LOTOS – A Formal Description Technique Based on the Temporal Ordering of Observational Behaviour*, ISO 8807, ISO – Information Processing Systems – Open Systems Interconnection, 1989.

[5] ISO/IEC, *Enhancements to LOTOS (E-LOTOS)*, ISO/IEC 15437, ISO – Information Technology, 2001.

[6] M. Kapus-Kolar, Restoring the concept of observability in E-LOTOS, *Computer Standards & Interfaces* **22** (2000) 55–66.

[7] L. Léonard and G. Leduc, An introduction to ET-LOTOS for the description of time-sensitive systems, *Computer Networks and ISDN Systems* **29** (1997) 271–292.

[8] M. Sighireanu (revision by Frédéric Lang), *LOTOS NT User's Manual* (Version 2.1) Technical Report INRIA Rhône-Alpes/VASY, November 2000.

[9] A. Verdejo, *E-LOTOS: Tutorial and Semantics*, M.S. thesis, Universidad Complutense de Madrid, 1999.

Monika Kapus-Kolar received the B.S. degree in electrical engineering from the University of Maribor, Slovenia, in 1981, and the M.S. and Ph.D. degrees in computer science from the University of Ljubljana, Slovenia, in 1984 and 1989, respectively. Since 1981 she has been with the Jožef Stefan Institute, Ljubljana, where she is currently a senior researcher at the Department of Digital Communications and Networks. Her current research interests include formal specification techniques and methods for development of distributed systems and computer networks.