A Correction to "Compositional Service-Based Construction of Multi-Party Time-Sharing-Based Protocols"

Jožef Stefan Institute Technical Report #8896

Monika Kapus-Kolar

October 21, 2003

Abstract

We describe and correct a subtle error in our protocol synthesis algorithm published in *IEICE Trans. on Fundamentals, vol.E86-A, no. 9, pp. 2405–2412.*

Keywords: distributed service implementation, protocol synthesis, LOTOS/T+.

1 Introduction

In [1], we have proposed a method for deriving protocol specifications from service specifications written in LOTOS/T+ [2]. The method is based on the following assumptions:

1. When a place p executes a service primitive (SP) s for which reporting to another place p' is specified, it transmits the required report on s with no delay.
2. The worst-case transit delay for the protocol message is a finite and globally known $d^{p,p'}$.
3. When the message reaches p', it is with no delay accepted by $PE_{p'}$, the protocol entity of the place.

However, we have discovered that the third assumption is not always true in the derived protocols. Fortunately, the error can be easily corrected.

2 An Example of an Erroneous Protocol

Take the service "$a; S$", where S is "$(((b; \mathbf{exit})|||\mathbf{stop})[> (c; \mathbf{exit}))$". In the service, SP a is followed by SP b potentially disrupted by SP c. Let a, b and c belong to different places p, p' and p'', respectively, where $d^{p'',p'}$ is 2.

According to [1], a possible solution for the distributed conflict between b and c is to allow execution of b at times 0,4,8..., and execution of c at times 1,5,9.... With these restrictions, a prompt report on c reaches p' before b is enabled again. a must also be reported, both to p' and p'', for it guards execution of b and c.

Suppose that $PE_{p''}$ receives a report on a at time 3, and afterwards executes c at time 5. On the other hand, suppose that $PE_{p'}$ receives a report on a much later, at time 8. According to [1], this is also the moment when $PE_{p'}$ starts executing the protocol for S. Hence at time 8, $PE_{p'}$ suddenly becomes ready both for b and for a report on c. A report on c is already in the input buffer, having reached p' at time 7 or earlier. However, that does not prevent $PE_{p'}$ from executing b instead, but b after c is illegal.

3 A Solution to the Problem

The problem lies in how [1] encodes the intended behaviour of protocol entities. Take $\mathbf{M}(S_1)$ and $\mathbf{M}(S_2)$, the protocols derived for a service S_1 and a subsequent service S_2. For a PE_p with duties in $\mathbf{M}(S_1)$, [1] defines that $\mathbf{M}_p(S_1 \gg S_2)$, its behaviour within the protocol $\mathbf{M}(S_1 \gg S_2)$, is "$\mathbf{M}_p(S_1) \gg \mathbf{M}_p(S_2)$". Hence $\mathbf{M}_p(S_1)$ and $\mathbf{M}_p(S_2)$ are in strong sequential composition, while protocol message receptions in $\mathbf{M}_p(S_2)$ should actually commute with actions in $\mathbf{M}_p(S_1)$.

A starting action of $\mathbf{M}_p(S_2)$ might be an SP or a protocol message reception. Let $SS_p(S_2)$ be the set of those starting actions which are SPs.

If $SS_p(S_2)$ is empty, "$\mathbf{M}_p(S_1)|||\mathbf{M}_p(S_2)$", i.e. parallel composition of $\mathbf{M}_p(S_1)$ and $\mathbf{M}_p(S_2)$, is an adequate $\mathbf{M}_p(S_1 \gg S_2)$, because a premature start of $\mathbf{M}_p(S_2)$ is prevented already by other places.

If $SS_p(S_2)$ is not empty, an adequate $\mathbf{M}_p(S_1 \gg S_2)$ is
"$(\mathbf{M}_p(S_1) \gg ((|||_{\forall s \in SS_p(S_2)}(s;\mathbf{stop}))[>\mathbf{exit}))|[SS_p(S_2)]|\mathbf{M}_p(S_2)$",
where the operator "$|[SS_p(S_2)]|$" defines that the two parallel processes may execute actions in $SS_p(S_2)$ and successful termination "**exit**" only in co-operation. Hence as in the case of "$\mathbf{M}_p(S_1) \gg \mathbf{M}_p(S_2)$", PE_p takes care that actions in $SS_p(S_2)$ never occur before successful termination of $\mathbf{M}_p(S_1)$.

As a conclusion, we observe that $\mathbf{M}_p(S_1 \gg S_2)$ is the only case where [1] introduces guarding of protocol message receptions. Hence with the describe correction, all message are indeed accepted with no delay.

References

[1] M. Kapus-Kolar, "Compositional service-based construction of multi-party time-sharing-based protocols," IEICE Trans. on Fundamentals, vol.E86-A, no. 9, pp. 2405–2412, 2003.

[2] A. Nakata, T. Higashino, and K. Taniguchi, "Protocol synthesis from timed and structured specifications," in Proc. of ICNP'95, pp. 74–81, IEEE Computer Society Press, 1995.