A Refinement Relation Supporting the Transition from Unbounded to Bounded Communication Buffers

Ketil Stølen

Fakultät für Informatik, TU München
Arcisstrasse 21, D-80290 München

Abstract. This paper proposes a refinement relation supporting the transition from unbounded to bounded communication buffers. Employing this refinement relation, a system specification based on purely asynchronous communication can for example be refined into a system specification where the components communicate purely in terms of handshakes. First a weak version called partial refinement is introduced. Partial refinement guarantees only the preservation of safety properties — preservation in the sense that any implementation of the more concrete specification can be understood as an implementation of the more abstract specification if the latter is a safety property. This refinement relation is then strengthened into total refinement which preserves both safety and liveness properties. Thus a total refinement is also a partial refinement. The suitability of this refinement relation for top-down design is discussed and some examples are given.

1 Introduction

During the final phases of a system development many implementation dependent constraints have to be taken into consideration. This is not a problem as long as the introduction of these constraints is supported by the refinement relation being used — supported in the sense that the specifications in which these constraints have been embedded can be understood as refinements of the earlier more abstract system specifications where these implementation dependent constraints did not occur. Unfortunately this is not always the case.

One important class of such implementation dependent constraints, which (in general) is not supported by standard refinement relations like behavioral refinement and interface refinement, is the class of requirements imposing upper-bounds on the memory available for a communication channel. Such a requirement may for example characterize the maximum number of messages which at one point can be stored in a certain channel without risking malfunction because of channel overflow. Clearly this number may vary from one channel to another depending on the type of messages that are sent along the channel, and the way the channel is implemented.

Of course one way to treat such channel constraints is to introduce them already at the most abstract level. However, this solution is not very satisfactory because these rather trivial constraints may considerably complicate the
specifications and the whole refinement process. The other alternative is to introduce them first in the final phases of a development. However, as already pointed out, this requires a refinement relation supporting the introduction of such constraints.

Consider a network consisting of two specifications $S_1$ and $S_2$ communicating purely asynchronously via an internal channel $y$, as indicated by Network 1 of Fig. 1.

![Network 1](image1)

![Network 2](image2)

**Fig. 1. Introducing Synchronization**

We want to refine Network 1 into a network of two specifications $\tilde{S}_1$ and $\tilde{S}_2$ communicating in a synchronous manner — in other words into a network of the same form as Network 2 of Fig. 1.

That Network 2 is a refinement of Network 1 in the sense that any external behavior of Network 2 is also a behavior of Network 1 is only a necessary requirement, because we may still instantiate $\tilde{S}_1$ and $\tilde{S}_2$ in such a way that the communication via $w$ is completely independent of the communication along $z$. Thus that Network 2 is a refinement of Network 1 does not necessarily mean that we have managed to synchronize the communication. It is still up to the developer to formulate $\tilde{S}_1$ and $\tilde{S}_2$ in such a way that they communicate in accordance with the synchronization protocol the developer prefers.

Nevertheless what is needed is a refinement relation supporting this way of introducing feedback loops. Clearly this refinement relation must allow for the formulation of rules which do not require the proof efforts already invested at the earlier abstraction levels to be repeated. For example, if it has already been proved that Network 1 has the desired overall effect, then it should not be necessary to repeat this proof when Network 1 is refined into Network 2. The formulation of such a refinement relation is the objective of this paper.

The close relationship between specification formalisms based on hand-shake communication and purely asynchronous communication is well-documented in the literature. For example [HJH90] shows how the process algebra of CSP can be extended to handle asynchronous communication by representing each asynchronous communication channel as a separate process. A similar technique