

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

Ketil Stølen
Fakultät für Informatik
Technische Universität München
Postfach 20 24 20, 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 hand-shakes. First a weak version called partial refinement is introduced. Partial refinement guarantees only the preservation of safety properties. 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.

*This work is supported by the Sonderforschungsbereich 342 “Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen”.

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 Figure 1.

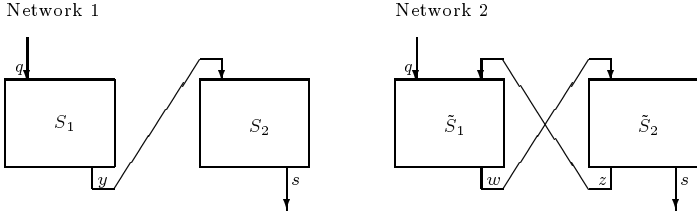


Figure 1: Introducing Synchronization

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 Figure 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 allows different types of synchronous communication to be introduced in an asynchronous system specification: each asynchronous channel is refined into a network of two components which internally communicate in a synchronous manner, and which externally behave like the identity component.

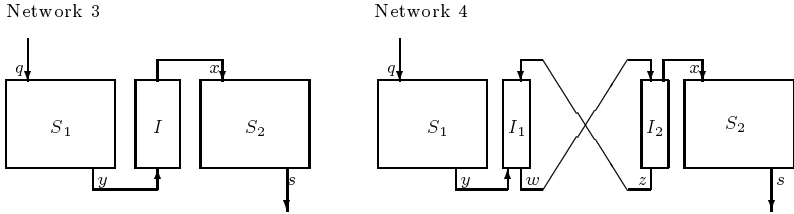


Figure 2: Naive Transformation

In fact with respect to the two networks of Figure 1, using this strategy, we may move from

Network 1 to Network 2 in three steps, employing the usual relation of behavioral refinement, which basically says that a specification S' is a refinement of a specification S iff any behavior of S' is also a behavior of S :

- Step 1: Insert an identity specification I between S_1 and S_2 of Network 1, as indicated by Network 3 of Figure 2. The soundness of this refinement step is obvious.
- Step 2: Refine the identity specification into two subspecifications I_1 and I_2 which communicate in accordance with the desired protocol. We then get Network 4 of Figure 2.
- Step 3: Refine the network consisting of S_1 and I_1 into \tilde{S}_1 and the network consisting of S_2 and I_2 into \tilde{S}_2 , in which case we get Network 2 of Figure 1.

Unfortunately, this strategy is rather tedious, and more importantly: it can only be employed to internal channels. To handle external channels accordingly, a more general refinement relation than behavioral refinement is needed — namely a refinement concept which allows the more concrete specifications to have additional input and output channels.

One might expect that some sort of interface refinement would be sufficient. However, the principles of interface refinement known to us either are not sufficiently general or do not have the desired compositionality properties. The principle of (interaction) interface refinement proposed in [Bro93] allows a channel to be refined into a pair of channels, but only as long as the channels are all of the same direction. Thus the refinement of a channel into two channels of opposite directions is not supported. On the other hand, refinement principles in the tradition of [Hoa72], [Jon87], [AL88], where the concrete state is related to the abstract state via a refinement function, seems to allow for this type of refinement only if the external interface is kept constant. (In our context the state can be understood as a mapping from channel identifiers to their communication histories.)

Below we attempt to deal with this problem by introducing two generalizations of behavioral refinement — one for partial correctness, and one for total correctness — referred to as partial and total refinement, respectively. Partial refinement is sufficient when only safety properties are considered. Total refinement preserves both safety and liveness properties. Thus a total refinement is also a partial refinement.

Total refinement allows for the introduction of both acknowledgement based and demand driven synchronization. However, it is mainly suited for synchronization protocols which do not distinguish between different types of acknowledgements or demands. For example, in certain demand driven algorithms like [AvT87] it is important that the demands sent along a channel are fairly distributed over a set of demand types. Such synchronization protocols are not supported.

The investigations are conducted in the context of dataflow networks modelled by sets of continuous functions. The proposed relation can easily be restated in the context of other models for reactive systems.

The paper is organized as follows: Section 2 introduces the basic concepts; what we mean by specification and refinement is formalized in Section 3; then partial and total refinement are the subjects of Sections 4 and 5; finally, Section 6 contains a summary and discusses a possible generalization.

2 Basic Notations

\mathbb{N} denotes the set of positive natural numbers. A stream is a finite or infinite sequence of actions. It models the communication history of a directed channel. Each action represents one message

sent along the channel. Throughout the paper D denotes the set of all streams. We do not distinguish between different types of streams (streams of naturals etc.). However all our results can easily be generalized to such a setting.

Let d be an action, r and s be streams, and j be a natural number, then:

- ϵ denotes the empty stream;
- $\text{ft}(r)$ denotes the first element of r if r is not empty;
- $\#r$ denotes the length of r ;
- $r|_j$ denotes the prefix of r of length j if $j < \#r$, and r otherwise;
- $d\&s$ denotes the result of appending d to s ;
- $r \frown s$ denotes r if r is infinite and the result of concatenating r with s , otherwise;
- $r \sqsubseteq s$ holds if r is a prefix of s .

A named stream-tuple is a mapping from a finite set of identifiers to the set of streams. It can be thought of as an assignment of channel histories to channel identifiers. Given a set of identifiers I , then I^ω denotes the set of all named stream tuples of signature $I \rightarrow D$. Moreover, $I \mapsto \epsilon$ denotes the element of I^ω which for each identifier in I returns the empty stream; I^∞ denotes the subset of I^ω which maps every identifier to an infinite stream; I^* denotes the subset of I^ω which maps every identifier to a finite stream.

The prefix ordering \sqsubseteq is also used to order named stream-tuples. Given two named stream-tuples $\alpha \in I^\omega$ and $\beta \in O^\omega$, then $\alpha \sqsubseteq \beta$ iff $I = O$ and for all $i \in I$: $\alpha(i) \sqsubseteq \beta(i)$. We also overload the concatenation and length operators: $\alpha \frown \beta$ denotes the named stream tuple in $(I \cup O)^\omega$ such that:

$$\begin{aligned} i \in I \setminus O &\Rightarrow (\alpha \frown \beta)(i) = \alpha(i), \\ i \in O \setminus I &\Rightarrow (\alpha \frown \beta)(i) = \beta(i), \\ i \in I \cap O &\Rightarrow (\alpha \frown \beta)(i) = \alpha(i) \frown \beta(i). \end{aligned}$$

$\#\alpha$ denotes $\min\{\#\alpha(i) \mid i \in I\}$. Finally, α/O denotes the projection of α on O , namely the named stream tuple $\alpha' \in (I \cap O)^\omega$ such that for all $i \in I \cap O$, $\alpha'(i) = \alpha(i)$.

By a chain of named stream tuples we mean an infinite sequence of named stream tuples ordered by \sqsubseteq . Since streams may be infinite any such chain δ has a least upper bound denoted by $\sqcup\delta$.

When convenient named stream tuples are represented as sets of maplets. For example, the set

$$\{a \mapsto \hat{a}, b \mapsto \hat{b}\}$$

denotes the named stream tuple $\alpha \in \{a, b\}^\omega$, where $\alpha(a) = \hat{a}$ and $\alpha(b) = \hat{b}$.

Following [BD92] components are modelled by sets of functions mapping named stream tuples onto named stream tuples. Each such function

$$f : I^\omega \rightarrow O^\omega$$

is required to be monotonic:

for all named stream tuples $\alpha, \beta : \alpha \sqsubseteq \beta \Rightarrow f(\alpha) \sqsubseteq f(\beta)$,

and continuous:

for all chains δ of named stream tuples : $f(\sqcup \delta) = \sqcup \{f(\delta_j) \mid j \in \mathbb{N}\}$.

In the sequel we refer to such functions as stream processing functions.

To reduce the use of the projection operator and thereby simplify the presentation, each function $f \in I^\omega \rightarrow O^\omega$ is overloaded to any domain $Q^\omega \rightarrow O^\omega$ where $I \subseteq Q$, by requiring that for any $\alpha \in Q^\omega$, $f(\alpha) \stackrel{\text{def}}{=} f(\alpha/I)$.

Given two stream processing functions

$$f \in I^\omega \rightarrow O^\omega, \quad \tilde{f} \in \tilde{I}^\omega \rightarrow \tilde{O}^\omega,$$

where $I \cap \tilde{I} = O \cap \tilde{O} = \emptyset$, then $f \parallel \tilde{f}$ is a function of signature

$$I \cup \tilde{I} \rightarrow O \cup \tilde{O},$$

such that $f \parallel \tilde{f}(\alpha) = f(\alpha) \frown \tilde{f}(\alpha)$. If in addition $I \cap O = \tilde{I} \cap \tilde{O} = \emptyset$, then $f \otimes \tilde{f}$ and $f \hat{\otimes} \tilde{f}$ are functions of signatures

$$(I \setminus \tilde{O}) \cup (\tilde{I} \setminus O) \rightarrow (O \setminus \tilde{I}) \cup (\tilde{O} \setminus I), \quad (I \setminus \tilde{O}) \cup (\tilde{I} \setminus O) \rightarrow O \cup \tilde{O},$$

respectively, such that

$$f \otimes \tilde{f}(\alpha) = \beta / (O \setminus \tilde{I} \cup \tilde{O} \setminus I), \quad f \hat{\otimes} \tilde{f}(\alpha) = \beta,$$

given that $f \parallel \tilde{f}(\alpha \frown \beta) = \beta$ is the least fixpoint solution with respect to \sqsubseteq . In Figure 3, Network 1 represents composition by \otimes and Network 2 represents composition by $\hat{\otimes}$. Thus \otimes differs from $\hat{\otimes}$ in that it hides the feedback channels.

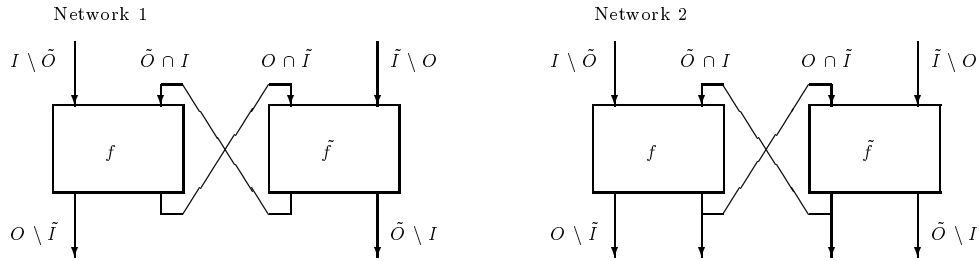


Figure 3: Networks Relating the Operators \otimes and $\hat{\otimes}$

Given $n > 1$ stream processing functions

$$f_j \in I_j^\omega \rightarrow O_j^\omega \quad 1 \leq j \leq n,$$

such that $I_j \cap O_j = \emptyset$ and $l \neq k$ imply $I_l \cap I_k = O_l \cap O_k = \emptyset$, then $\otimes_{j=1}^n f_j$ is a short-hand for $f_1 \otimes \dots \otimes f_n$. Note that the restrictions imposed on the identifier sets imply that \otimes is associative — thus the bracketing is unimportant. $\hat{\otimes}$ and \parallel are generalized accordingly.

3 Specification and Refinement

A specification is represented by a triple

$$(I, O, R),$$

where I and O are disjoint sets of identifiers, and R is a formula with the elements of I and O as its only free variables. We refer to the elements of I and O as the input- and the output-identifiers, respectively. Each input-identifier models the communication history of an input channel and each output-identifier models the communication history of an output channel. R characterizes the allowed relation between the communication histories of the input-channels and the communication histories of the output-channels and is therefore called the input/output relation. Each communication history is modelled by a (possibly infinite) stream of actions.

The denotation of a specification $S \stackrel{\text{def}}{=} (I, O, R)$ is a set of stream processing functions mapping named stream tuples onto named stream tuples, namely the set characterized by:

$$\llbracket S \rrbracket \stackrel{\text{def}}{=} \{f \in I^\omega \rightarrow O^\omega \mid \forall \alpha \in I^\omega : (\alpha, f(\alpha)) \models R\},$$

where $(\alpha, f(\alpha)) \models R$ iff R evaluates to true when each input identifier $i \in I$ is interpreted as $\alpha(i)$, and each output identifier $o \in O$ is interpreted as $f(\alpha)(o)$.

The basic refinement relation is represented by \rightsquigarrow . It holds only for specifications with identical sets of input/output identifiers. Given two specifications S_1 and S_2 , then $S_1 \rightsquigarrow S_2$ iff $\llbracket S_2 \rrbracket \subseteq \llbracket S_1 \rrbracket$. Thus a specification S_2 refines a specification S_1 iff any function which satisfies S_2 also satisfies S_1 . This corresponds to what is normally referred to as behavioral refinement.

Given two specifications $S_1 \stackrel{\text{def}}{=} (I_1, O_1, R_1)$ and $S_2 \stackrel{\text{def}}{=} (I_2, O_2, R_2)$, such that

$$I_1 \cap I_2 = O_1 \cap O_2 = \emptyset,$$

then $S_1 \otimes S_2$ represents the network pictured in Figure 4. The channels modelled by $O_1 \cap I_2$ and $O_2 \cap I_1$ are internal. The external input channels are represented by $(I_1 \setminus O_2) \cup (I_2 \setminus O_1)$, and $(O_1 \setminus I_2) \cup (O_2 \setminus I_1)$ represents the external output channels. The denotation of this network is characterized by

$$\llbracket S_1 \otimes S_2 \rrbracket \stackrel{\text{def}}{=} \{f_1 \otimes f_2 \mid f_1 \in \llbracket S_1 \rrbracket \wedge f_2 \in \llbracket S_2 \rrbracket\}.$$

The operator $\otimes_{j=1}^n$ is lifted from functions to specifications in a similar way.

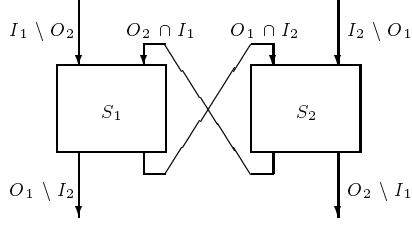


Figure 4: $S_1 \otimes S_2$

4 Partial Refinement

This section introduces a refinement relation, called partial refinement, which guarantees the preservation of safety properties. The suitability of this refinement relation for top-down system development is investigated.

Given two specifications

$$S \stackrel{\text{def}}{=} (Q, O, R), \quad \tilde{S} \stackrel{\text{def}}{=} (\tilde{Q}, \tilde{O}, \tilde{R}),$$

where $Q \subseteq \tilde{Q}$ and $O \subseteq \tilde{O}$, then \tilde{S} is a partial refinement of S , written $S \rightsquigarrow \tilde{S}$, iff

$$\forall \tilde{f} \in \llbracket \tilde{S} \rrbracket \Rightarrow \exists f \in \llbracket S \rrbracket : \forall \alpha \in \tilde{Q}^\omega : \tilde{f}(\alpha)/O \sqsubseteq f(\alpha).$$

Thus \tilde{S} is a partial refinement of S iff for any function \tilde{f} which satisfies \tilde{S} , there is a function f which satisfies S , such that for any input history α for the channels represented by \tilde{Q} , the projection of $\tilde{f}(\alpha)$ on O is a prefix of $f(\alpha)$. Remember that $f(\alpha)$ by definition is equal to $f(\alpha/Q)$.

We now prove that partial refinement is reflexive, transitive and a congruence with respect to \otimes . This implies that whenever we have refined a specification S into a network of specifications $\otimes_{j=1}^n S_j$ such that

$$S \rightsquigarrow \otimes_{j=1}^n S_j, \quad (*)$$

and there is a network of specifications $\otimes_{j=1}^n S'_j$ such that

$$S_j \rightsquigarrow S'_j \quad 1 \leq j \leq n,$$

then it also holds that

$$S \rightsquigarrow \otimes_{j=1}^n S'_j.$$

Thus the workload invested in establishing $(*)$ does not have to be repeated when the refinement of the component specifications of $\otimes_{j=1}^n S_j$ is continued.

We start by proving that \rightsquigarrow is reflexive and transitive.

Proposition 1 *Given three specifications S_1, S_2, S_3 whose input/output identifiers are characterized by $(Q_1, O_1), (Q_2, O_2), (Q_3, O_3)$, respectively. Assume that $Q_1 \subseteq Q_2 \subseteq Q_3$ and $O_1 \subseteq O_2 \subseteq O_3$. Then*

- (1) : $S_1 \overset{p}{\rightsquigarrow} S_1$,
- (2) : $S_1 \overset{p}{\rightsquigarrow} S_2 \wedge S_2 \overset{p}{\rightsquigarrow} S_3 \Rightarrow S_1 \overset{p}{\rightsquigarrow} S_3$.

Proof: (1) follows trivially. To prove (2), assume

- (3) : $S_1 \overset{p}{\rightsquigarrow} S_2$,
- (4) : $S_2 \overset{p}{\rightsquigarrow} S_3$.

Let f_3 be such that

- (5) : $f_3 \in \llbracket S_3 \rrbracket$.

(4), (5) imply there is an f_2 such that

- (6) : $f_2 \in \llbracket S_2 \rrbracket$,
- (7) : $f_3(\alpha)/O_2 \sqsubseteq f_2(\alpha)$.

(3), (6) imply there is an f_1 such that

- (8) : $f_1 \in \llbracket S_1 \rrbracket$,
- (9) : $f_2(\alpha)/O_1 \sqsubseteq f_1(\alpha)$.

(7), (9) and $O_1 \subseteq O_2$ imply

- (10) : $f_3(\alpha)/O_1 = f_3(\alpha)/O_2/O_1 \sqsubseteq f_2(\alpha)/O_1 \sqsubseteq f_1(\alpha)$.

The way (10) was deduced from (3), (4) implies (2).

end of proof

Before stating the general congruence property for partial refinement, we prove an intermediate result, whose conclusion (3) is visualized by Figure 5. Thus we have four specifications $S_1, S_2, \tilde{S}_1, \tilde{S}_2$. Their sets of input/output identifiers are characterized by $(Q \cup X, O \cup Y)$, $(Y \cup Z, X \cup K)$, $(\tilde{Q} \cup \tilde{X}, \tilde{O} \cup \tilde{Y})$, $(\tilde{Y} \cup \tilde{Z}, \tilde{X} \cup \tilde{K})$, respectively. It is assumed that the sets of identifiers $\tilde{Q}, \tilde{X}, \tilde{O}, \tilde{Y}, \tilde{Z}, \tilde{K}$ are all disjoint, and that $Q \subseteq \tilde{Q}, X \subseteq \tilde{X}, O \subseteq \tilde{O}, Y \subseteq \tilde{Y}, Z \subseteq \tilde{Z}, K \subseteq \tilde{K}$.

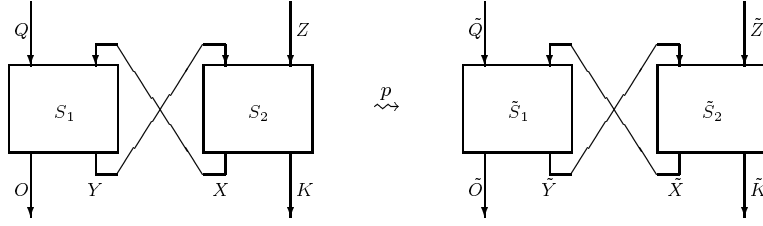


Figure 5: Partial Refinement

Proposition 2 *If*

- (1) : $S_1 \xrightarrow{p} \tilde{S}_1$,
- (2) : $S_2 \xrightarrow{p} \tilde{S}_2$

then

- (3) : $S_1 \otimes S_2 \xrightarrow{p} \tilde{S}_1 \otimes \tilde{S}_2$.

Proof: Let \tilde{f}_1 and \tilde{f}_2 be such that

- (4) : $\tilde{f}_1 \in \llbracket \tilde{S}_1 \rrbracket$,
- (5) : $\tilde{f}_2 \in \llbracket \tilde{S}_2 \rrbracket$.

(1), (2), (4), (5) imply there are f_1 and f_2 such that

- (6) : $f_1 \in \llbracket S_1 \rrbracket$,
- (7) : $f_2 \in \llbracket S_2 \rrbracket$,
- (8) : $\tilde{f}_1(\alpha)/(O \cup Y) \sqsubseteq f_1(\alpha)$,
- (9) : $\tilde{f}_2(\alpha)/(X \cup K) \sqsubseteq f_2(\alpha)$.

(3) follows if it can be shown that

- (10) : $\tilde{f}_1 \otimes \tilde{f}_2(\alpha)/(O \cup K) \sqsubseteq f_1 \otimes f_2(\alpha)$.

Given some $\alpha \in (\tilde{Q} \cup \tilde{Z})^\omega$ and assume that

- (11) : $f_1 \hat{\otimes} f_2(\alpha) = \beta$.

The monotonicity of \tilde{f}_1 and \tilde{f}_2 implies there are chains $\tilde{\alpha}, \tilde{\beta}$ such that

$$(12) : \tilde{\alpha}_1 = \alpha \frown (\tilde{X} \cup \tilde{Y} \mapsto \epsilon),$$

$$(13) : \tilde{\beta}_j = \tilde{f}_1 \parallel \tilde{f}_2(\tilde{\alpha}_j),$$

$$(14) : \tilde{\alpha}_{j+1} = \alpha \frown \tilde{\beta}_j.$$

(Remember that any stream processing function $f \in I^\omega \rightarrow O^\omega$ is overloaded to any domain $Q^\omega \rightarrow O^\omega$ where $I \subseteq Q$.)

We want to prove that

$$(15) : \tilde{\beta}_j / (O \cup Y \cup X \cup K) \sqsubseteq \beta.$$

The base-case follows trivially from (8), (9), (11), (12), (13) and the monotonicity of f_1 and f_2 . Assume for some $k \geq 1$

$$(16) : \tilde{\beta}_k / (O \cup Y \cup X \cup K) \sqsubseteq \beta.$$

We show that

$$(17) : \tilde{\beta}_{k+1} / (O \cup Y \cup X \cup K) \sqsubseteq \beta.$$

(13) implies that

$$(18) : \tilde{\beta}_{k+1} / (O \cup Y \cup X \cup K) = \tilde{f}_1 \parallel \tilde{f}_2(\tilde{\alpha}_{k+1}) / (O \cup Y \cup X \cup K).$$

(18) and the definition of \parallel imply that

$$(19) : \tilde{\beta}_{k+1} / (O \cup Y \cup X \cup K) = \tilde{f}_1(\tilde{\alpha}_{k+1}) / (O \cup Y) \frown \tilde{f}_2(\tilde{\alpha}_{k+1}) / (X \cup K).$$

(8), (9), (19) imply

$$(20) : \tilde{\beta}_{k+1} / (O \cup Y \cup X \cup K) \sqsubseteq f_1(\tilde{\alpha}_{k+1}) \frown f_2(\tilde{\alpha}_{k+1}).$$

(14), (20) imply

$$(21) : \tilde{\beta}_{k+1} / (O \cup Y \cup X \cup K) \sqsubseteq f_1(\alpha \frown \tilde{\beta}_k) \frown f_2(\alpha \frown \tilde{\beta}_k).$$

(16), (21) and the monotonicity of f_1 and f_2 imply

$$(22) : \tilde{\beta}_{k+1} / (O \cup Y \cup X \cup K) \sqsubseteq f_1(\alpha \frown \beta) \frown f_2(\alpha \frown \beta).$$

(11), (22) imply (17). This ends the proof of (15).

(15) and the continuity of \sqcup imply

$$(23) : \sqcup \tilde{\beta} / (O \cup Y \cup X \cup K) \sqsubseteq \beta.$$

(12), (13), (14) imply

$$(24) : \tilde{f}_1 \hat{\otimes} \tilde{f}_2(\alpha) = \sqcup \tilde{\beta}.$$

(11), (23), (24) and the fact that \otimes is equal to $\hat{\otimes}$ plus hiding imply (10).

end of proof

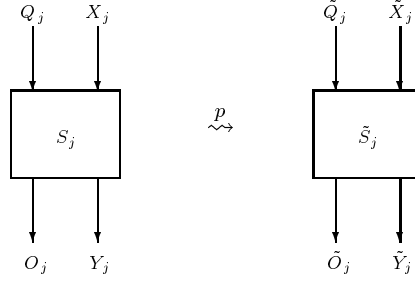


Figure 6: Partial Refinement of the j 'th Component Specification

We now extend Proposition 2 to finite networks of n specifications. Each of the n component specifications S_j is partially refined into a component specification \tilde{S}_j in accordance with Figure 6. \tilde{Q}_j represents the external input channels, \tilde{X}_j represents the internal input channels, \tilde{O}_j represents the external output channels, and \tilde{Y}_j represents the internal output channels. This means that $\cup_{j=1}^n X_j = \cup_{j=1}^n Y_j$. It is assumed that the $3 \times n$ sets $\tilde{Q}_j, \tilde{X}_j, \tilde{O}_j$ are all disjoint, that the n sets \tilde{Y}_j are all disjoint and that $Q_j \subseteq \tilde{Q}_j, X_j \subseteq \tilde{X}_j$ etc. .

Proposition 3 *If*

$$(1) : S_j \overset{p}{\rightsquigarrow} \tilde{S}_j \quad 1 \leq j \leq n,$$

then

$$(2) : \otimes_{j=1}^n S_j \overset{p}{\rightsquigarrow} \otimes_{j=1}^n \tilde{S}_j.$$

Proof: Follows from Proposition 2 by induction on n .

end of proof

5 Total Refinement

In the previous section a refinement relation called partial refinement was introduced. It was shown that this relation is reflexive, transitive and a congruence with respect to \otimes . Thus partial

refinement is well-suited as a principle for top-down design. Unfortunately, partial refinement only preserves safety properties. To ensure the preservation of both safety and liveness properties a stronger refinement relation is needed — namely what we refer to as total refinement.

Given two specifications

$$S \stackrel{\text{def}}{=} (Q, O, R), \quad \tilde{S} \stackrel{\text{def}}{=} (\tilde{Q}, \tilde{O}, \tilde{R}),$$

where $Q \subseteq \tilde{Q}$ and $O \subseteq \tilde{O}$, then \tilde{S} is a total refinement of S , written $S \overset{t}{\rightsquigarrow} \tilde{S}$, iff

$$\forall \tilde{f} \in \llbracket \tilde{S} \rrbracket \Rightarrow \exists f \in \llbracket S \rrbracket : \forall \alpha \in \tilde{Q}^\omega : \#\alpha / (\tilde{Q} \setminus Q) = \infty \Rightarrow \tilde{f}(\alpha) / O = f(\alpha).$$

Thus \tilde{S} is a total refinement of S iff for any function \tilde{f} which satisfies \tilde{S} , there is a function f which satisfies S , such that for any input history α , whose projection on $\tilde{Q} \setminus Q$ is infinite, the projection of $\tilde{f}(\alpha)$ on O is equal to $f(\alpha)$.

The antecedent “projection on $\tilde{Q} \setminus Q$ is infinite” may seem too strong. However, since we in this paper restrict ourselves to synchronization protocols whose behavior depend only upon whether an acknowledgement (demand) is received or not, and not upon what sort of acknowledgement (demand) is received, this is exactly what is needed.

Assume $Q = \tilde{Q}$ and $O = \tilde{O}$, then $S \overset{t}{\rightsquigarrow} \tilde{S}$, iff

$$\forall \tilde{f} \in \llbracket \tilde{S} \rrbracket \Rightarrow \exists f \in \llbracket S \rrbracket : \forall \alpha \in \tilde{Q}^\omega : \tilde{f}(\alpha) = f(\alpha),$$

which is equivalent to $\llbracket \tilde{S} \rrbracket \subseteq \llbracket S \rrbracket$. Thus in this case total refinement degenerates to behavioral refinement, which implies that total refinement can be seen as a generalization of behavioral refinement.

We start by proving that $\overset{t}{\rightsquigarrow}$ is reflexive and transitive.

Proposition 4 *Given three specifications S_1, S_2, S_3 whose input/output identifiers are characterized by $(Q_1, O_1), (Q_2, O_2), (Q_3, O_3)$, respectively. Assume that $Q_1 \subseteq Q_2 \subseteq Q_3$ and $O_1 \subseteq O_2 \subseteq O_3$. Then*

- (1) : $S_1 \overset{t}{\rightsquigarrow} S_1$,
- (2) : $S_1 \overset{t}{\rightsquigarrow} S_2 \wedge S_2 \overset{t}{\rightsquigarrow} S_3 \Rightarrow S_1 \overset{t}{\rightsquigarrow} S_3$.

Proof: (1) follows trivially. To prove (2), assume

- (3) : $S_1 \overset{t}{\rightsquigarrow} S_2$,
- (4) : $S_2 \overset{t}{\rightsquigarrow} S_3$.

Let f_3 be such that

- (5) : $f_3 \in \llbracket S_3 \rrbracket$.

(4), (5) imply there is an f_2 such that

$$(6) : f_2 \in \llbracket S_2 \rrbracket,$$

$$(7) : \#\alpha/(Q_3 \setminus Q_2) = \infty \Rightarrow f_3(\alpha)/O_2 = f_2(\alpha).$$

(3), (6) imply there is an f_1 such that

$$(8) : f_1 \in \llbracket S_1 \rrbracket,$$

$$(9) : \#\alpha/(Q_2 \setminus Q_1) = \infty \Rightarrow f_2(\alpha)/O_1 = f_1(\alpha).$$

Assume

$$(10) : \#\alpha/(Q_3 \setminus Q_1) = \infty.$$

(10) and $Q_1 \subseteq Q_2$ imply

$$(11) : \#\alpha(Q_3 \setminus Q_2) = \infty.$$

(7), (11) imply

$$(12) : f_3(\alpha)/O_2 = f_2(\alpha) = f_2(\alpha/Q_2).$$

(10) and $Q_2 \subseteq Q_3$ imply

$$(13) : \#(\alpha/Q_2)/(Q_2 \setminus Q_1) = \infty.$$

(9), (13) and $Q_1 \subseteq Q_2$ imply

$$(14) : f_2(\alpha/Q_2)/O_1 = f_1(\alpha/Q_2) = f_1(\alpha).$$

(12) and $O_1 \subseteq O_2$ imply

$$(15) : f_2(\alpha/Q_2)/O_1 = f_3(\alpha)/O_2/O_1 = f_3(\alpha)/O_1.$$

(14), (15) imply

$$(16) : f_3(\alpha)/O_1 = f_1(\alpha).$$

The way (16) was deduced from (10) implies

$$(17) : \#\alpha/(Q_3 \setminus Q_1) = \infty \Rightarrow f_3(\alpha)/O_1 = f_1(\alpha).$$

The way (17) was deduced from (3), (4) implies (2).
end of proof

It has been proved that partial refinement is a congruence with respect to the composition operator \otimes . The same does not hold for total refinement.

Example 1 Congruence Problem:

To see that total refinement does not have this property, let

$$\begin{aligned} S_1 &\stackrel{\text{def}}{=} (\{q\}, \{y\}, y = q), & \tilde{S}_1 &\stackrel{\text{def}}{=} (\{q, x\}, \{y\}, y \sqsubseteq q \wedge \#y = \min\{\#x + 1, \#q\}), \\ S_2 &\stackrel{\text{def}}{=} (\{y\}, \{k\}, k = y), & \tilde{S}_2 &\stackrel{\text{def}}{=} (\{y\}, \{x, k\}, k = y \wedge \#x = \max\{\#y - 1, 0\}). \end{aligned}$$

Clearly $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$ and $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$. Unfortunately, for all $f \in \llbracket S_1 \otimes S_2 \rrbracket$ and $\tilde{f} \in \llbracket \tilde{S}_1 \otimes \tilde{S}_2 \rrbracket$, and any nonempty stream s , it holds that

$$\begin{aligned} f(\{q \mapsto s\}) &= \{k \mapsto s\}, \\ \tilde{f}(\{q \mapsto s\}) &= \{k \mapsto \text{ft}(s) \& \epsilon\}. \end{aligned}$$

Since $s \neq \text{ft}(s) \& \epsilon$ if $\#s > 1$ it follows that

$$S_1 \otimes S_2 \not\overset{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2.$$

□

What is required is some additional proof-obligation characterizing under what conditions total refinement is a “congruence” with respect to \otimes . To allow systems to be developed in a top-down style this proof obligation must be checkable based on the information available at the point in time where the refinement step is carried out — for example this proof-obligation should not require knowledge about how \tilde{S}_1 and \tilde{S}_2 are implemented. With respect to Example 1 the following condition is obviously sufficient:

$$\forall \tilde{f} \in \llbracket \tilde{S}_1 \otimes \tilde{S}_2 \rrbracket \Rightarrow \exists f \in \llbracket S_1 \otimes S_2 \rrbracket : \tilde{f}(\alpha) = f(\alpha) \quad (*).$$

If (*) holds there is no need to require that $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$ and $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$. This fact also characterizes the weakness of (*). If we later decide to compose $\tilde{S}_1 \otimes \tilde{S}_2$ with another network \tilde{S}_3 such that $S_3 \overset{t}{\rightsquigarrow} \tilde{S}_3$, then it is not easy to exploit the fact that we have already proved (*) when we now decide to prove that

$$\otimes_{j=1}^3 S_j \overset{t}{\rightsquigarrow} \otimes_{j=1}^3 \tilde{S}_j.$$

What we want is a proof obligation which takes advantage of the fact that $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$ and $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$ in the sense that the formulation of this additional obligation is independent of S_1 and S_2 .

The problem observed in Example 1 is that total refinement may lead to premature termination when the specifications are composed into networks with feedback loops. This phenomenon can be understood as deadlock caused by an erroneous synchronization protocol.

With respect to the given semantics this problem occurs only when the refinement step introduces a new least fixpoint — new in the sense that the least fixpoint is reached too early. For the refinement step conducted in Example 1, it therefore seems sensible to require that for any $\alpha \in \{q\}^\omega$:

$$\tilde{f}_1 \in \llbracket \tilde{S}_1 \rrbracket \wedge \tilde{f}_2 \in \llbracket \tilde{S}_2 \rrbracket \wedge \tilde{f}_1 \hat{\otimes} \tilde{f}_2(\alpha) = \beta \wedge \alpha' \in \{x\}^\infty \Rightarrow \tilde{f}_1(\alpha \frown \beta \frown \alpha') = \beta/\{y\} \quad (**)$$

This condition states that when the least fixpoint has been reached then the output along y will not be extended if additional input is received along the feedback channel x . It makes sure that no new least fixpoint has been introduced as a result of the synchronization.

In some sense the proof-obligation corresponds to the freedom from deadlock tests in more traditional proof systems [OG76], [Stø91] and [PJ91]. In Example 1 this proof obligation is not fulfilled. However, if \tilde{S}_2 's input/output relation is replaced by

$$k = y \wedge \#x = \#y$$

then $(**)$ holds. Thus in the case of Example 1, $(**)$ seems to be a reasonable proof-obligation. The next step is to figure out how this obligation should look like in the general case.

Example 2 :

Let S_1 , S_2 and \tilde{S}_1 be as in Example 1, and let

$$\tilde{S}_2 \stackrel{\text{def}}{=} (\{y, z\}, \{x, k\}, k \sqsubseteq y \wedge \#k = \#x = \min\{\#y, \#z\}).$$

We then have that

$$S_1 \otimes S_2 \overset{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2.$$

Unfortunately, $(**)$ does not hold. To see that, let $\tilde{f}_1 \in \llbracket \tilde{S}_1 \rrbracket$, $\tilde{f}_2 \in \llbracket \tilde{S}_2 \rrbracket$, and assume that \hat{q} is a stream such that $\#\hat{q} > 1$. Clearly

$$\tilde{f}_1 \hat{\otimes} \tilde{f}_2(\{q \mapsto \hat{q}, z \mapsto \epsilon\}) = \{y \mapsto \text{ft}(\hat{q}) \& \epsilon, x \mapsto \epsilon, k \mapsto \epsilon\}.$$

Moreover

$$\tilde{f}_1(\{q \mapsto \hat{q}, x \mapsto \hat{q}\}) = \{y \mapsto \hat{q}\}.$$

Thus $(**)$ is not satisfied.

□

In fact $(**)$ must be weakened by adding assumptions about the environment's behavior. In the case of Example 2 it seems sensible to require that for any $\alpha \in \{q, z\}^\omega$:

$$\begin{aligned} \tilde{f}_1 \in \llbracket \tilde{S}_1 \rrbracket \wedge \tilde{f}_2 \in \llbracket \tilde{S}_2 \rrbracket \wedge \tilde{f}_1 \hat{\otimes} \tilde{f}_2(\alpha) = \beta \wedge \#(\alpha/\{z\}) = \infty \wedge \alpha' \in \{x\}^\infty \\ \Rightarrow \\ \tilde{f}_1(\alpha \frown \beta \frown \alpha') = \beta/\{y\}. \end{aligned}$$

This motivates the next proposition, which characterizes a condition under which a total refinement corresponding to Figure 5 is valid. It is assumed that \tilde{Q} , \tilde{X} , \tilde{O} , \tilde{Y} , \tilde{Z} , \tilde{K} are disjoint sets of identifiers with corresponding subsets Q , \hat{Q} , X , \hat{X} , etc. such that $\hat{Q} = \tilde{Q} \setminus Q$, $\hat{X} = \tilde{X} \setminus X$, etc.

Proposition 5 *If for any $\alpha \in (\tilde{Q} \cup \tilde{Z})^\omega$, $\beta \in (\tilde{O} \cup \tilde{Y} \cup \tilde{X} \cup \tilde{K})^\omega$, $\alpha' \in (\hat{X} \cup \hat{Y})^\infty$*

$$\begin{aligned}
(1) : S_1 &\overset{t}{\rightsquigarrow} \tilde{S}_1, \\
(2) : S_2 &\overset{t}{\rightsquigarrow} \tilde{S}_2, \\
(3) : \tilde{f}_1 \in \llbracket \tilde{S}_1 \rrbracket \wedge \tilde{f}_2 \in \llbracket \tilde{S}_2 \rrbracket \wedge \tilde{f}_1 \hat{\otimes} \tilde{f}_2(\alpha) = \beta \wedge \#\alpha / (\hat{Q} \cup \hat{Z}) = \infty \\
&\Rightarrow \\
&\tilde{f}_1 \parallel \tilde{f}_2(\alpha \frown \beta \frown \alpha') / (O \cup Y \cup X \cup K) = \beta / (O \cup Y \cup X \cup K)
\end{aligned}$$

then

$$(4) : S_1 \otimes S_2 \overset{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2.$$

Proof: Assume (1), (2), (3). Let \tilde{f}_1 and \tilde{f}_2 be such that

$$\begin{aligned}
(5) : \tilde{f}_1 &\in \llbracket \tilde{S}_1 \rrbracket, \\
(6) : \tilde{f}_2 &\in \llbracket \tilde{S}_2 \rrbracket.
\end{aligned}$$

(1), (2), (5), (6) imply there are f_1 and f_2 such that

$$\begin{aligned}
(7) : f_1 &\in \llbracket S_1 \rrbracket, \\
(8) : f_2 &\in \llbracket S_2 \rrbracket, \\
(9) : \#\alpha / (\hat{Q} \cup \hat{X}) = \infty &\Rightarrow \tilde{f}_1(\alpha) / (O \cup Y) = f_1(\alpha), \\
(10) : \#\alpha / (\hat{Y} \cup \hat{Z}) = \infty &\Rightarrow \tilde{f}_2(\alpha) / (X \cup K) = f_2(\alpha).
\end{aligned}$$

It is enough to show that

$$(11) : \#\alpha / (\hat{Q} \cup \hat{Z}) = \infty \Rightarrow \tilde{f}_1 \otimes \tilde{f}_2(\alpha) / (O \cup K) = f_1 \otimes f_2(\alpha).$$

Given some $\alpha \in (\tilde{Q} \cup \tilde{Z})^\omega$ such that

$$(12) : \#\alpha / (\hat{Q} \cup \hat{Z}) = \infty.$$

Assume that

$$(13) : f_1 \hat{\otimes} f_2(\alpha) = \beta.$$

The monotonicity of \tilde{f}_1 and \tilde{f}_2 implies there are chains such that

$$\begin{aligned}
(14) : \tilde{\alpha}_1 &= \alpha \frown (\tilde{X} \cup \tilde{Y} \mapsto \epsilon), \\
(15) : \tilde{\alpha}_{j+1} &= \alpha \frown \tilde{\beta}_j, \\
(16) : \tilde{\beta}_j &= \tilde{f}_1 \parallel \tilde{f}_2(\tilde{\alpha}_j).
\end{aligned}$$

As in the proof of Proposition 2 it follows straightforwardly by induction on j that

$$(17) : \tilde{\beta}_j / (O \cup Y \cup X \cup K) \sqsubseteq \beta.$$

(17) and the continuity of \sqcup imply

$$(18) : \sqcup \tilde{\beta} / (O \cup Y \cup X \cup K) \sqsubseteq \beta.$$

Since $\tilde{\beta}$ characterizes the Kleene-chain, it also holds that

$$(19) : \tilde{f}_1 \hat{\otimes} \tilde{f}_2(\alpha) = \sqcup \tilde{\beta}.$$

Assume

$$(20) : \alpha' \in (\hat{X} \cup \hat{Y})^\infty.$$

(3), (5), (6), (12), (19), (20) imply

$$(21) : \tilde{f}_1 \parallel \tilde{f}_2(\alpha \frown (\sqcup \tilde{\beta}) \frown \alpha') / (O \cup Y \cup X \cup Z) = \sqcup \tilde{\beta} / (O \cup Y \cup X \cup Z).$$

(9), (10), (12), (20) imply

$$(22) : \tilde{f}_1 \parallel \tilde{f}_2(\alpha \frown (\sqcup \tilde{\beta}) \frown \alpha') / (O \cup Y \cup X \cup Z) = f_1 \parallel f_2(\alpha \frown (\sqcup \tilde{\beta}) \frown \alpha').$$

(20), (21), (22) imply

$$(23) : f_1 \parallel f_2(\alpha \frown (\sqcup \tilde{\beta}) \frown \alpha') = f_1 \parallel f_2(\alpha \frown (\sqcup \tilde{\beta})) = \sqcup \tilde{\beta} / (O \cup Y \cup X \cup Z).$$

(13), (18), (23) imply

$$(24) : f_1 \parallel f_2(\alpha \frown (\sqcup \tilde{\beta})) = f_1 \hat{\otimes} f_2(\alpha).$$

(23), (24) imply

$$(25) : \sqcup \tilde{\beta} / (O \cup Y \cup X \cup Z) = f_1 \hat{\otimes} f_2(\alpha).$$

(19), (25) imply

$$(26) : \tilde{f}_1 \hat{\otimes} \tilde{f}_2(\alpha) / (O \cup Y \cup X \cup K) = f_1 \hat{\otimes} f_2(\alpha).$$

(26) and the fact that \otimes is equal to $\hat{\otimes}$ minus hiding imply (11).

end of proof

It can be argued that the freedom from deadlock test (3) of Proposition 5 is too strong, because we may find specifications S_1 , S_2 , \tilde{S}_1 and \tilde{S}_2 which satisfy (1), (2) and (4), but not (3). For

example this is the case if:

$$\begin{aligned} S_1 &\stackrel{\text{def}}{=} (\{q\}, \{y\}, y = q), & \tilde{S}_1 &\stackrel{\text{def}}{=} (\{q, x\}, \{y\}, y \sqsubseteq q \wedge \#y = \min\{\#q, \#x + 1\}), \\ S_2 &\stackrel{\text{def}}{=} (\{y\}, \{k\}, k = y|_{10}), & \tilde{S}_2 &\stackrel{\text{def}}{=} (\{y\}, \{k, x\}, x = k = y|_{10}). \end{aligned}$$

However, whenever we run into such a problem, which seems to be rather artificial, there are specifications $S'_1, S'_2, \tilde{S}'_1, \tilde{S}'_2$ such that

$$S_1 \otimes S_2 \rightsquigarrow S'_1 \otimes S'_2, \quad \tilde{S}_1 \otimes \tilde{S}_2 \rightsquigarrow \tilde{S}'_1 \otimes \tilde{S}'_2,$$

holds, and

$$S'_1 \otimes S'_2 \overset{t}{\rightsquigarrow} \tilde{S}'_1 \otimes \tilde{S}'_2$$

follows by Proposition 5. For example, with respect to our example, this is the case if

$$\begin{aligned} S'_1 &\stackrel{\text{def}}{=} (\{q\}, \{y\}, y = q|_{10}), & \tilde{S}'_1 &\stackrel{\text{def}}{=} (\{q, x\}, \{y\}, y \sqsubseteq q \wedge \#y = \min\{\#q|_{10}, \#x + 1\}), \\ S'_2 &\stackrel{\text{def}}{=} S_2, & \tilde{S}'_2 &\stackrel{\text{def}}{=} \tilde{S}_2. \end{aligned}$$

Thus it is enough to strengthen the specifications in such a way that the communication along the internal channels is halted as soon as the external channels have reached their final value.

Since \rightsquigarrow is a special case of $\overset{t}{\rightsquigarrow}$ it follows that Proposition 5 is (relative, semantic) complete modulo a (relative, semantic) complete set of rules for behavioral refinement.

Another point to note is that in practise it is normally so that whenever (3) holds we also have that

$$\begin{aligned} \tilde{f}_1 \in \llbracket \tilde{S}_1 \rrbracket \wedge \tilde{f}_2 \in \llbracket \tilde{S}_2 \rrbracket \wedge \tilde{f}_1 \parallel \tilde{f}_2(\alpha \frown \beta) = \beta \wedge \alpha / (\hat{Q} \cup \hat{Z}) = \infty \\ \Rightarrow \\ \tilde{f}_1 \parallel \tilde{f}_2(\alpha \frown \beta \frown \alpha') / (O \cup Y \cup X \cup K) = \beta / (O \cup Y \cup X \cup K). \end{aligned}$$

Thus in order to use Proposition 5 it is in most cases not necessary to characterize the least-fixpoint solution.

We now generalize Proposition 5 in the same way as Proposition 2 was generalized in the previous section. Thus we have a network of n component specifications S_j which are totally refined into component specifications \tilde{S}_j in accordance with Figure 6. As before \tilde{Q}_j represents the external input channels, \tilde{X}_j represents the internal input channels, \tilde{O}_j represents the external output channels, and \tilde{Y}_j represents the internal output channels. Moreover, we also have the same constraints as earlier, namely that $\cup_{j=1}^n \tilde{X}_j = \cup_{j=1}^n \tilde{Y}_j$, that the $3 \times n$ sets $\tilde{Q}_j, \tilde{X}_j, \tilde{O}_j$ are all disjoint, that the n sets Y_j are all disjoint, and that $Q_j \subseteq \tilde{Q}_j, X_j \subseteq \tilde{X}_j$ etc. In addition, let $\tilde{Q} = \cup_{j=1}^n \tilde{Q}_j, \tilde{O} = \cup_{j=1}^n \tilde{O}_j, \tilde{Y} = \cup_{j=1}^n \tilde{Y}_j, O = \cup_{j=1}^n O_j, Y = \cup_{j=1}^n Y_j, \hat{Q} = \cup_{j=1}^n (\tilde{Q}_j \setminus Q_j), \hat{X} = \cup_{j=1}^n (\tilde{X}_j \setminus X_j), \hat{Y} = \tilde{Y} \setminus Y_j$.

Proposition 6 *If for any $\alpha \in \tilde{Q}^\omega$, $\beta \in (\tilde{O} \cup \tilde{Y})^\omega$, $\alpha' \in \hat{X}^\infty$*

$$(1) : S_j \xrightarrow{t} \tilde{S}_j \quad 1 \leq j \leq n,$$

$$(2) : \wedge_{j=1}^n \tilde{f}_j \in \llbracket \tilde{S}_j \rrbracket \wedge (\hat{\otimes}_{j=1}^n \tilde{f}_j)(\alpha) = \beta \wedge \#\alpha/\hat{Q} = \infty \\ \Rightarrow \\ (\|\|_{j=1}^n \tilde{f}_j)(\alpha \frown \beta \frown \alpha')/(O \cup Y) = \beta/(O \cup Y)$$

then

$$(3) : \otimes_{j=1}^n S_j \xrightarrow{t} \otimes_{j=1}^n \tilde{S}_j.$$

Proof: Assume (1), (2). Let

$$(4) : \tilde{f}_j \in \llbracket \tilde{S}_j \rrbracket \quad 1 \leq j \leq n.$$

(1), (4) imply there are functions f_1, \dots, f_n such that

$$(5) : f_j \in \llbracket S_j \rrbracket \quad 1 \leq j \leq n, \\ (6) : \#\alpha/(\hat{Q}_j \cup \hat{X}_j) = \infty \Rightarrow \tilde{f}_j(\alpha)/(O_j \cup Y_j) = f_j(\alpha) \quad 1 \leq j \leq n.$$

It is enough to show that

$$(7) : \#\alpha/\hat{Q} = \infty \Rightarrow (\otimes_{j=1}^n \tilde{f}_j)(\alpha)/O = (\otimes_{j=1}^n f_j)(\alpha).$$

Given some $\alpha \in \tilde{Q}^\omega$ such that

$$(8) : \#\alpha/\hat{Q} = \infty.$$

Assume that

$$(9) : (\hat{\otimes}_{j=1}^n \tilde{f}_j)(\alpha) = \beta.$$

The monotonicity of the functions $\tilde{f}_1, \dots, \tilde{f}_n$ implies there are chains such that

$$(10) : \tilde{\alpha}_1 = \alpha \frown (\tilde{X} \cup \tilde{Y} \mapsto \epsilon), \\ (11) : \tilde{\alpha}_{j+1} = \alpha \frown \tilde{\beta}_j, \\ (12) : \tilde{\beta}_j = (\|\|_{l=1}^n \tilde{f}_l)(\tilde{\alpha}_j).$$

As in the proof of Proposition 2 it follows straightforwardly by induction on j that

$$(13) : \tilde{\beta}_j/(O \cup Y) \sqsubseteq \beta.$$

(13) and the continuity of \sqcup imply

$$(14) : \sqcup \tilde{\beta} / (O \cup Y) \sqsubseteq \beta.$$

Since $\tilde{\beta}$ characterizes the Kleene-chain, it also holds that

$$(15) : (\hat{\otimes}_{j=1}^n \tilde{f}_j)(\alpha) = \sqcup \tilde{\beta}.$$

Assume

$$(16) : \alpha' \in \hat{X}^\infty.$$

(2), (4), (8), (15), (16) imply

$$(17) : (\|_{j=1}^n \tilde{f}_j)(\alpha \frown (\sqcup \tilde{\beta}) \frown \alpha') / (O \cup Y) = \sqcup \tilde{\beta} / (O \cup Y).$$

(6), (8), (16) imply

$$(18) : (\|_{j=1}^n \tilde{f}_j)(\alpha \frown (\sqcup \tilde{\beta}) \frown \alpha') / (O \cup Y) = (\|_{j=1}^n f_j)(\alpha \frown (\sqcup \tilde{\beta}) \frown \alpha').$$

(16), (17), (18) imply

$$(19) : (\|_{j=1}^n f_j)(\alpha \frown (\sqcup \tilde{\beta}) \frown \alpha') = (\|_{j=1}^n f_j)(\alpha \frown (\sqcup \tilde{\beta})) = \sqcup \tilde{\beta} / (O \cup Y).$$

(9), (14), (19) imply

$$(20) : (\|_{j=1}^n f_j)(\alpha \frown (\sqcup \tilde{\beta})) = (\hat{\otimes}_{j=1}^n f_j)(\alpha).$$

(19), (20) imply

$$(21) : \sqcup \tilde{\beta} / (O \cup Y) = (\hat{\otimes}_{j=1}^n f_j)(\alpha).$$

(15), (21) imply

$$(22) : (\hat{\otimes}_{j=1}^n \tilde{f}_j)(\alpha) / (O \cup Y) = (\hat{\otimes}_{j=1}^n f_j)(\alpha)$$

(22) and the fact that \otimes is equal to $\hat{\otimes}$ minus hiding imply (7).

end of proof

Example 3 Introducing Channel Constraints:

To see how Proposition 6 can be employed in practise, assume we have a network consisting of n specifications composed in sequence as indicated by Figure 7. The network communicates with its environment via x_0 and x_n . Each specification S_j characterizes a component which applies

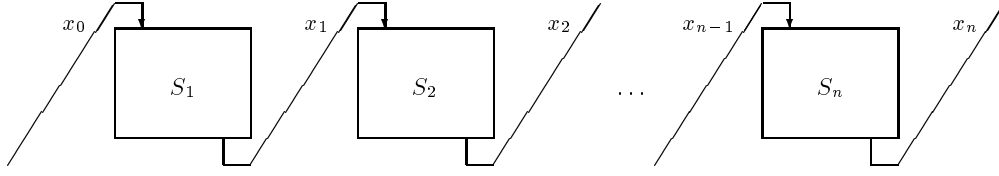


Figure 7: Asynchronous Network

an operation represented by the function g_j to each message received on x_{j-1} and outputs the result along x_j . This means that the j 'th component is required to satisfy the specification

$$S_j \stackrel{\text{def}}{=} (\{x_{j-1}\}, \{x_j\}, x_j = \text{map}(x_{j-1}, g_j)),$$

where $\text{map}(s, f)$ is equal to the stream we get by applying the function f to each element of the stream s .

Assume we want to implement this network employing some architecture based on hand-shake communication. We then get the network pictured in Figure 8.

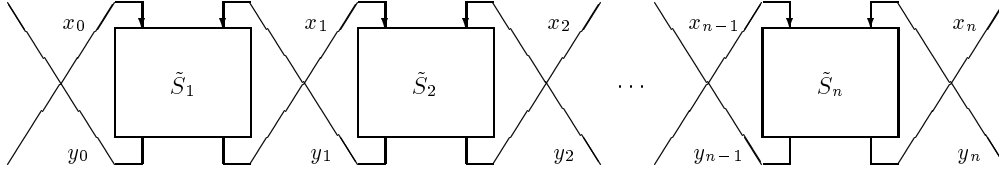


Figure 8: Synchronous Network

Each of these new components is characterized by

$$\tilde{S}_j \stackrel{\text{def}}{=} (\{x_{j-1}, y_j\}, \{x_j, y_{j-1}\}, \tilde{R}_j)$$

where

$$\tilde{R}_j \stackrel{\text{def}}{=} x_j \sqsubseteq \text{map}(x_{j-1}, g_j) \wedge \#x_j = \min\{\#x_{j-1}, \#y_j + 1\} \wedge y_{j-1} = x_j.$$

Clearly

$$S_j \stackrel{t}{\sim} \tilde{S}_j \quad 1 \leq j \leq n.$$

Since each \tilde{R}_j is deterministic and

$$\#y_n = \infty \wedge (\bigwedge_{j=1}^n \tilde{R}_j) \Rightarrow \bigwedge_{j=1}^n \tilde{R}_j^{[y_j \sim y'_j]},$$

where $\tilde{R}_j^{[y_j \sim y'_j]}$ denotes that each occurrence of y_j in \tilde{R}_j is replaced by the expression $y_j \sim y'_j$, it follows by Proposition 6 that

$$\otimes_{j=1}^n S_j \overset{t}{\rightsquigarrow} \otimes_{j=1}^n \tilde{S}_j \quad (***) .$$

Hand-shake communication is of course not the only way to synchronize the network pictured in Figure 7. Assume the architecture chosen for the implementation offers channels which can store up to 100 messages. We may then redefine the input/output relation of \tilde{S}_j as below:

$$\begin{aligned} \tilde{R}_j \stackrel{\text{def}}{=} x_j \sqsubseteq \text{map}(x_{j-1}, g_j) \wedge \\ \#x_j = \min\{\#x_{j-1}, (\#y_j + 1) \times 100\} \wedge \#y_{j-1} = \#x_j // 100. \end{aligned}$$

Again it follows straightforwardly by Proposition 6 that this is a correct total refinement.

Of course that the network in Figure 8 is a total refinement of the network in Figure 7 does not mean that buffer overflow cannot occur. It remains the developer's responsibility to formulate a correct protocol. For example if

$$\tilde{R}_j \stackrel{\text{def}}{=} x_j = \text{map}(x_{j-1}, g_j)$$

then (***) although there is no synchronization between the components in $\otimes_{j=1}^n \tilde{S}_j$ — the output along x_j is completely independent of the input along y_j . On the other hand, if

$$\tilde{R}_j \stackrel{\text{def}}{=} x_j \sqsubseteq \text{map}(x_{j-1}, g_j) \wedge \#x_j = \min\{\#x_{j-1}, \#y_j + 1\} \wedge y_{j-1} = x_{j-1},$$

then (***) holds, and buffer overflow cannot occur. However, there is no correct implementation of \tilde{S}_j which require only a bounded amount of local memory. Thus in this case the buffer overflow problem has been transferred from the channels to the components.

□

6 Conclusions

Since Kahn's influential paper on the modeling of deterministic dataflow networks was published in 1974 [Kah74], a number of authors has proposed formalisms for the specification of reactive systems based on asynchronous communication via unbounded, directed channels (see for example [Kel78], [BA81], [Par83], [Kok87], [Jon87], [LT87], [BDD⁺93]). The unboundedness assumption is very useful when specifying and reasoning about systems at an abstract level. However, at some point in a development this assumption must be discharged in the sense that the communication is synchronized in order to avoid channel overflow. The contribution of this paper is the formulation of a refinement relation allowing this transition from unbounded to bounded communication to be conducted in a natural way.

We first proposed a relation for partial correctness — called partial refinement, which then was generalized into a refinement relation for total correctness — called total refinement. Partial refinement guarantees only the preservation of safety properties. To be sure that both safety and liveness properties are preserved, the principle of total refinement is required.

Partial refinement was proved to be reflexive, transitive and a congruence with respect to the composition operator on specifications. It was shown that total refinement characterizes a reflexive and transitive relation, but does not satisfy the congruence property. The problem was found to be that deadlocks can be introduced when feedback loops are added — deadlock in the

sense that the least fixpoint is reached too early. Nevertheless, we have shown that rules can be formulated which allow for top-down system development in a modular style — modular in the sense that design decisions can be checked at the point in a development where they are made, i.e., on the basis of the component specifications alone, without knowing how they are finally implemented. In addition to the obvious premise that each (concrete) component specification is a total refinement of the corresponding (abstract) component specification, a freedom from deadlock test must be fulfilled.

As already explained the proposed refinement relation is mainly suited for synchronization which does not distinguish between different types of acknowledgements (or demands) — what matters is whether an acknowledgement (demand) is received or not, i.e., not whether an acknowledgement (demand) of a special sort is received.

There are several ways of generalizing total refinement. For example the following definition

$$\forall \tilde{f} \in \llbracket \tilde{S} \rrbracket \Rightarrow \exists f \in \llbracket S \rrbracket : \forall \alpha \in \tilde{Q}^* : \exists \alpha' \in (\tilde{Q} \setminus Q)^\omega \Rightarrow \tilde{f}(\alpha \frown \alpha')/O = f(\alpha),$$

is stronger, but leads to more complicated proof-obligations based on an assumption/commitment style of reasoning [AL90], [SDW93].

Another approach is to try to combine the ideas of this paper with what [Bro93] calls interface interaction refinement, which can be understood as behavioral refinement modulo two representation specifications allowing also the input and the output histories (including the number of channels and their types) to be refined. When the representation specifications are sufficiently constrained interface interaction refinement is a congruence with respect to the composition operator on specifications [Bro92]. We believe the resulting relation would be sufficiently general to support the more sophisticated types of synchronization like for example demand driven communication in terms of the class-climbing principle [AvT87] where the demands sent along certain channels are required to be fairly distributed over a set of classes.

7 Acknowledgements

The author has benefited from discussions with Manfred Broy and Bernhard Schätz.

References

- [AL88] M. Abadi and L. Lamport. The existence of refinement mappings. Technical Report 29, Digital, Palo Alto, 1988.
- [AL90] M. Abadi and L. Lamport. Composing specifications. Technical Report 66, Digital, Palo Alto, 1990.
- [AvT87] J.K. Annot and R.A.H. van Twist. A novel deadlock free and starvation free packet switching communication processor. In *Lecture Notes in Computer Science 258*, pages 68–85, 1987.
- [BA81] J. D. Brock and W. B. Ackermann. Scenarios: A model of non-determinate computation. In J. Diaz and I. Ramos, editors, *Proc. Formalization of Programming Concepts, Lecture Notes in Computer Science 107*, pages 252–259, 1981.
- [BD92] M. Broy and C. Dendorfer. Modelling operating system structures by timed stream processing functions. *Journal of Functional Programming*, 2:1–21, 1992.

- [BDD⁺93] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber. The design of distributed systems — an introduction to Focus (revised version). Technical Report SFB 342/2/92 A, Technische Universität München, 1993.
- [Bro92] M. Broy. Compositional refinement of interactive systems. Technical Report 89, Digital, Palo Alto, 1992.
- [Bro93] M. Broy. (Inter-) Action refinement: The easy way. In M. Broy, editor, *Proc. Program Design Calculi, Summerschool, Marktoberdorf*, pages 121–158. Springer, 1993.
- [HJH90] J. He, M. Josephs, and C. A. R. Hoare. A theory of synchrony and asynchrony. In M. Broy and C. B. Jones, editors, *Proc. IFIP WG 2.2/2.3 Working Conference on Programming Concepts and Methods*, pages 459–478, 1990.
- [Hoa72] C. A. R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1:271–282, 1972.
- [Jon87] B. Jonsson. *Compositional Verification of Distributed Systems*. PhD thesis, Uppsala University, 1987.
- [Kah74] G. Kahn. The semantics of a simple language for parallel programming. In J.L. Rosenfeld, editor, *Proc. Information Processing 74*, pages 471–475. North-Holland, 1974.
- [Kel78] R. M. Keller. Denotational models for parallel programs with indeterminate operators. In E. J. Neuhold, editor, *Proc. Formal Description of Programming Concepts*, pages 337–366. North-Holland, 1978.
- [Kok87] J. N. Kok. A fully abstract semantics for data flow nets. In *Proc. PARLE'87, Lecture Notes in Computer Science 259*, pages 351–368, 1987.
- [LT87] N. Lynch and M. R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In *Proc. 6th Annual ACM Symposium on Principles of Distributed Computing*, pages 137–151, 1987.
- [OG76] S. Owicki and D. Gries. An axiomatic proof technique for parallel programs. *Acta Informatica*, 6:319–340, 1976.
- [Par83] D. Park. The “fairness” problem and nondeterministic computing networks. In J. W. de Bakker and J van Leeuwen, editors, *Proc. 4th Foundations of Computer Science, Mathematical Centre Tracts 159*, pages 133–161. Mathematisch Centrum Amsterdam, 1983.
- [PJ91] P. K. Pandya and M. Joseph. P-A logic — a compositional proof system for distributed programs. *Distributed Computing*, 5:37–54, 1991.
- [SDW93] K. Stølen, F. Dederichs, and R. Weber. Assumption/commitment rules for networks of asynchronously communicating agents. Technical Report SFB 342/2/93 A, Technische Universität München, 1993.
- [Stø91] K. Stølen. A method for the development of totally correct shared-state parallel programs. In J. C. M. Baeten and J. F. Groote, editors, *Proc. CONCUR'91, Lecture Notes in Computer Science 527*, pages 510–525, 1991.