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.

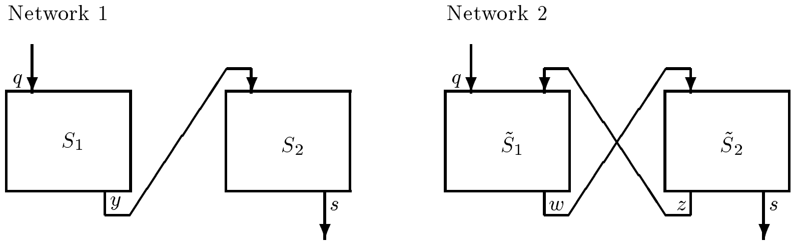

Fig. 1. Introducing Synchronization

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 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.

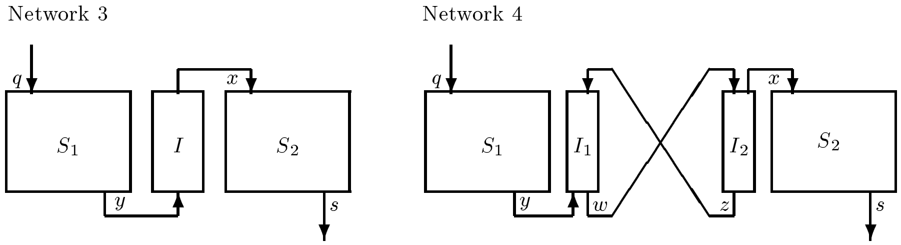

Fig. 2. Naive Transformation

In fact with respect to the two networks of Fig. 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 Fig. 2. The soundness of this refinement step is obvious.
- Step 2: Refine the identity specification into two sub-specifications I_1 and I_2 which communicate in accordance with the desired protocol. We then get Network 4 of Fig. 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 Fig. 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, do not seem to offer the required flexibility. (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 (and also any conjunction of safety and liveness properties) — preserves in the sense that any implementation of the more concrete specification can be understood as an implementation of the more abstract specification. Thus a total refinement is also a partial refinement.

Total refinement allows for the introduction of both acknowledgment based and demand-driven synchronization. However, it is suited only for synchronization protocols which do not depend upon that acknowledgments (demands) sent along a channel are fairly distributed over sets of acknowledgments (demands).

Of course the use of hand-shake synchronization is not the only way to avoid buffer overflow — another alternative is to synchronize the computation by imposing real-time constraints on the behavior of processes and channels. However, this alternative can be used only if the programming language in which the specified system is to be implemented supports the realization of such constraints.

The investigations are conducted in the context of data-flow networks modeled 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 Sect. 3. Then partial and total refinement are the subjects of Sects. 4 and 5 Finally, Sect. 6 contains a summary and discusses a possible generalization.

2 Basic Notations

N denotes the set of natural numbers, and N_+ denotes $\mathsf{N} \setminus \{0\}$. 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 (and this is exploited in Ex. 4).

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

- ϵ denotes the empty stream;
- $\mathsf{ft}(r)$ 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 to 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 $\mathsf{min}\{\#\alpha(i) \,|\, i \in I\}$. 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

$$\{a \mapsto r, b \mapsto s\}$$

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

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

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

is required to be monotonic:

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

and continuous:

$$\text{for all chains } \delta \text{ of named stream tuples } : f(\sqcup\delta) = \sqcup\{f(\delta_j) \,|\, j \in \mathsf{N}_+\}.$$

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 $\alpha \in Q^\omega$, $f(\alpha) \stackrel{\mathsf{def}}{=} f(\alpha/I)$.

Given two stream processing functions

$$f \in I^\omega \to O^\omega, \qquad \tilde{f} \in \tilde{I}^\omega \to \tilde{O}^\omega,$$

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

$$(I \cup \tilde{I})^\omega \to (O \cup \tilde{O})^\omega,$$

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

$$((I \setminus \tilde{O}) \cup (\tilde{I} \setminus O))^\omega \to ((O \setminus \tilde{I}) \cup (\tilde{O} \setminus I))^\omega,$$
$$((I \setminus \tilde{O}) \cup (\tilde{I} \setminus O))^\omega \to (O \cup \tilde{O})^\omega,$$

respectively, such that

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

where β is the least fix-point of $(f \parallel \tilde{f})(\alpha \frown \beta) = \beta$ with respect to $\sqsubseteq$.

It follows straightforwardly that $f \parallel \tilde{f}$, $f \otimes \tilde{f}$ and $f \hat{\otimes} \tilde{f}$ are stream processing functions.

In Fig. 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.

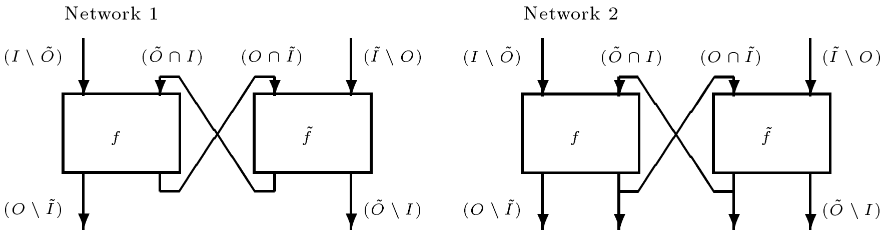
Network 1
$(I \setminus \tilde{O})$
$(\tilde{O} \cap I)$
$(O \cap \tilde{I})$
$(\tilde{I} \setminus O)$
f
$\tilde{f}$
$(O \setminus \tilde{I})$
$(\tilde{O} \setminus I)$
Network 2
$(I \setminus \tilde{O})$
$(\tilde{O} \cap I)$
$(O \cap \tilde{I})$
$(\tilde{I} \setminus O)$
f
$\tilde{f}$
$(O \setminus \tilde{I})$
$(\tilde{O} \setminus I)$

Fig. 3. Networks Relating the Operators $\otimes$ and $\hat{\otimes}$

Given $n > 1$ stream processing functions

$$f_j \in I_j^\omega \to O_j^\omega \qquad 1 \leq j \leq n,$$

such that $I_j \cap O_j = \emptyset$ and $l \neq k$ implies $I_l \cap I_k = O_l \cap O_k = \emptyset$, then $\otimes_{j=1}^{n} f_j$ is a short-hand for $f_1 \otimes \ldots \otimes f_n$. 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. The identifiers in I and O name the input channels and the output channels, respectively. We refer to these identifiers as the input and the output identifiers. Moreover, (I, O) is called the specification's interface. In R each such identifier is of type stream. 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.

For example the specification

$$(\{a\}, \{c, d, e\}, \#c = \#a \wedge d = e = 0 \,\&\, a)$$

characterizes a component with one input channel a and three output channels c, d, e. Along the channel c this component outputs exactly one (arbitrary) message for each message it receives on a, and along the channels d and e the component first outputs a 0 and thereafter any message received on a.

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

$$[\![\ S\ ]\!] \overset{\mathsf{def}}{=} \{f \in I^\omega \rightarrow O^\omega \mid \forall \alpha \in I^\omega : (\alpha \frown f(\alpha)) \models R\},$$

where for any named stream tuple $\beta \in Q^\omega$ and formula P whose free variables are contained in Q, $\beta \models P$ iff P evaluates to true when each identifier $i \in Q$ is interpreted as $\beta(i)$.

The basic refinement relation is represented by $\rightsquigarrow$. It holds only for specifications whose interfaces are identical. Given two specifications S_1 and S_2, then $S_1 \rightsquigarrow S_2$ iff $[\![\ S_2\ ]\!] \subseteq [\![\ S_1\ ]\!]$. 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 $S_1 \overset{\mathsf{def}}{=} (I_1, O_1, R_1)$ and $S_2 \overset{\mathsf{def}}{=} (I_2, O_2, R_2)$, such that

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

then $S_1 \otimes S_2$ represents the network pictured in Fig. 4. The channels modeled 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

$$[\![\ S_1 \otimes S_2 \ ]\!] \overset{\mathsf{def}}{=} \{f_1 \otimes f_2 \mid f_1 \in [\![\ S_1 \ ]\!] \wedge f_2 \in [\![\ S_2 \ ]\!]\}.$$

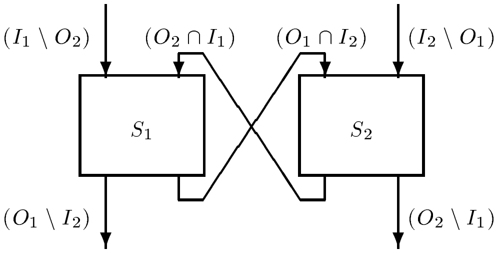

Fig. 4. $S_1 \otimes S_2$

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

The properties characterized by a specification can be split into several classes. For example, there is a long tradition for distinguishing between safety and liveness properties [AS85]. Informally speaking:

- a safety property characterizes what a correct implementation is not allowed to do,
- a liveness property characterizes what a correct implementation is required to do.

Given a specification $S \overset{\mathsf{def}}{=} (I, O, R)$ then S characterizes a safety property iff

$$\forall \alpha \in I^{\omega} : \forall \beta \in O^{\omega} : (\alpha \frown \beta) \models R \Leftrightarrow \forall \beta' \in O^{*} : \beta' \sqsubseteq \beta \Rightarrow (\alpha \frown \beta') \models R,$$

and a liveness property iff

$$\forall \alpha \in I^{\omega} : \forall \beta \in O^{*} : \exists \beta' \in O^{\omega} : \beta \sqsubseteq \beta' \wedge (\alpha \frown \beta') \models R.$$

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

$$S \stackrel{\mathsf{def}}{=} (Q, O, R), \qquad \tilde{S} \stackrel{\mathsf{def}}{=} (\tilde{Q}, \tilde{O}, \tilde{R}),$$

where $Q \subseteq \tilde{Q}$ and $O \subseteq \tilde{O}$. We want to characterize what it means for $\tilde{S}$ to refine S. If only the "old" channels are considered one might expect this to be equivalent to insisting that for each function $\tilde{f} \in [\![\ \tilde{S}\ ]\!]$ there is a function $f \in [\![\ S\ ]\!]$ which behaves in the same way as $\tilde{f}$. Since by definition $f(\alpha)$ is equal to $f(\alpha/Q)$ this suggests:

$$\forall \tilde{f} \in [\![\ \tilde{S}\ ]\!] : \exists f \in [\![\ S\ ]\!] : \forall \alpha \in \tilde{Q}^{\omega} : \tilde{f}(\alpha)/O = f(\alpha).$$

However, due to the synchronization conducted via the new channels the output can be halted too early because the required acknowledgments have not been received. Thus, in the general case, unless we make certain assumptions about the environment's behavior, the insistence upon equality is too strong. On the other hand, if only safety properties are considered, the following constraint is sufficient:

$$\forall \tilde{f} \in [\![\ \tilde{S}\ ]\!] : \exists f \in [\![\ S\ ]\!] : \forall \alpha \in \tilde{Q}^{\omega} : \tilde{f}(\alpha)/O \sqsubseteq f(\alpha).$$

If $\tilde{S}$ and S are related in this way, we say that $\tilde{S}$ is a partial refinement of S, and we write $S \overset{p}{\rightsquigarrow} \tilde{S}$. 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 $f(\alpha)$.

The rest of this section is devoted to partial refinement. In the next section, we will introduce a more general refinement relation which guarantees equality under the assumption that sufficiently many acknowledgments are received.

Clearly, if S is a safety property and $S \overset{p}{\rightsquigarrow} \tilde{S}$ then $\tilde{S}$ behaves in accordance with S with respect to the interface of S. Thus $\overset{p}{\rightsquigarrow}$ preserves safety properties in the sense that $\tilde{S}$ does not falsify S. Note that this does not mean that $\tilde{S}$ has to be a safety property. On the other hand, if S is a liveness property then there is no guarantee that $\tilde{S}$ behaves in accordance with S. Thus $\overset{p}{\rightsquigarrow}$ preserves safety properties but not liveness properties.

It is straightforward to prove that partial refinement is reflexive and transitive, and below we show that it is also 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

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

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

$$S_j \overset{p}{\rightsquigarrow} S'_j \qquad 1 \leq j \leq n,$$

then it also holds that

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

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. This implies that the principle of partial refinement is well-suited for top-down system development.

Before stating the general congruence property for partial refinement, we prove an intermediate result, whose conclusion (3) is visualized by Fig. 5. Thus we have four specifications $S_1, S_2, \tilde{S}_1, \tilde{S}_2$. Their interfaces 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 $Q \subseteq \tilde{Q}$, $X \subseteq \tilde{X}$, $O \subseteq \tilde{O}$, $Y \subseteq \tilde{Y}$, $Z \subseteq \tilde{Z}$, $K \subseteq \tilde{K}$.

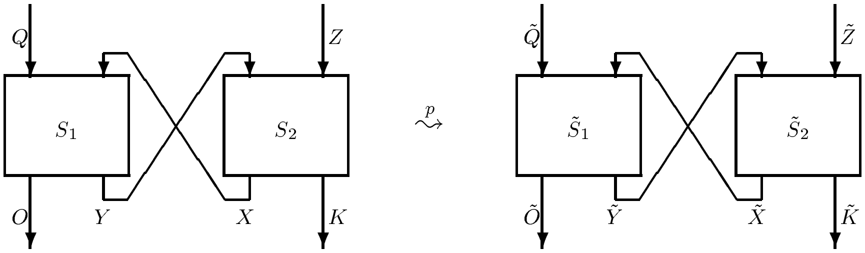

Fig. 5. Partial Refinement

Proposition 1. *If*

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

then

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

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

$(6) : f_1 \in \llbracket\ S_1\ \rrbracket,$

$(7) : f_2 \in \llbracket\ S_2\ \rrbracket,$

$(8) : \forall \alpha \in (\tilde{Q} \cup \tilde{X})^\omega : \tilde{f}_1(\alpha)/(O \cup Y) \sqsubseteq f_1(\alpha),$

$(9) : \forall \alpha \in (\tilde{Y} \cup \tilde{Z})^\omega : \tilde{f}_2(\alpha)/(X \cup K) \sqsubseteq f_2(\alpha).$

(3) follows if it can be shown that

$(10) : \forall \alpha \in (\tilde{Q} \cup \tilde{Z})^\omega : (\tilde{f}_1 \otimes \tilde{f}_2)(\alpha)/(O \cup K) \sqsubseteq (f_1 \otimes f_2)(\alpha).$

Given some $\alpha \in (\tilde{Q} \cup \tilde{Z})^\omega$ and let $\beta \in (O \cup Y \cup X \cup K)^\omega$ be defined by

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

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 $I \subseteq Q$.)
(12), (13), (14) imply

$(15) : (\tilde{f}_1 \hat{\otimes} \tilde{f}_2)(\alpha) = \sqcup\tilde{\beta}.$

We want to prove that

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

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

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

We show that

$(18) : \tilde{\beta}_{k+1}/(O \cup Y \cup X \cup K) \sqsubseteq \beta.$

(13) implies that

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

(19) and the definition of $\parallel$ imply that

$$(20): \tilde{\beta}_{k+1}/(O \cup Y \cup X \cup K) = \tilde{f}_1(\tilde{\alpha}_{k+1})/(O \cup Y) \frown \tilde{f}_2(\tilde{\alpha}_{k+1})/(X \cup K).$$

(8), (9), (20) imply

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

(14), (21) imply

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

(17), (22) and the monotonicity of f_1 and f_2 imply

$$(23): \tilde{\beta}_{k+1}/(O \cup Y \cup X \cup K) \sqsubseteq f_1(\alpha \frown \beta) \frown f_2(\alpha \frown \beta).$$

(11), (23) imply (18). This ends the proof of (16).
(16) and the definition of $\sqcup$ imply

$$(24): \sqcup\tilde{\beta}/(O \cup Y \cup X \cup K) \sqsubseteq \beta.$$

(11), (15), (24) and the fact that $\otimes$ is equal to $\hat{\otimes}$ plus hiding imply (10).

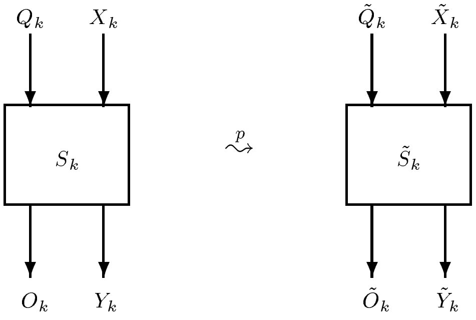

Fig. 6. Partial Refinement of the k'th Component Specification

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

Proposition 2. *If*

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

then

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

Proof. Follows from Prop. 1 by induction on n.

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

$$S \overset{\mathsf{def}}{=} (Q, O, R), \qquad\qquad \tilde{S} \overset{\mathsf{def}}{=} (\tilde{Q}, \tilde{O}, \tilde{R}),$$

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

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

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 acknowledgment (demand) is received or not, and not upon what sort of acknowledgment (demand) is received, this is exactly what is needed. Note that this antecedent can be thought of as an environment assumption. We will later discuss the generality of total refinement in more detail.

We first prove that total refinement degenerates to behavioral refinement if the interface is not extended, and that total refinement implies partial refinement.

Proposition 3. *Given two specifications S and $\tilde{S}$ whose interfaces are characterized by (Q, O) and $(\tilde{Q}, \tilde{O})$, respectively. Then:*

$$\begin{aligned} &(1) : Q = \tilde{Q} \wedge O = \tilde{O} \Rightarrow (S \overset{t}{\leadsto} \tilde{S} \Leftrightarrow S \leadsto \tilde{S}),\\ &(2) : S \overset{t}{\leadsto} \tilde{S} \Rightarrow S \overset{p}{\leadsto} \tilde{S}. \end{aligned}$$

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

$$(3) : S \overset{t}{\leadsto} \tilde{S}.$$

It must be shown that

$$(4) : S \overset{p}{\leadsto} \tilde{S}.$$

Let

$$(5) : \tilde{f} \in \llbracket\ \tilde{S}\ \rrbracket.$$

(3), (5) imply there is an f such that

$$\begin{aligned} &(6) : f \in \llbracket\ S\ \rrbracket,\\ &(7) : \forall \alpha \in \tilde{Q}^{\omega} : \#\alpha/(\tilde{Q} \setminus Q) = \infty \Rightarrow \tilde{f}(\alpha)/O = f(\alpha). \end{aligned}$$

Given some arbitrary $\alpha \in \tilde{Q}^{\omega}$. (4) follows if it can be shown that

$$(8) : \tilde{f}(\alpha)/O \sqsubseteq f(\alpha).$$

If $\#\alpha/(\tilde{Q} \setminus Q) = \infty$ then (8) follows trivially from (7). Otherwise, there is an $\alpha' \in \tilde{Q}^{\omega}$ such that

$$\begin{aligned} &(9) : \alpha'/Q = \alpha/Q,\\ &(10) : \alpha \sqsubseteq \alpha',\\ &(11) : \#\alpha'/(\tilde{Q} \setminus Q) = \infty. \end{aligned}$$

(7), (9), (11) imply

$$(12) : \tilde{f}(\alpha')/O = f(\alpha'/Q) = f(\alpha/Q) = f(\alpha).$$

(10), (12) and the monotonicity of $\tilde{f}$ imply (8).

The next step is to prove that $\overset{t}{\leadsto}$ is reflexive and transitive.

Proposition 4. *Given three specifications S_1, S_2, S_3 whose interfaces 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*

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

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

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

Let f_3 be such that

$$(5): f_3 \in [\![S_3]\!].$$

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

$$(6): f_2 \in [\![S_2]\!],$$
$$(7): \forall \alpha \in (Q_3)^\omega : \#\alpha/(Q_3 \setminus Q_2) = \infty \Rightarrow f_3(\alpha)/O_2 = f_2(\alpha).$$

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

$$(8): f_1 \in [\![S_1]\!],$$
$$(9): \forall \alpha \in (Q_2)^\omega : \#\alpha/(Q_2 \setminus Q_1) = \infty \Rightarrow f_2(\alpha)/O_1 = f_1(\alpha).$$

Given an $\alpha \in (Q_3)^\omega$ such that

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

(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

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

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

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

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

Clearly $S_1 \overset{t}{\leadsto} \tilde{S}_1$ and $S_2 \overset{t}{\leadsto} \tilde{S}_2$. Unfortunately, for all $f \in [\![S_1 \hat{\otimes} S_2]\!]$ and $\tilde{f} \in [\![\tilde{S}_1 \hat{\otimes} \tilde{S}_2]\!]$, and any nonempty stream s, it holds that

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

Thus

$$\begin{aligned}
&f \in [\![S_1 \otimes S_2]\!] \Rightarrow f(\{q \rightarrow s\}) = \{k \mapsto s\}\\
&\tilde{f} \in [\![\tilde{S}_1 \otimes \tilde{S}_2]\!] \Rightarrow f(\{q \rightarrow s\}) = \{k \mapsto \mathsf{ft}(s) \,\&\, \epsilon\}.
\end{aligned}$$

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

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

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 Ex. 1 the following condition is obviously sufficient:

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

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 Ex. 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 fix-point — new in the sense that the least fix-point is reached too early. For the refinement step conducted in Ex. 1, it therefore seems sensible to require that for any $\alpha \in \{q\}^\omega$, $\tilde{f}_1 \in \llbracket\ \tilde{S}_1\ \rrbracket$, $\tilde{f}_2 \in \llbracket\ \tilde{S}_2\ \rrbracket$:

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

This condition states that when the least fix-point 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 fix-point 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 Ex. 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 Ex. 1, $(***)$ seems to be a reasonable proof obligation. The next step is to figure out how this obligation should look in the general case.

Example 2. Let S_1, S_2 and $\tilde{S}_1$ be as in Ex. 1, and let

$$\tilde{S_2} \overset{\mathsf{def}}{=} (\{y, z\}, \{x, k\}, k \sqsubseteq y \wedge \#k = \#x = \mathsf{min}\{\#y, \#z\}).$$

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 s is a stream such that $\#s > 1$. Clearly

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

Moreover

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

Thus ($***$) is not satisfied.

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

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

This motivates the next proposition, which characterizes a condition under which a total refinement corresponding to Fig. 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* $\alpha \in (\tilde{Q} \cup \tilde{Z})^{\omega}$, $\beta \in (\tilde{O} \cup \tilde{Y} \cup \tilde{X} \cup \tilde{K})^{\omega}$, $\alpha' \in (\hat{X} \cup \hat{Y})^{\infty}$

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

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

$$(5) : \tilde{f_1} \in [\![\tilde{S}_1]\!],$$
$$(6) : \tilde{f_2} \in [\![\tilde{S}_2]\!].$$

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

$$(7) : f_1 \in [\![S_1]\!],$$
$$(8) : f_2 \in [\![S_2]\!],$$
$$(9) : \forall \alpha \in (\tilde{Q} \cup \tilde{X})^\omega : \#\alpha/(\hat{Q} \cup \hat{X}) = \infty \Rightarrow \tilde{f_1}(\alpha)/(O \cup Y) = f_1(\alpha),$$
$$(10) : \forall \alpha \in (\tilde{Y} \cup \tilde{Z})^\omega : \#\alpha/(\hat{Y} \cup \hat{Z}) = \infty \Rightarrow \tilde{f_2}(\alpha)/(X \cup K) = f_2(\alpha).$$

It is enough to show that

$$(11) : \forall \alpha \in (\tilde{Q} \cup \tilde{Z})^\omega : \#\alpha/(\hat{Q} \cup \hat{Z}) = \infty \Rightarrow$$
$$(\tilde{f_1} \otimes \tilde{f_2})(\alpha)/(O \cup K) = (f_1 \otimes f_2)(\alpha).$$

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

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

Let $\beta \in (O \cup Y \cup X \cup K)^\omega$ be such that

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

The monotonicity of $\tilde{f_1}$ and $\tilde{f_2}$ implies there are chains $\tilde{\alpha}, \tilde{\beta}$ such that

$$(14) : \tilde{\alpha}_1 = \alpha \frown (\tilde{X} \cup \tilde{Y} \mapsto \epsilon),$$
$$(15) : \tilde{\alpha}_{j+1} = \alpha \frown \tilde{\beta}_j,$$
$$(16) : \tilde{\beta}_j = (\tilde{f_1} \parallel \tilde{f_2})(\tilde{\alpha}_j).$$

As in the proof of Prop. 1 it follows straightforwardly by induction on j that

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

(17) and the definition 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

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

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

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

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

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

(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

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

(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 $\hat{\otimes}$ plus hiding imply (11).

It can be argued that the freedom from deadlock test (3) of Prop. 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:

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

However, whenever we run into such a problem, which seems to be a rather artificial one, there are specifications $S_1', S_2', \tilde{S}_1', \tilde{S}_2'$ such that

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

holds, and

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

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

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

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 Prop. 5 is (relative, semantic) complete modulo a (relative, semantic) complete set of rules for behavioral refinement.

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

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

Thus in order to use Prop. 5 it is in most cases not necessary to characterize the least fix-point solution.

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

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

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

$$\begin{aligned}
&(2) : \wedge_{j=1}^{n} \tilde{f}_j \in [\![\ \tilde{S}_j \ ]\!] \wedge (\hat{\otimes}_{j=1}^{n} \tilde{f}_j)(\alpha) = \beta \wedge \#\alpha/\hat{Q} = \infty\\
&\qquad \Rightarrow\\
&\qquad (\|_{j=1}^{n} \ \tilde{f}_j)(\alpha \frown \beta \frown \alpha')/(O \cup Y) = \beta/(O \cup Y)
\end{aligned}$$

then

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

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

$$(4) : \tilde{f}_j \in [\![\ \tilde{S}_j\ ]\!] \qquad 1 \leq j \leq n.$$

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

$$(5) : f_j \in [\![\ S_j\ ]\!] \qquad 1 \leq j \leq n,$$
$$(6) : \forall \alpha \in (\tilde{Q}_j \cup \tilde{X}_j)^{\omega} :$$
$$\#\alpha/(\hat{Q}_j \cup \hat{X}_j) = \infty \Rightarrow \tilde{f}_j(\alpha)/(O_j \cup Y_j) = f_j(\alpha) \qquad 1 \leq j \leq n.$$

It is enough to show that

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

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

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

Let $\beta \in (O \cup Y)^{\omega}$ be such that

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

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

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

As in the proof of Prop. 1 it follows straightforwardly by induction on j that

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

(13) and the definition 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 $\hat{\otimes}$ plus hiding imply (7).

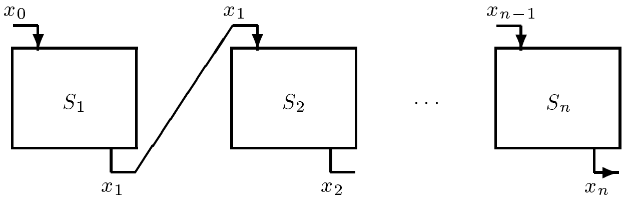

Fig. 7. Asynchronous Network

Example 3. To see how Prop. 6 can be employed in practice, assume we have a network consisting of n specifications composed in sequence as indicated by Fig. 7. The network communicates with its environment via x_0 and x_n. Each specification S_j characterizes a component which applies 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

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

where $map(s, f)$ 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 Fig. 8.

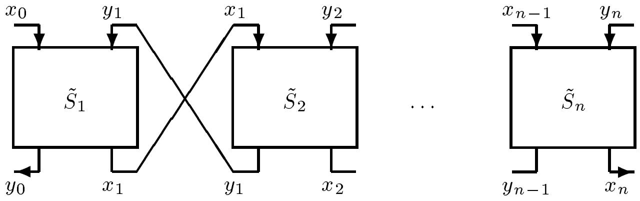

Fig. 8. Synchronous Network

Each of these new components is characterized by

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

where

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

Clearly

$$S_j \overset{t}{\rightsquigarrow} \tilde{S}_j \qquad 1 \leq j \leq n.$$

Since each $\tilde{R}_j$ is deterministic in the sense that for any (x_{j-1}, y_j) there is a unique pair (x_j, y_{j-1}) such that $\tilde{R}_j$ holds, and we have that

$$\#y_n = \infty \wedge (\wedge_{j=1}^{n} \tilde{R}_j) \Rightarrow \wedge_{j=1}^{n-1} \tilde{R}_j[^{y_j}_{y_j \frown y'_j}] \wedge \tilde{R}_n,$$

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

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

This is of course not the only way to synchronize the network pictured in Fig. 7. Assume the architecture chosen for the implementation offers channels which can store up to 100 messages. Given that // is the operator for integer division, we may then redefine the input/output relation of $\tilde{S}_j$ as below:

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

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

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

$$\tilde{R}_j \overset{\mathsf{def}}{=} x_j = \mathit{map}(x_{j-1}, g_j)$$

then (†) holds 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

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

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

As already pointed out in the introduction, the use of hand-shake communication is not the only way to avoid buffer overflow — another alternative is to synchronize the computation by imposing real-time constraints on the behavior of processes and channels. Since in this paper we use a semantic model without any notation of time, we can obviously not handle this kind of refinement. However, by adding ticks to the streams along the lines of [Par83], [BS94], we believe this type of refinement can be dealt with using ordinary behavioral refinement.

In the untimed case some sort of hand-shake algorithm must be used. As mentioned in the introduction total refinement is not sufficiently general to deal with all sorts of hand-shake protocols. To clearly see the limit of our approach, consider the following example.

Example 4. So far in this paper we have worked in an untyped setting. However, our approach can of course be generalized straightforwardly to handle typed channels, and this will be exploited here. Thus assume each channel is assigned a type, and moreover that the definition of $\overset{t}{\rightsquigarrow}$ is modified in the obvious way to take typed channels into account.

In this example, for any set M, we use $M^{\overline{\infty}}$ to denote the set of all infinite streams over M. For any infinite stream s and $k \in \mathsf{N}_+$, we use $s(k)$ to denote the k'th element of s. Finally, the concatenation operator $\frown$ is overloaded to pairs of streams in the obvious point-wise way.

Given the specifications S_1 and S_2 of Ex. 1. Moreover, let $\tilde{S}_1$ and $\tilde{S}_2$ be specifications with the same interfaces as in Ex. 1. Assume $\tilde{S}_1$ and $\tilde{S}_2$ work in a demand-driven fashion, in the sense that whenever $\tilde{S}_2$ is ready to receive a (positive) number of data elements along y it informs $\tilde{S}_1$ about this by sending the corresponding number along x. Thus we assume that the channel x is of type N_+ — the set of positive natural numbers. After having sent a demand $\tilde{S}_2$ waits until the requested number of data elements have been received before it sends a new demand along x. $\tilde{S}_1$, on the other hand, waits until it receives a demand along x, and then outputs the requested number of data elements along y. If the number of data elements demanded by $\tilde{S}_2$ exceeds the number of data elements that is forwarded to $\tilde{S}_1$ by the environment, it outputs what it has received. The input/output relations $\tilde{R}_1$ (of $\tilde{S}_1$) and $\tilde{R}_2$ (of $\tilde{S}_2$) are characterized as below:

$$
\begin{array}{l}
\tilde{R}_1 \overset{\mathsf{def}}{=} y = g(0)(q, x) \\
\quad \mathsf{where} \\
\qquad n = 0 \Rightarrow \\
\qquad\qquad g(n)(q, \epsilon) = \epsilon \\
\qquad\qquad g(n)(q, m \,\&\, x) = g(m)(q, x) \\
\qquad n > 0 \Rightarrow \\
\qquad\qquad g(n)(\epsilon, x) = \epsilon \\
\qquad\qquad g(n)(a \,\&\, q, x) = a \,\&\, g(n-1)(q, x)
\end{array}
$$

$$
\begin{array}{l}
\tilde{R}_2 \overset{\mathsf{def}}{=} \exists p \in (\mathsf{N}_+)^{\overline{\infty}} : (x, k) = f(0)(p, y) \\
\quad \mathsf{where} \\
\qquad n = 0 \Rightarrow \\
\qquad\qquad f(n)(\epsilon, y) = (\epsilon, \epsilon) \\
\qquad\qquad f(n)(m \,\&\, p, y) = (m \,\&\, \epsilon, \epsilon) \frown f(m)(p, y) \\
\qquad n > 0 \Rightarrow \\
\qquad\qquad f(n)(p, \epsilon) = (\epsilon, \epsilon) \\
\qquad\qquad f(n)(p, a \,\&\, y) = (\epsilon, a \,\&\, \epsilon) \frown f(n-1)(p, y)
\end{array}
$$

It follows straightforwardly that $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$, $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$. Moreover, it is also clear that

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

If we change the type of the channel x from N_+ to N and also replace N_+ by N in the definition of $\tilde{R}_2$ then ($\ddagger$) does not hold anymore. For example, p in the definition of $\tilde{R}_2$ may consist of only 0's in which case nothing will be output along k. However, if we add the liveness constraint

$$\forall j \in \mathsf{N}_+ : \exists k \in \mathsf{N}_+ : k \geq j \wedge p(k) \neq 0,$$

to the definition of $\tilde{R}_2$ then ($\ddagger$) is valid. Unfortunately,

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

does not hold, because our definition of total refinement does not allow the liveness constraint guaranteed by $\tilde{S}_2$ to be exploited. This clearly points out the limit of our approach: synchronization protocols whose correctness depend upon that the demands (acknowledgments) sent along a channel are fairly distributed over sets of demands (acknowledgments) cannot be handled.

6 Conclusions

Since Kahn's influential paper on the modeling of deterministic data-flow networks was published in 1974 [Kah74], a number of authors have proposed formalisms for the representation 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 the 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 fix-point 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 (see Ex. 4) the proposed refinement relation is not suited for synchronization protocols whose correctness depend upon that the demands (acknowledgments) sent along a channel are fairly distributed over sets of demands (acknowledgments). Such a protocol is for example proposed in [AvT87].

However, there are several ways of generalizing total refinement. For example, let A be a formula whose free variables are contained in $\tilde{Q} \setminus Q$, we may then define $\overset{A}{\rightsquigarrow}$ to be the refinement relation characterized by

$$\forall \tilde{f} \in [\![\ \tilde{S} \ ]\!] : \exists f \in [\![\ S \ ]\!] : \forall \alpha \in \tilde{Q}^{\omega} : (\alpha/(\tilde{Q} \setminus Q)) \models A \Rightarrow \tilde{f}(\alpha)/O = f(\alpha).$$

This refinement relation seems to be sufficiently general, 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].

7 Acknowledgments

The author has benefited from discussions with Manfred Broy and Bernhard Schätz. Pierre Collette read an earlier version of this paper and provided valuable comments. The recommendations of the referees led to several improvements.

References

[AL88] M. Abadi and L. Lamport. The existence of refinement mappings. Technical Report 29, Digital, SRC, Palo Alto, 1988.

[AL90] M. Abadi and L. Lamport. Composing specifications. Technical Report 66, Digital, SRC, Palo Alto, 1990.

[AS85] B. Alpern and F. B. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, 1985.

[AvT87] J. K. Annot and R. A. H. van Twist. A novel deadlock free and starvation free packet switching communication processor. In *Proc. PARLE'87, 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 *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, SRC, Palo Alto, 1992.

[Bro93] M. Broy. (Inter-) Action refinement: The easy way. In *Proc. Program Design Calculi, Summerschool, Marktoberdorf*, pages 121–158. Springer, 1993.

[BS94] M. Broy and K. Stølen. Specification and refinement of finite dataflow networks — a relational approach. In *Proc. FTRTFT'94, Lecture Notes in Computer Science 863*, pages 247–267, 1994.

[HJH90] J. He, M. Josephs, and C. A. R Hoare. A theory of synchrony and asynchrony. In *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 *Proc. Information Processing 74*, pages 471–475. North-Holland, 1974.

[Kel78] R. M. Keller. Denotational models for parallel programs with indeterminate operators. In *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 *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. To appear in Formal Aspects of Computing.

[Stø91] K. Stølen. A method for the development of totally correct shared-state parallel programs. In *Proc. CONCUR'91, Lecture Notes in Computer Science 527*, pages 510–525, 1991.

This article was processed using the LaTeX macro package with LLNCS style