TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication

Ketil Stølen

TUM-I9537
SFB-Bericht Nr.342/20/95 A
November 1995

TUM–INFO–11-95-I37-100/1.–FI

Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©1995 SFB 342 Methoden und Werkzeuge für
die Nutzung paralleler Architekturen

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication

Ketil Stølen
Institut für Informatik, TU München, D-80290 München, Germany
email:stoelen@informatik.tu-muenchen.de

Abstract

We present three refinement principles supporting the transition from system specifications based on (unbounded) asynchronous communication to system specifications based on (bounded) synchronous communication. We refer to these principles as partial, total and conditional refinement, respectively. We distinguish between two synchronization techniques, namely synchronization by hand-shake and synchronization by real-time constraints. Partial refinement supports synchronization by hand-shake with respect to safety properties. Total refinement supports synchronization by hand-shake with respect to both safety and liveness properties. Finally, conditional refinement supports both synchronization by hand-shake and by real-time constraints. We discuss, relate and show the use of these principles in a number of small examples.

1 Introduction

Any method for system development, which depends on that boundedness constraints — constraints imposing upper bounds on the memory available for some data structure, component or channel — are imposed already in the requirement specification, is not a very useful method from a practical point of view.
Firstly, such boundedness constraints may have a very complicating effect and thereby lead to a reduced understanding of the system to be developed. Boundedness constraints also complicate formal reasoning and design. Thus, it seems sensible to avoid imposing these constraints as long as possible — in other words, to impose these boundedness constraints only in the later phases of a system development.
Secondly, the exact nature of these constraints is often not known when the requirement specification is written. For example, in the requirement engineering phase of a system development, it is often not clear in what programming language(s) the system is to be implemented or on what sort of architecture the system is supposed to run. Thus, in that case, it is known that some boundedness constraints are to be imposed, but not exactly what these are.
On the other hand, since any computer system has only a bounded amount of memory, it is clear that at some point in a system development such boundedness constraints have to be imposed. Thus, in a system development it must be possible to

move from system specifications based on unbounded resources to system specifications based on bounded resources. Unfortunately, the usual principles of behavioral and interface refinement do not always support this type of refinements.
In this paper we concentrate on a particular aspect of this problem, namely the transition from system specifications based on (unbounded) asynchronous communication to system specifications based on (bounded) synchronous communication. We distinguish between two synchronization techniques, namely synchronization by *hand-shake* and synchronization by *real-time constraints*. By synchronization by hand-shake we mean all sorts of time independent, demand driven or acknowledgment based synchronization.
We propose three refinement principles, namely *partial*, *total* and *conditional* refinement. Partial and total refinement support synchronization by hand-shake. Partial refinement is restricted to specifications which only impose safety properties. Total refinement preserves both safety and liveness properties, but is not as general as we would have liked. Conditional refinement supports both synchronization by hand-shake and by real-time constraints.
The rest of this paper is split into five sections. In Section 2 we introduce the underlying semantics. In Section 3 we explain what we mean by a specification, and we define the usual principle of behavioral refinement. In Section 4 we introduce the three refinement principles, namely partial, total and conditional refinement, and show how they can be used to support synchronization by hand-shake. In Section 5 we show how conditional refinement can be used to support synchronization by real-time constraints. Finally, there is a conclusion giving a brief summary and a comparison to approaches known from the literature.

2 Semantic Model

We represent the communication histories of channels by *timed streams*. A timed stream is a finite or infinite sequence of messages and time ticks. A time tick is represented by $\surd$. The interval between two consecutive ticks represents the least unit of time. A tick occurs in a stream at the end of each time unit.
An infinite timed stream represents a complete communication history; a finite timed stream represents a partial communication history. Since time never halts, any infinite timed stream is required to have infinitely many ticks. We do not want timed streams to end in the middle of a time unit. Thus, we insist that a timed stream is either empty, infinite or ends with a tick.
Given a set of messages M, by M^∞, M^* and M^ω we denote respectively the set of all *infinite* timed streams over M, the set of all *finite* timed streams over M, and the set of all *finite* and *infinite* timed streams over M. We use N to denote the set of natural numbers, and $\mathsf{N}_{+\infty}$ to denote $\mathsf{N} \cup \{\infty\}$. Given $s \in M^\omega$ and $j \in \mathsf{N}_{+\infty}$, $s{\downarrow_j}$ denotes the prefix of s characterizing the behavior until time j, i.e., $s{\downarrow_j}$ denotes s if j is greater than the number of ticks in s, and the shortest prefix of s containing j ticks, otherwise. Note that $s{\downarrow_\infty} = s$. This operator is overloaded to tuples of timed streams in a point-wise style, i.e., $t{\downarrow_j}$ denotes the tuple we get by applying $\downarrow_j$ to each component of t.
A *named stream tuple* is a mapping $\alpha \in a$

timed streams. Intuitively, α assigns a (possibly partial) communication history to each channel named by the channel identifiers in a. The operator $\downarrow$ is overloaded to named stream tuples in the same point-wise style as for tuples of timed streams. Given two named stream tuples $\alpha \in a$

where $i_1 \cap i_2 = o_1 \cap o_2 = i_1 \cap o_1 = i_2 \cap o_2 = \emptyset$, is characterized by $\tau_1 \otimes \tau_2$. Informally speaking, any output channel of τ_1 and input channel of τ_2, and any output channel of τ_2 and input channel of τ_1, whose names are *identical*, are connected and hidden in the sense that they cannot be observed from the outside.
Given that

$$i = (i_1 \setminus o_2) \cup (i_2 \setminus o_1), \qquad o = (o_1 \setminus i_2) \cup (o_2 \setminus i_1),$$

for any $\alpha \in i$

- $r|_j$ denotes $\langle\rangle$ if $j \leq 0$, the prefix of r of length j if $0 < j < \#r$, and r otherwise. We define $r|_\infty = r$. This operator is overloaded to stream tuples in a point-wise way. Note the way $\downarrow$ differs from this operator.

- $r \frown s$ denotes the result of *concatenating* r and s. Thus, $\langle a, b\rangle \frown \langle c, d\rangle = \langle a, b, c, d\rangle$. If r is infinite we have that $r \frown s = r$.

- $\overline{r}$ denotes the result of *removing* all ticks in r. Thus, $\overline{\langle a, \surd, b, \surd\rangle} = \langle a, b\rangle$.

3.2 Specification Formats

We write *time dependent* specifications in the following form

$$S \equiv (i \rhd o) \overset{\mathsf{td}}{::} R$$

S is the specification's name, and i and o are finite, repetition free lists of identifiers. The identifiers in i name the input channels, and the identifiers in o name the output channels. The lists are not allowed to have identifiers in common. We refer to the elements of these lists as the *input* and *output* identifiers, respectively. The label td is used to distinguish time dependent specifications from time independent specifications. As we will see below, the latter are labeled by ti. R is a formula in predicate logic with the identifiers of i and o as its only free variables. In R each of these identifiers represents a timed infinite stream modeling the complete communication history of the channel named by the identifier. Thus, i and o name the input and output channels, respectively, and R characterizes the relationship between their communication histories. We will often refer to R as the *i/o-relation* and to $(i \rhd o)$ as the *syntactic interface*.
For any mapping $\alpha \in C$

The *composition* operator $\otimes$ can be lifted from pulse-driven functions to specifications in a straightforward way. Let S_1 and S_2 be two specifications whose syntactic interfaces are characterized by $(i_1 \rhd o_1)$ and $(i_2 \rhd o_2)$, respectively. If $i_1 \cap i_2 = o_1 \cap o_2 = \emptyset$, $i = (i_1 \setminus o_2) \cup (i_2 \setminus o_1)$ and $o = (o_1 \setminus i_2) \cup (o_2 \setminus i_1)$, then $S_1 \otimes S_2$ denotes the network pictured in Figure 1 with τ_1 and τ_2 replaced by S_1 and S_2, respectively. We define $[\![S_1 \otimes S_2]\!]$ to be the set of all

$\tau \in (i$

4 Synchronization by Hand-Shake

As already mentioned, in this paper we consider two synchronization techniques, namely synchronization by *hand-shake* and synchronization by *real-time constraints*. In this section we propose refinement principles supporting the former.
The close relationship between specification formalisms based on hand-shake communication and purely asynchronous communication is well-documented in the literature. For example, [6] shows how the process algebra of CSP can be extended to handle asynchronous communication by representing each asynchronous communication channel by a separate process. A similar technique allows different types of hand-shake communication to be introduced in a system specification based on purely asynchronous communication: 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.
Consider a network consisting of two time independent specifications S_1 and S_2 communicating *purely asynchronously* via an internal channel y, as indicated by Network 1 of Figure 2. We want to refine Network 1 into a network of two specifications $\tilde{S}_1$ and $\tilde{S}_2$ communicating in a *synchronous* manner employing some sort of *hand-shake* protocol — in other words, into a network of the same form as Network 4 of Figure 2.
Using the technique proposed above, we may move from Network 1 to Network 4 in three steps, employing the usual principle of behavioral refinement:

- Step 1: Insert an identity specification I between S_1 and S_2 of Network 1, as indicated by Network 2 of Figure 2. It follows trivially that

  $$S_1 \otimes S_2 \rightsquigarrow S_1 \otimes I \otimes S_2.$$

  Thus, Network 2 is a behavioral refinement of Network 1.

- 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 3 of Figure 2. Clearly, it must be shown that

  $$I \rightsquigarrow I_1 \otimes I_2,$$

  in which case it follows by transitivity and congruence of $\rightsquigarrow$ that Network 3 is a behavioral refinement of Network 1.

- Step 3: Finally, if we can show that

  $$S_1 \otimes I_1 \rightsquigarrow \tilde{S}_1, \qquad I_2 \otimes S_2 \rightsquigarrow \tilde{S}_2,$$

  we have that

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

  by transitivity and congruence of $\rightsquigarrow$. Thus, in that case, Network 4 is a behavioral refinement of Network 1.

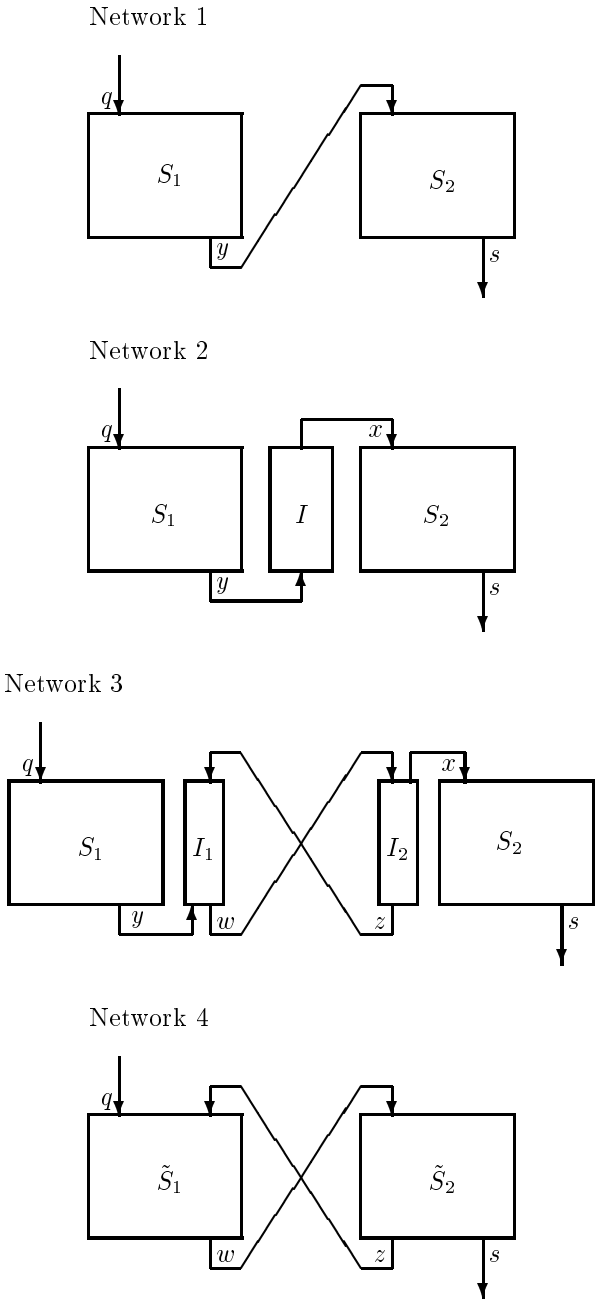

Figure 2: Introducing Synchronization by Hand-Shake

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 principle than behavioral refinement is needed. This refinement principle must allow for the introduction of additional feedback loops. For example, without this generality it is not possible to synchronize the communication between S_1 and S_2 in Network 1, using a hand-shake protocol. Of course, one may argue that the synchronization could be conducted via the environment, but this is not what we want. Thus, with respect to our example, this generality is needed in order to build up a connection from S_2 to S_1 allowing S_2 to communicate acknowledgments or demands.

4.1 Partial Refinement

Consider two time independent specifications S and $\tilde{S}$ such that $(i \rhd o)$ is the syntactic interface of S. In the previous section we have seen that a refinement principle supporting synchronization by hand-shake must allow for the introduction of additional feedback loops. This implies that if $\tilde{S}$ is a refinement of S in this sense, $\tilde{S}$ must be allowed to have additional input and output channels. Thus, given that $(\tilde{\imath} \rhd \tilde{o})$ is the syntactic interface of $\tilde{S}$, we assume that $i \subseteq \tilde{\imath}$ and $o \subseteq \tilde{o}$.
We now 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 any function $\tilde{\tau}$ satisfying $\tilde{S}$ and any input history there is a function τ satisfying S which behaves in the same way as $\tilde{\tau}$ with respect to this input history. However, due to the synchronization conducted via the new channels, the computation of $\tilde{\tau}$ can be halted too early because the required acknowledgments or demands are not received. Thus, in the general case, unless we make certain assumptions about the environment's behavior, this requirement is too strong. On the other hand, since a safety property only says something about what a component is not allowed to do, and nothing about what it has to do, the possibility that the computation of $\tilde{\tau}$ is halted too early is not a problem if S is safe. Thus, the proposed definition is adequate if we are only interested in safety properties. Formally, given that S and $\tilde{S}$ are safe, we say that $\tilde{S}$ is a *partial refinement* of S, written $S \overset{p}{\rightsquigarrow} \tilde{S}$, iff

$\forall \tilde{\tau} \in [\![\ \tilde{S}\ ]\!], \alpha \in \tilde{\imath}$

4.2 Total Refinement

Consider once more the two time independent specifications of the previous section. As already argued, since the computation of a component satisfying $\tilde{S}$ can be halted too early because a required acknowledgment or demand is not received, the definition of partial refinement is too strong if S also imposes liveness properties. In that case, the relation has to be weakened by some sort of environment assumption — an assumption constraining the communication histories of the "new" input channels. Let $\alpha \in \tilde{\imath}$

along k. Since both $S_1 \otimes S_2$ and $\tilde{S}_1 \otimes \tilde{S}_2$ have q as their only input channel, and since it may be the case that $\#q > 1$, it follows that

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

□

The problem observed in Example 1 can be understood as *deadlock* caused by an erroneous synchronization protocol. What is required is some proof obligation, more explicitly — some freedom from deadlock test, characterizing under what conditions total refinement is a "congruence" with respect to $\otimes$. Firstly, we want a proof obligation which takes advantage of the fact that we have already proved that $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$ and $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$. This suggests it should be independent of S_1 and S_2. Secondly, 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, it should not require knowledge about how $\tilde{S}_1$ and $\tilde{S}_2$ are implemented.
With respect to a network as in Example 1, it is enough to check that, when the computation halts, then the output along z will not be extended if additional input is received on the feedback channel x. This is equivalent to verifying the proof obligation below

$$R_{\tilde{S}_1} \wedge R_{\tilde{S}_2} \Rightarrow R_{\tilde{S}_1}[{}^{x}_{x^\frown x'}],$$

where x' is a new identifier. We now prove that this proof obligation guarantees $S_1 \otimes S_2 \overset{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2$, given that $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$, $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$ and that the syntactic interfaces are as in Example 1.
Let $\tilde{\tau} \in [\![\ \tilde{S}_1 \otimes \tilde{S}_2\ ]\!]$, $\alpha \in \{q\}$

Then $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$ and $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$ imply there are $\tau_1 \in [\![S_1]\!]$, $\tau_2 \in [\![S_2]\!]$ such that $\tau_1(\alpha) = \delta$ and $\tau_2(\delta) = \beta$.

Since $\tau_1 \in [\![S_1]\!]$, $\tau_2 \in [\![S_2]\!]$ imply $\tau_1 \otimes \tau_2 \in [\![S_1 \otimes S_2]\!]$, it then follows that $S_1 \otimes S_2 \overset{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2$, which is what we wanted to prove.
With respect to Example 1, if the i/o-relation of $\tilde{S}_2$ is redefined as below

$$R_{\tilde{S}_2} \overset{\text{def}}{=} k = z \wedge x = z,$$

it follows by the proposed proof obligation that $S_1 \otimes S_2 \overset{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2$.
We now show how this proof obligation can be generalized to handle arbitrary composition modulo $\otimes$.

Example 2 Handling additional external input channels:
To indicate the weakness of the test we have already formulated, we go through another example. Let S_1, S_2 and $\tilde{S}_1$ be as in Example 1, and let

$$\tilde{S}_2 \equiv (z, i \rhd x, k) \overset{\text{ti}}{::} k = z \wedge x = z|_{\#i}.$$

The behavior of $\tilde{S}_2$ now depends on an additional input channel i. As before, $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$ and $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$. Moreover, it is also clear that $S_1 \otimes S_2 \overset{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2$. Unfortunately, our proof obligation does not hold. For example, we have that

$$\#q > 1 \wedge i = \langle\rangle \wedge \#z = 1 \Rightarrow R_{\tilde{S}_1} \wedge R_{\tilde{S}_2}, \quad R_{\tilde{S}_1}[{}^{x}_{x^\frown x'}] \wedge \#x' = \infty \Rightarrow z = q.$$

Thus, since x' does not occur in the antecedent of the proof obligation, it follows that it is falsified by at least this instantiation. □

With respect to Example 2, the problem is that our proof obligation does not take the new channel i into account. Since i is not an output channel of $\tilde{S}_1$, but connected to the overall environment, the implicit environment assumption built into the definition of total refinement implies we only have to consider the situation that infinitely many messages are received on i. Thus, the proof obligation can be weakened as below:

$$\forall i \in M^{\overline{\infty}} : R_{\tilde{S}_1} \wedge R_{\tilde{S}_2} \Rightarrow R_{\tilde{S}_1}[{}^{x}_{x^\frown x'}].$$

Note that this proof obligation is satisfied by the refinement step considered in Example 2.
It is now straightforward to formulate a general proof obligation. Let x, y, i be lists consisting of respectively the new input channels of $\tilde{S}_1$ connected to $\tilde{S}_2$, the new input channels of $\tilde{S}_2$ connected to $\tilde{S}_1$, and the new input channels of $\tilde{S}_1$ and $\tilde{S}_2$ connected to the overall environment. Then we get the following refinement rule

$$\frac{\begin{array}{l} S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1 \\ S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2 \\ \forall i \in M^{\overline{\infty}} : R_{\tilde{S}_1} \wedge R_{\tilde{S}_2} \Rightarrow R_{\tilde{S}_1}[{}^{x}_{x^\frown x'}] \wedge R_{\tilde{S}_2}[{}^{y}_{y^\frown y'}] \end{array}}{S_1 \otimes S_2 \overset{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2}$$

It is assumed that the specifications are basic. The rule can easily be generalized to deal with $n > 2$ specifications.

The proof that this rule is sound is a straightforward generalization of the proof for the restricted case given above. See appendix for details. Note that this rule does not require proof work conducted earlier in the development process to be redone. The two first premises can be checked locally; the third premise is a co-existence check making sure that no deadlock has been introduced.
As already mentioned, although total refinement is sufficient for many hand-shake protocols, this principle is not as general as we would have liked. The problem is that certain synchronization protocols impose *fairness* constraints on the distribution of acknowledgments or demands sent along a channel.

Example 3 Lack of generality:
To see the lack of generality, let S_1 and S_2 be defined as in Example 1. Moreover, assume that $\tilde{S}_1$ and $\tilde{S}_2$ have the same syntactic interfaces as in Example 1, and that their i/o-relations are redefined as below

$$R_{\tilde{S}_1} \stackrel{\mathsf{def}}{=} z = q|_{\#(1©x)+1}, \quad R_{\tilde{S}_2} \stackrel{\mathsf{def}}{=} k = z \wedge \#(1©x) = \#z.$$

Clearly, $S_2 \stackrel{t}{\rightsquigarrow} \tilde{S}_2$. Moreover, we also have that $S_1 \otimes S_2 \stackrel{t}{\rightsquigarrow} \tilde{S}_1 \otimes \tilde{S}_2$. However, it does not hold that $S_1 \stackrel{t}{\rightsquigarrow} \tilde{S}_1$. The reason is of course that the implicit environment assumption of total refinement, namely that infinitely many messages are received on x, does not guarantee that the required number of 1's are received. □

4.3 Conditional Refinement

Consider once more the two time independent specifications S and $\tilde{S}$ of the two previous sections. As already argued, for certain hand-shake protocols the implicit environment assumption of total refinement is too weak. One way to deal with this problem is to make the environment assumption explicit and let the user himself specify the required assumption. More explicitly, let B be a formula whose free variables are contained in $\tilde{\imath} \cup \tilde{o}$ and vary over untimed streams, we say that $\tilde{S}$ is a *conditional refinement* of S with respect to B, written $S \rightsquigarrow_B \tilde{S}$, iff

$$\forall \tilde{\tau} \in [\![\ \tilde{S}\ ]\!], \alpha \in \tilde{\imath}$$

It is assumed that the specifications are basic. The rule can easily be generalized to deal with $n > 2$ specifications.
Also this rule has the nice property that proof work conducted earlier in the development process does not have to be redone. The two first premises are local constraints; the third is a co-existence check making sure that no deadlock has been introduced.
We now prove that the rule is sound. Let $\tilde{\imath}_1, \tilde{\imath}_2, \tilde{o}_1, \tilde{o}_2, \tilde{x}, \tilde{y}$ be mutually disjoint lists of identifiers, and let $\cdot$ be a concatenation operator for such lists. Moreover, let $(i_1 \cdot x \rhd o_1 \cdot y)$, $(y \cdot i_2 \rhd x \cdot o_2)$, $(\tilde{\imath}_1 \cdot \tilde{x} \rhd \tilde{o}_1 \cdot \tilde{y})$ and $(\tilde{y} \cdot \tilde{\imath}_2 \rhd \tilde{x} \cdot \tilde{o}_2)$ be the syntactic interfaces of S_1, S_2, $\tilde{S}_1$ and $\tilde{S}_2$, respectively. Assume that $i_1 \subseteq \tilde{\imath}_1$, $o_1 \subseteq \tilde{o}_1$, $i_2 \subseteq \tilde{\imath}_2$, $o_2 \subseteq \tilde{o}_2$, $x \subseteq \tilde{x}$, $y \subseteq \tilde{y}$, and that the three premises hold.
Let $\tilde{\tau} \in [\![\tilde{S}_1 \otimes \tilde{S}_2]\!]$, $\alpha_1 \in \tilde{\imath}_1$

capture real-time in the sense that the interval between each pair of consecutive time ticks represents the same least unit of time. Consider two time dependent specifications S and $\tilde{S}$. For simplicity, since we in this section do not consider synchronization by hand-shake, we assume that both specifications have the same syntactic interface $(i \rhd o)$. Let B be a formula whose free variables are contained in $i \cup o$ and vary over infinite timed streams. We say $\tilde{S}$ is a *conditional refinement* of S with respect to B, iff

$\forall \tilde{\tau} \in [\![\tilde{S}]\!], \alpha \in i$

hand-shake synchronization, one may introduce a well-formedness condition making sure that the condition B only constrains what is received on the "new" feedback channels. However, a detailed discussion of well-formedness conditions is beyond the scope of this paper.

6 Conclusions

In this paper we have introduced three principles of refinement. Their properties can be summed up as below. Partial refinement supports synchronization by hand-shake with respect to safety properties and is a congruence modulo $\otimes$, but does not support synchronization by real-time constraints. Total refinement supports synchronization by hand-shake with respect to both safety and liveness properties and allows modular top-down design, but is not very general and does not support synchronization by real-time constraints. Conditional refinement supports both synchronization by hand-shake and by real-time constraints with respect to both safety and liveness properties and allows modular top-down design. As we see it, the main contribution of this paper is that we have shown how refinement principles based on explicit or implicit environment assumptions can be used to support the transition from system specifications based on purely asynchronous communication to system specifications based on synchronous communication. However, in particular conditional refinement seems to have a much broader application area. See appendix for detailed proofs of the different claims made in this paper.

As explained in [4], behavioral refinement can be generalized to interface refinement by relating the concrete and abstract interface by a representation function in the style of [7]. The three refinement principles proposed above can be generalized accordingly.

We refer to [5] for a detailed investigation of the underlying semantic model.

The principles of partial and total refinement were defined in [10], but in a less general setting. Conditional refinement is a straightforward generalization of behavioral refinement — so straightforward that it seems unlikely that this idea is new. For example, what [1] refers to as conditional implementation is closely related. Moreover, the decomposition theorem of [1] seems to allow related refinements with respect to complete systems. Contrary to us, their co-existence proof is formulated with respect to the more abstract specifications. An attempt to tackle the transition from unbounded to bounded resources in the context of algebraic specifications can be found in [3].

With respect to conditional refinement, instead of using explicit conditions one may calculate the weakest conditions under which the concrete specifications refine the abstract specifications. However, we find the use of explicit conditions more practical.

The refinement principles proposed above can of course be reformulated in other settings. For example, if the refinement principle of the rely/guarantee method [8] is weakened along the lines proposed in this paper some of the problems reported in [12] seem to disappear.

The proposed refinement principles have not been justified with respect to some sort of observation language as for example advocated in [9]. Instead, the well-suitedness of behavioral refinement as defined in [4] has been taken for granted. Both total and

conditional refinement characterize behavioral refinement in the sense of [4] modulo certain assumptions about the environment.
In practice, specifications are often written in an assumption/commitment form. Some of the proof-obligations proposed above can then be replaced by more sophisticated rules. See [11] for assumption/commitment rules with respect to the semantic setting of this paper.

7 Acknowledgments

The author has benefited from many discussions with Manfred Broy on this and related topics. Pierre Collette, Bernhard Möller and Oscar Slotosch have read erlier drafts of this paper and provided many helpful comments. Financial support has been received from the Sonderforschungsbereich 342 "Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen".

References

[1] M. Abadi and L. Lamport. Conjoining specifications. Technical Report 118, Digital, SRC, Palo Alto, 1993.

[2] P. America, J. de Bakker, J. N. Kok, and J. Rutten. Denotational semantics of a parallel object-oriented language. *Information and Computation*, 83:152–205, 1989.

[3] M. Breu. *Endliche Implementierung algebraischer Spezifikationen.* PhD thesis, Technische Universität München, 1991. Also available as Report TUM-I9111, Technische Universität München.

[4] M. Broy. Compositional refinement of interactive systems. Technical Report 89, Digital, SRC, Palo Alto, 1992.

[5] R. Grosu and K. Stølen. A denotational model for mobile point-to-point dataflow networks. Technical Report SFB 342/14/95 A, Technische Universität München, 1995.

[6] 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.

[7] C. A. R. Hoare. Proof of correctness of data representations. *Acta Informatica*, 1:271–282, 1972.

[8] C. B. Jones. Specification and design of (parallel) programs. In *Proc. Information Processing 83*, pages 321–331. North-Holland, 1983.

[9] T Nipkow. Non-deterministic data types: Models and implementations. *Acta Informatica*, 22:629–661, 1986.

[10] K. Stølen. A refinement relation supporting the transition from unbounded to bounded communication buffers. In *Proc. MPC'95, Lecture Notes in Computer Science 947*, pages 423–451, 1995.

[11] K. Stølen. Assumption/commitment rules for data-flow networks — with an emphasis on completeness. 1996. To appear in Proc. ESOP'96.

[12] J. C. P. Woodcock and B. Dickinson. Using VDM with rely and guarantee-conditions. Experiences from a real project. In *Proc. VDM'88, Lecture Notes in Computer Science 328*, pages 434–458, 1988.

A Proofs

We use $S \in (i \rhd o)$ to say that $(i \rhd o)$ is the syntactic interface of the specification S. Moreover, we employ $\langle i \rangle$ as a short-hand for $(i$

(1) : $S_1 \overset{p}{\rightsquigarrow} \tilde{S}_1$,
(2) : $S_2 \overset{p}{\rightsquigarrow} \tilde{S}_2$,
(3) : $i_1 \subseteq \tilde{\imath}_1,\ o_1 \subseteq \tilde{o}_1,\ i_2 \subseteq \tilde{\imath}_2,\ o_2 \subseteq \tilde{o}_2,\ x \subseteq \tilde{x},\ y \subseteq \tilde{y}$.

It then holds that

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

Proof: Let

(5) : $\tilde{\tau} \in [\![\ \tilde{S}_1 \otimes \tilde{S}_2\ ]\!]$,
(6) : $\alpha_1 \in \langle \tilde{\imath}_1 \rangle,\ \alpha_2 \in \langle \tilde{\imath}_2 \rangle$.

(5), (6) imply there are $\tilde{\tau}_1, \tilde{\tau}_2$ such that

(7) : $\tilde{\tau}_1 \in [\![\ \tilde{S}_1\ ]\!]$,
(8) : $\tilde{\tau}_2 \in [\![\ \tilde{S}_2\ ]\!]$,
(9) : $\tilde{\tau}(\alpha_1 \uplus \alpha_2) = (\tilde{\tau}_1 \otimes \tilde{\tau}_2)(\alpha_1 \uplus \alpha_2)$.

(9) implies there are β_1, β_2, δ, σ such that

(10) : $\beta_1 \in \langle \tilde{o}_1 \rangle,\ \beta_2 \in \langle \tilde{o}_2 \rangle,\ \delta \in \langle \tilde{x} \rangle,\ \sigma \in \langle \tilde{y} \rangle$,
(11) : $\tilde{\tau}_1(\alpha_1 \uplus \delta) = (\beta_1 \uplus \sigma)$,
(12) : $\tilde{\tau}_2(\sigma \uplus \alpha_2) = (\delta \uplus \beta_2)$,
(13) : $(\tilde{\tau}_1 \otimes \tilde{\tau}_2)(\alpha_1 \uplus \alpha_2) = (\beta_1 \uplus \beta_2)$.

(1), (2), (3), (6), (7), (8), (10), (11), (12) imply there are τ_1 and τ_2 such that

(14) : $\tau_1 \in [\![\ S_1\ ]\!]$,
(15) : $\tau_2 \in [\![\ S_2\ ]\!]$,
(16) : $\tau_1 \otimes \tau_2 \in [\![\ S_1 \otimes S_2\ ]\!]$,
(17) : $\tilde{\tau}_1(\alpha_1 \uplus \delta)|_{o_1 \cup y} = \beta_1|_{o_1} \uplus \sigma|_y = \tau_1(\alpha_1|_{i_1} \uplus \delta|_x)$,
(18) : $\tilde{\tau}_2(\sigma \uplus \alpha_2)|_{x \cup o_2} = \delta|_x \uplus \beta_2|_{o_2} = \tau_2(\sigma|_y \uplus \alpha_2|_{i_2})$.

(13), (17), (18) imply

(19) : $(\tilde{\tau}_1 \otimes \tilde{\tau}_2)(\alpha_1 \uplus \alpha_2)|_{o_1 \cup o_2} = \beta_1|_{o_1} \uplus \beta_2|_{o_2} = (\tau_1 \otimes \tau_2)(\alpha_1|_{i_1} \uplus \alpha_2|_{i_2})$.

The way (19) was deduced from (5), (6) implies (4).

A.2 Total Refinement

Proposition 3 (total refinement implies partial refinement) *Given* $S \in (i \rhd o)$ *and* $\tilde{S} \in (\tilde{\imath} \rhd \tilde{o})$ *such that*

(1) : $S \overset{t}{\rightsquigarrow} \tilde{S}$,
(2) : $i \subseteq \tilde{\imath}, o \subseteq \tilde{o}$,
(3) : $S, \tilde{S}$ *are safe and time independent.*

It then holds that

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

Proof: Let

(5) : $\hat{\imath} = \tilde{\imath} \setminus i$.

Assume

(6) : $S \overset{p}{\not\rightsquigarrow} \tilde{S}$.

We prove that this assumption leads to a contradiction. (6) implies that there are $\tilde{\tau}, \alpha$ such that

(7) : $\tilde{\tau} \in [\![\tilde{S}]\!]$,
(8) : $\alpha \in \langle \tilde{\imath} \rangle$,
(9) : $\forall \tau \in [\![S]\!] : \tilde{\tau}(\alpha)|_o \neq \tau(\alpha|_i)$.

(3), (7), (9) imply that there is a j such that

(10) : $0 < j < \infty$,
(11) : $\forall \tau \in [\![S]\!] : (\tilde{\tau}(\alpha)|_o)\downarrow_j \neq \tau(\alpha|_i)\downarrow_j$.

Let α' be such that

(12) : $\alpha' \in \langle \tilde{\imath} \rangle$,
(13) : $\alpha'\downarrow_{j-1} = \alpha\downarrow_{j-1}$,
(14) : $\forall c \in \hat{\imath} : \overline{\alpha'(c)} = \infty$.

(1), (7), (14) imply there is a τ such that

(15) : $\tau \in [\![S]\!]$,
(16) : $\tilde{\tau}(\alpha')|_o = \tau(\alpha'|_i)$.

(13), (16) and the pulse-drivenness of $\tau, \tilde{\tau}$ imply

(17) : $(\tilde{\tau}(\alpha)|_o)\downarrow_j = (\tilde{\tau}(\alpha')|_o)\downarrow_j = (\tau(\alpha'|_i))\downarrow_j = (\tau(\alpha|_i))\downarrow_j$.

(17) contradicts (11).

Proposition 4 (transitivity) *Given time independent specifications* $S_1 \in (i_1 \rhd o_1), S_2 \in (i_2 \rhd o_2)$ *and* $S_3 \in (i_3 \rhd o_3)$ *such that*

(1) : $S_1 \overset{t}{\rightsquigarrow} S_2$,
(2) : $S_2 \overset{t}{\rightsquigarrow} S_3$,
(3) : $i_1 \subseteq i_2 \subseteq i_3$,
(4) : $o_1 \subseteq o_2 \subseteq o_3$.

It then holds that

(5) : $S_1 \overset{t}{\rightsquigarrow} S_3$.

Proof: Let

(6) : $\tau_3 \in [\![S_3]\!]$,
(7) : $\sigma \in \langle i_3 \rangle$,
(8) : $\forall c \in i_3 \setminus i_1 : \#\alpha(c) = \infty$.

(3), (8) imply

(9) : $\forall c \in i_3 \setminus i_2 : \#\alpha(c) = \infty$.

(2), (3), (4), (6), (7), (9) imply there is a τ_2 such that

(10) : $\tau_2 \in [\![\ S_2\ ]\!]$,
(11) : $\tau_3(\alpha)|_{o_2} = \tau_2(\alpha|_{i_2})$.

(3), (8) imply

(12) : $\forall c \in i_2 \setminus i_1 : \#(\alpha|_{i_2})(c) = \infty$.

(1), (3), (4), (7), (10), (11), (12) imply there is a τ_1 such that

(13) : $\tau_1 \in [\![\ S_1\ ]\!]$,
(14) : $\tau_3(\alpha)|_{o_2}|_{o_1} = \tau_2(\alpha|_{i_2})|_{o_1} = \tau_1(\alpha|_{i_2}|_{i_1})$.

(3), (4), (14) imply

(15) : $\tau_3(\alpha)|_{o_1} = \tau_1(\alpha|_{i_1})$.

The way (15) was deduced from (6), (7), (8) implies (5).

Lemma 1 *Given two time independent specifications $S \in (i \rhd o)$, $\tilde{S} \in (\tilde{\imath} \cdot \tilde{r} \rhd \tilde{o})$, a function $\tilde{\tau}$, two named stream tuples $\alpha \in \langle \tilde{\imath} \rangle$, $\beta \in \langle \tilde{r} \rangle$ and $j \in \mathsf{N}$ such that*

(1) : $i \subseteq \tilde{\imath}, \quad o \subseteq \tilde{o}, \quad \tilde{\imath} \cap \tilde{r} = \emptyset$,
(2) : $S \overset{t}{\rightsquigarrow} \tilde{S}$,
(3) : $\tilde{\tau} \in [\![\ \tilde{S}\ ]\!]$,
(4) : $\forall c \in (\tilde{\imath} \setminus i) : \#\overline{\alpha(c)} = \infty$,
(5) : $\forall c \in \tilde{r} : \beta(c) = \beta(c)\downarrow_j \frown \surd^\infty$,
(6) : $\forall \sigma \in \langle \tilde{r} \rangle : \overline{(\alpha \uplus (\beta\downarrow_j \frown \sigma) \uplus \tilde{\tau}(\alpha \uplus \beta))} \models R_{\tilde{S}}$.

Then there is a function τ such that

(7) : $\tau \in [\![\ S\ ]\!]$,
(8) : $\tau(\alpha|_i) = \tilde{\tau}(\alpha \uplus \beta)|_o$.

Proof: Let $\hat{\tau}$ be a function such that

(9) : $(\alpha \uplus \beta)\downarrow_j \not\sqsubseteq w \Rightarrow \hat{\tau}(w) = \tilde{\tau}(w)$,
(10) : $(\alpha \uplus \beta\downarrow_j) \sqsubseteq w \Rightarrow \hat{\tau}(w) = \tilde{\tau}(\alpha \uplus \beta)$,
(11) : $\alpha\downarrow_k \sqsubseteq v \wedge \alpha\downarrow_{(k+1)} \not\sqsubseteq v \wedge k \geq j \Rightarrow \hat{\tau}(v \uplus (\beta\downarrow_j \frown u)) = \tilde{\tau}(v \uplus (\beta\downarrow_k \frown u))$.

It is clear that $\hat{\tau}$ is well-defined. We want to prove that $\hat{\tau}$ is pulse-driven. Assume there are $w, w' \in \langle \tilde{\imath} \cup \tilde{r} \rangle$ and $k \in \mathsf{N}$ such that

(12) : $w\downarrow_k = w'\downarrow_k$.

It is enough to show that

(13) : $\hat{\tau}(w)\downarrow_{k+1} = \hat{\tau}(w')\downarrow_{k+1}$.

There are four cases to consider

(14) : $k \leq j$,
(15) : $k > j \wedge (\alpha \uplus \beta)\downarrow_j \not\sqsubseteq w$,
(16) : $k > j \wedge (\alpha \uplus \beta)\downarrow_j \sqsubseteq w \wedge (w|_i)\downarrow_k \sqsubseteq \alpha$,
(17) : $k > j \wedge (\alpha \uplus \beta)\downarrow_j \sqsubseteq w \wedge (w|_i)\downarrow_k \not\sqsubseteq \alpha$.

Assume (14): (13) follows from the pulse-drivenness of $\tilde{\tau}$ since (9), (10), (11) imply that $\hat{\tau}$ behaves as $\tilde{\tau}$ until time $j + 1$.

Assume (15): (13) follows from (9) and the pulse-drivenness of $\tilde{\tau}$.
Assume (16): (13) follows from (10), (11) and the pulse-drivenness of $\tilde{\tau}$.
Assume (17): (13) follows from (11) and the pulse-drivenness of $\tilde{\tau}$.
(3), (6), (9), (10), (11) imply

(18) : $\hat{\tau} \in [\![\ \tilde{S}\ ]\!]$.

Let w be such that

(19) : $(\alpha \uplus \beta{\downarrow_j}) \sqsubseteq w$,
(20) : $\forall c \in \tilde{r} : \#\overline{w(c)} = \infty$

(2), (4), (18), (20) imply there is a τ such that

(21) : $\tau \in [\![\ S\ ]\!]$,
(22) : $\tau(w|_i) = \hat{\tau}(w)|_o$.

(1), (10), (19), (22) imply

(23) : $\tau(\alpha|_i) = \tau(w|_i) = \hat{\tau}(w)|_o = \tilde{\tau}(\alpha \uplus \beta)|_o$

(23) implies (8).

Proposition 5 (congruence) *Given mutually disjoint lists of identifiers $\tilde{\imath}_1$, $\tilde{\imath}_2$, $\tilde{o}_1$, $\tilde{o}_2$, $\tilde{x}$, $\tilde{y}$ and time independent specifications $S_1 \in (i_1 \cdot x \rhd o_1 \cdot y), S_2 \in (y \cdot i_2 \rhd x \cdot o_2), \tilde{S}_1 \in (\tilde{\imath}_1 \cdot \tilde{x} \rhd \tilde{o}_1 \cdot \tilde{y})$ and $\tilde{S}_2 \in (\tilde{y} \cdot \tilde{\imath}_2 \rhd \tilde{x} \cdot \tilde{o}_2)$ such that*

(1) : $\hat{\imath}_1 = \tilde{\imath}_1 \setminus i_1,\ \hat{o}_1 = \tilde{o}_1 \setminus o_1,\ \hat{\imath}_2 = \tilde{\imath}_2 \setminus i_2,\ \hat{o}_2 = \tilde{o}_2 \setminus o_2,\ \hat{x} = \tilde{x} \setminus x,\ \hat{y} = \tilde{y} \setminus y$,
(2) : $S_1 \overset{t}{\rightsquigarrow} \tilde{S}_1$,
(3) : $S_2 \overset{t}{\rightsquigarrow} \tilde{S}_2$,
(4) : $i_1 \subseteq \tilde{\imath}_1,\ o_1 \subseteq \tilde{o}_1,\ i_2 \subseteq \tilde{\imath}_2,\ o_2 \subseteq \tilde{o}_2,\ x \subseteq \tilde{x},\ y \subseteq \tilde{y}$,
(5) : $\forall \hat{\imath}_1, \hat{\imath}_2 \in M^{\overline{\infty}} : R_{\tilde{S}_1} \wedge R_{\tilde{S}_2} \Rightarrow R_{\tilde{S}_1}[{}^{\hat{x}}_{\hat{x} \frown \hat{x}'}] \wedge R_{\tilde{S}_2}[{}^{\hat{y}}_{\hat{y} \frown \hat{y}'}]$.

It then holds that

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

Proof: Let

(7) : $\tilde{\tau} \in [\![\ \tilde{S}_1 \otimes \tilde{S}_2\ ]\!]$,
(8) : $\alpha_1 \in \langle \tilde{\imath}_1 \rangle,\ \alpha_2 \in \langle \tilde{\imath}_2 \rangle$,
(9) : $\forall c \in \hat{\imath}_1 \cup \hat{\imath}_2 : \#(\overline{\alpha_1 \uplus \alpha_2})(c) = \infty$.

(7), (8) imply there are $\tilde{\tau}_1, \tilde{\tau}_2$ such that

(10) : $\tilde{\tau}_1 \in [\![\ \tilde{S}_1\ ]\!],\ \tilde{\tau}_2 \in [\![\ \tilde{S}_2\ ]\!]$,
(11) : $\tilde{\tau}(\alpha_1 \uplus \alpha_2) = (\tilde{\tau}_1 \otimes \tilde{\tau}_2)(\alpha_1 \uplus \alpha_2)$.

(11) implies there are β_1, β_2, δ, σ such that

(12) : $\beta_1 \in \langle \tilde{o}_1 \rangle,\ \beta_2 \in \langle \tilde{o}_2 \rangle,\ \delta \in \langle \tilde{x} \rangle,\ \sigma \in \langle \tilde{y} \rangle$,
(13) : $\tilde{\tau}_1(\alpha_1 \uplus \delta) = (\beta_1 \uplus \sigma)$,
(14) : $\tilde{\tau}_2(\sigma \uplus \alpha_2) = (\delta \uplus \beta_2)$,
(15) : $(\tilde{\tau}_1 \otimes \tilde{\tau}_2)(\alpha_1 \uplus \alpha_2) = (\beta_1 \uplus \beta_2)$.

Let $\hat{x}_1, \hat{x}_2, \hat{y}_1, \hat{y}_2$ be such that

(16) : $\hat{x} = \hat{x}_1 \cup \hat{x}_2,\ \hat{y} = \hat{y}_1 \cup \hat{y}_2$,
(17) : $\forall c \in (\hat{x}_1 \cup \hat{y}_1) : \#(\overline{\delta \uplus \sigma})(c) = \infty$,
(18) : $\forall c \in (\hat{x}_2 \cup \hat{y}_2) : \#(\overline{\delta \uplus \sigma})(c) \neq \infty$.

(18) implies there is a j such that

(19) : $0 < j < \infty$,
(20) : $\forall c \in (\hat{x}_2 \cup \hat{y}_2) : (\delta \uplus \sigma)(c) = (\delta \uplus \sigma)(c)\downarrow_j \frown \sqrt{}^{\infty}$.

(10), (13), (14) imply

(21) : $(\overline{\alpha_1 \uplus \delta \uplus \beta_1 \uplus \sigma}) \models R_{\tilde{S}_1}$,
(22) : $(\overline{\sigma \uplus \alpha_2 \uplus \delta \uplus \beta_2}) \models R_{\tilde{S}_2}$.

(5), (9), (13), (21), (22), (23) imply

(23) : $(\overline{\alpha_1 \uplus \delta|_{\hat{x}_1} \uplus ((\delta|_{\hat{x}_2})\downarrow_j \frown \theta) \uplus \beta_1 \uplus \sigma}) \models R_{\tilde{S}_1}$,
(24) : $(\overline{\sigma|_{\hat{y}_1} \uplus ((\sigma|_{\hat{y}_2})\downarrow_j \frown \theta) \uplus \alpha_2 \uplus \delta \uplus \beta_2}) \models R_{\tilde{S}_2}$

(9) : $(\overline{\alpha \uplus \tau_3(\alpha)}) \models B_2$.

(2), (3), (4), (6), (7), (9) imply there is a τ_2 such that

(10) : $\tau_2 \in [\![\ S_2\ ]\!]$,
(11) : $\tau_3(\alpha)|_{o_2} = \tau_2(\alpha|_{i_2})$.

(3), (8), (11) imply

(12) : $(\overline{\alpha|_{i_2} \uplus \tau_2(\alpha|_{i_2})}) \models B_1$.

(1), (3), (4), (7), (10), (11), (12) imply there is a τ_1 such that

(13) : $\tau_1 \in [\![\ S_1\ ]\!]$,
(14) : $\tau_3(\alpha)|_{o_2}|_{o_1} = \tau_2(\alpha|_{i_2})|_{o_1} = \tau_1(\alpha|_{i_2}|_{i_1})$.

(3), (4), (14) imply

(15) : $\tau_3(\alpha)|_{o_1} = \tau_1(\alpha|_{i_1})$.

The way (15) was deduced from (6), (7), (8) implies (5).

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

342/1/90 A	Robert Gold, Walter Vogler: Quality Criteria for Partial Order Semantics of Place/Transition-Nets, Januar 1990
342/2/90 A	Reinhard Fösmeier: Die Rolle der Lastverteilung bei der numerischen Parallelprogrammierung, Februar 1990
342/3/90 A	Klaus-Jörn Lange, Peter Rossmanith: Two Results on Unambiguous Circuits, Februar 1990
342/4/90 A	Michael Griebel: Zur Lösung von Finite-Differenzen- und Finite-Element-Gleichungen mittels der Hierarchischen Transformations-Mehrgitter-Methode
342/5/90 A	Reinhold Letz, Johann Schumann, Stephan Bayerl, Wolfgang Bibel: SETHEO: A High-Performance Theorem Prover
342/6/90 A	Johann Schumann, Reinhold Letz: PARTHEO: A High Performance Parallel Theorem Prover
342/7/90 A	Johann Schumann, Norbert Trapp, Martin van der Koelen: SETHEO/PARTHEO Users Manual
342/8/90 A	Christian Suttner, Wolfgang Ertel: Using Connectionist Networks for Guiding the Search of a Theorem Prover
342/9/90 A	Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Jaroslav Kremenek, Robert Lindhof, Thomas Ludwig, Peter Luksch, Thomas Treml: TOPSYS, Tools for Parallel Systems (Artikelsammlung)
342/10/90 A	Walter Vogler: Bisimulation and Action Refinement
342/11/90 A	Jörg Desel, Javier Esparza: Reachability in Reversible Free- Choice Systems
342/12/90 A	Rob van Glabbeek, Ursula Goltz: Equivalences and Refinement
342/13/90 A	Rob van Glabbeek: The Linear Time - Branching Time Spectrum
342/14/90 A	Johannes Bauer, Thomas Bemmerl, Thomas Treml: Leistungsanalyse von verteilten Beobachtungs- und Bewertungswerkzeugen
342/15/90 A	Peter Rossmanith: The Owner Concept for PRAMs
342/16/90 A	G. Böckle, S. Trosch: A Simulator for VLIW-Architectures
342/17/90 A	P. Slavkovsky, U. Rüde: Schnellere Berechnung klassischer Matrix-Multiplikationen
342/18/90 A	Christoph Zenger: SPARSE GRIDS

Reihe A

342/19/90 A Michael Griebel, Michael Schneider, Christoph Zenger: A combination technique for the solution of sparse grid problems

342/20/90 A Michael Griebel: A Parallelizable and Vectorizable Multi- Level-Algorithm on Sparse Grids

342/21/90 A V. Diekert, E. Ochmanski, K. Reinhardt: On confluent semi-commutations-decidability and complexity results

342/22/90 A Manfred Broy, Claus Dendorfer: Functional Modelling of Operating System Structures by Timed Higher Order Stream Processing Functions

342/23/90 A Rob van Glabbeek, Ursula Goltz: A Deadlock-sensitive Congruence for Action Refinement

342/24/90 A Manfred Broy: On the Design and Verification of a Simple Distributed Spanning Tree Algorithm

342/25/90 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Peter Luksch, Roland Wismüller: TOPSYS - Tools for Parallel Systems (User's Overview and User's Manuals)

342/26/90 A Thomas Bemmerl, Arndt Bode, Thomas Ludwig, Stefan Tritscher: MMK - Multiprocessor Multitasking Kernel (User's Guide and User's Reference Manual)

342/27/90 A Wolfgang Ertel: Random Competition: A Simple, but Efficient Method for Parallelizing Inference Systems

342/28/90 A Rob van Glabbeek, Frits Vaandrager: Modular Specification of Process Algebras

342/29/90 A Rob van Glabbeek, Peter Weijland: Branching Time and Abstraction in Bisimulation Semantics

342/30/90 A Michael Griebel: Parallel Multigrid Methods on Sparse Grids

342/31/90 A Rolf Niedermeier, Peter Rossmanith: Unambiguous Simulations of Auxiliary Pushdown Automata and Circuits

342/32/90 A Inga Niepel, Peter Rossmanith: Uniform Circuits and Exclusive Read PRAMs

342/33/90 A Dr. Hermann Hellwagner: A Survey of Virtually Shared Memory Schemes

342/1/91 A Walter Vogler: Is Partial Order Semantics Necessary for Action Refinement?

342/2/91 A Manfred Broy, Frank Dederichs, Claus Dendorfer, Rainer Weber: Characterizing the Behaviour of Reactive Systems by Trace Sets

342/3/91 A Ulrich Furbach, Christian Suttner, Bertram Fronhöfer: Massively Parallel Inference Systems

342/4/91 A Rudolf Bayer: Non-deterministic Computing, Transactions and Recursive Atomicity

342/5/91 A Robert Gold: Dataflow semantics for Petri nets

342/6/91 A A. Heise; C. Dimitrovici: Transformation und Komposition von P/T-Netzen unter Erhaltung wesentlicher Eigenschaften

Reihe A

342/7/91 A Walter Vogler: Asynchronous Communication of Petri Nets and the Refinement of Transitions

342/8/91 A Walter Vogler: Generalized OM-Bisimulation

342/9/91 A Christoph Zenger, Klaus Hallatschek: Fouriertransformation auf dünnen Gittern mit hierarchischen Basen

342/10/91 A Erwin Loibl, Hans Obermaier, Markus Pawlowski: Towards Parallelism in a Relational Database System

342/11/91 A Michael Werner: Implementierung von Algorithmen zur Kompaktifizierung von Programmen für VLIW-Architekturen

342/12/91 A Reiner Müller: Implementierung von Algorithmen zur Optimierung von Schleifen mit Hilfe von Software-Pipelining Techniken

342/13/91 A Sally Baker, Hans-Jörg Beier, Thomas Bemmerl, Arndt Bode, Hubert Ertl, Udo Graf, Olav Hansen, Josef Haunerdinger, Paul Hofstetter, Rainer Knödlseder, Jaroslav Kremenek, Siegfried Langenbuch, Robert Lindhof, Thomas Ludwig, Peter Luksch, Roy Milner, Bernhard Ries, Thomas Treml: TOPSYS - Tools for Parallel Systems (Artikelsammlung); 2., erweiterte Auflage

342/14/91 A Michael Griebel: The combination technique for the sparse grid solution of PDE's on multiprocessor machines

342/15/91 A Thomas F. Gritzner, Manfred Broy: A Link Between Process Algebras and Abstract Relation Algebras?

342/16/91 A Thomas Bemmerl, Arndt Bode, Peter Braun, Olav Hansen, Thomas Treml, Roland Wismüller: The Design and Implementation of TOPSYS

342/17/91 A Ulrich Furbach: Answers for disjunctive logic programs

342/18/91 A Ulrich Furbach: Splitting as a source of parallelism in disjunctive logic programs

342/19/91 A Gerhard W. Zumbusch: Adaptive parallele Multilevel-Methoden zur Lösung elliptischer Randwertprobleme

342/20/91 A M. Jobmann, J. Schumann: Modelling and Performance Analysis of a Parallel Theorem Prover

342/21/91 A Hans-Joachim Bungartz: An Adaptive Poisson Solver Using Hierarchical Bases and Sparse Grids

342/22/91 A Wolfgang Ertel, Theodor Gemenis, Johann M. Ph. Schumann, Christian B. Suttner, Rainer Weber, Zongyan Qiu: Formalisms and Languages for Specifying Parallel Inference Systems

342/23/91 A Astrid Kiehn: Local and Global Causes

342/24/91 A Johann M.Ph. Schumann: Parallelization of Inference Systems by using an Abstract Machine

342/25/91 A Eike Jessen: Speedup Analysis by Hierarchical Load Decomposition

342/26/91 A Thomas F. Gritzner: A Simple Toy Example of a Distributed System: On the Design of a Connecting Switch

342/27/91 A Thomas Schnekenburger, Andreas Weininger, Michael Friedrich: Introduction to the Parallel and Distributed Programming Language ParMod-C

Reihe A

342/28/91 A	Claus Dendorfer: Funktionale Modellierung eines Postsystems
342/29/91 A	Michael Griebel: Multilevel algorithms considered as iterative methods on indefinite systems
342/30/91 A	W. Reisig: Parallel Composition of Liveness
342/31/91 A	Thomas Bemmerl, Christian Kasperbauer, Martin Mairandres, Bernhard Ries: Programming Tools for Distributed Multiprocessor Computing Environments
342/32/91 A	Frank Leške: On constructive specifications of abstract data types using temporal logic
342/1/92 A	L. Kanal, C.B. Suttner (Editors): Informal Proceedings of the Workshop on Parallel Processing for AI
342/2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS
342/2-2/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: The Design of Distributed Systems - An Introduction to FOCUS - Revised Version (erschienen im Januar 1993)
342/3/92 A	Manfred Broy, Frank Dederichs, Claus Dendorfer, Max Fuchs, Thomas F. Gritzner, Rainer Weber: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems
342/4/92 A	Claus Dendorfer, Rainer Weber: Development and Implementation of a Communication Protocol - An Exercise in FOCUS
342/5/92 A	Michael Friedrich: Sprachmittel und Werkzeuge zur Unterstüt- zung paralleler und verteilter Programmierung
342/6/92 A	Thomas F. Gritzner: The Action Graph Model as a Link between Abstract Relation Algebras and Process-Algebraic Specifications
342/7/92 A	Sergei Gorlatch: Parallel Program Development for a Recursive Numerical Algorithm: a Case Study
342/8/92 A	Henning Spruth, Georg Sigl, Frank Johannes: Parallel Algorithms for Slicing Based Final Placement
342/9/92 A	Herbert Bauer, Christian Sporrer, Thomas Krodel: On Distributed Logic Simulation Using Time Warp
342/10/92 A	H. Bungartz, M. Griebel, U. Rüde: Extrapolation, Combination and Sparse Grid Techniques for Elliptic Boundary Value Problems
342/11/92 A	M. Griebel, W. Huber, U. Rüde, T. Störtkuhl: The Combination Technique for Parallel Sparse-Grid-Preconditioning and -Solution of PDEs on Multiprocessor Machines and Workstation Networks
342/12/92 A	Rolf Niedermeier, Peter Rossmanith: Optimal Parallel Algorithms for Computing Recursively Defined Functions
342/13/92 A	Rainer Weber: Eine Methodik für die formale Anforderungsspezifkation verteilter Systeme
342/14/92 A	Michael Griebel: Grid– and point–oriented multilevel algorithms

Reihe A

342/15/92 A M. Griebel, C. Zenger, S. Zimmer: Improved multilevel algorithms for full and sparse grid problems

342/16/92 A J. Desel, D. Gomm, E. Kindler, B. Paech, R. Walter: Bausteine eines kompositionalen Beweiskalküls für netzmodellierte Systeme

342/17/92 A Frank Dederichs: Transformation verteilter Systeme: Von applikativen zu prozeduralen Darstellungen

342/18/92 A Andreas Listl, Markus Pawlowski: Parallel Cache Management of a RDBMS

342/19/92 A Erwin Loibl, Markus Pawlowski, Christian Roth: PART: A Parallel Relational Toolbox as Basis for the Optimization and Interpretation of Parallel Queries

342/20/92 A Jörg Desel, Wolfgang Reisig: The Synthesis Problem of Petri Nets

342/21/92 A Robert Balder, Christoph Zenger: The d-dimensional Helmholtz equation on sparse Grids

342/22/92 A Ilko Michler: Neuronale Netzwerk-Paradigmen zum Erlernen von Heuristiken

342/23/92 A Wolfgang Reisig: Elements of a Temporal Logic. Coping with Concurrency

342/24/92 A T. Störtkuhl, Chr. Zenger, S. Zimmer: An asymptotic solution for the singularity at the angular point of the lid driven cavity

342/25/92 A Ekkart Kindler: Invariants, Compositionality and Substitution

342/26/92 A Thomas Bonk, Ulrich Rüde: Performance Analysis and Optimization of Numerically Intensive Programs

342/1/93 A M. Griebel, V. Thurner: The Efficient Solution of Fluid Dynamics Problems by the Combination Technique

342/2/93 A Ketil Stølen, Frank Dederichs, Rainer Weber: Assumption / Commitment Rules for Networks of Asynchronously Communicating Agents

342/3/93 A Thomas Schnekenburger: A Definition of Efficiency of Parallel Programs in Multi-Tasking Environments

342/4/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: A Proof of Convergence for the Combination Technique for the Laplace Equation Using Tools of Symbolic Computation

342/5/93 A Manfred Kunde, Rolf Niedermeier, Peter Rossmanith: Faster Sorting and Routing on Grids with Diagonals

342/6/93 A Michael Griebel, Peter Oswald: Remarks on the Abstract Theory of Additive and Multiplicative Schwarz Algorithms

342/7/93 A Christian Sporrer, Herbert Bauer: Corolla Partitioning for Distributed Logic Simulation of VLSI Circuits

342/8/93 A Herbert Bauer, Christian Sporrer: Reducing Rollback Overhead in Time-Warp Based Distributed Simulation with Optimized Incremental State Saving

342/9/93 A Peter Slavkovsky: The Visibility Problem for Single-Valued Surface (z = f(x,y)): The Analysis and the Parallelization of Algorithms

Reihe A

342/10/93 A Ulrich Rüde: Multilevel, Extrapolation, and Sparse Grid Methods

342/11/93 A Hans Regler, Ulrich Rüde: Layout Optimization with Algebraic Multigrid Methods

342/12/93 A Dieter Barnard, Angelika Mader: Model Checking for the Modal Mu-Calculus using Gauß Elimination

342/13/93 A Christoph Pflaum, Ulrich Rüde: Gauß' Adaptive Relaxation for the Multilevel Solution of Partial Differential Equations on Sparse Grids

342/14/93 A Christoph Pflaum: Convergence of the Combination Technique for the Finite Element Solution of Poisson's Equation

342/15/93 A Michael Luby, Wolfgang Ertel: Optimal Parallelization of Las Vegas Algorithms

342/16/93 A Hans-Joachim Bungartz, Michael Griebel, Dierk Röschke, Christoph Zenger: Pointwise Convergence of the Combination Technique for Laplace's Equation

342/17/93 A Georg Stellner, Matthias Schumann, Stefan Lamberts, Thomas Ludwig, Arndt Bode, Martin Kiehl und Rainer Mehlhorn: Developing Multicomputer Applications on Networks of Workstations Using NXLib

342/18/93 A Max Fuchs, Ketil Stølen: Development of a Distributed Min/Max Component

342/19/93 A Johann K. Obermaier: Recovery and Transaction Management in Write-optimized Database Systems

342/20/93 A Sergej Gorlatch: Deriving Efficient Parallel Programs by Systemating Coarsing Specification Parallelism

342/01/94 A Reiner Hüttl, Michael Schneider: Parallel Adaptive Numerical Simulation

342/02/94 A Henning Spruth, Frank Johannes: Parallel Routing of VLSI Circuits Based on Net Independency

342/03/94 A Henning Spruth, Frank Johannes, Kurt Antreich: PHIroute: A Parallel Hierarchical Sea-of-Gates Router

342/04/94 A Martin Kiehl, Rainer Mehlhorn, Matthias Schumann: Parallel Multiple Shooting for Optimal Control Problems Under NX/2

342/05/94 A Christian Suttner, Christoph Goller, Peter Krauss, Klaus-Jörn Lange, Ludwig Thomas, Thomas Schnekenburger: Heuristic Optimization of Parallel Computations

342/06/94 A Andreas Listl: Using Subpages for Cache Coherency Control in Parallel Database Systems

342/07/94 A Manfred Broy, Ketil Stølen: Specification and Refinement of Finite Dataflow Networks - a Relational Approach

342/08/94 A Katharina Spies: Funktionale Spezifikation eines Kommunikationsprotokolls

342/09/94 A Peter A. Krauss: Applying a New Search Space Partitioning Method to Parallel Test Generation for Sequential Circuits

Reihe A

342/10/94 A Manfred Broy: A Functional Rephrasing of the Assumption/Commitment Specification Style

342/11/94 A Eckhardt Holz, Ketil Stølen: An Attempt to Embed a Restricted Version of SDL as a Target Language in Focus

342/12/94 A Christoph Pflaum: A Multi-Level-Algorithm for the Finite-Element-Solution of General Second Order Elliptic Differential Equations on Adaptive Sparse Grids

342/13/94 A Manfred Broy, Max Fuchs, Thomas F. Gritzner, Bernhard Schätz, Katharina Spies, Ketil Stølen: Summary of Case Studies in FOCUS - a Design Method for Distributed Systems

342/14/94 A Maximilian Fuchs: Technologieabhängigkeit von Spezifikationen digitaler Hardware

342/15/94 A M. Griebel, P. Oswald: Tensor Product Type Subspace Splittings And Multilevel Iterative Methods For Anisotropic Problems

342/16/94 A Gheorghe Ştefănescu: Algebra of Flownomials

342/17/94 A Ketil Stølen: A Refinement Relation Supporting the Transition from Unbounded to Bounded Communication Buffers

342/18/94 A Michael Griebel, Tilman Neuhoeffer: A Domain-Oriented Multilevel Algorithm-Implementation and Parallelization

342/19/94 A Michael Griebel, Walter Huber: Turbulence Simulation on Sparse Grids Using the Combination Method

342/20/94 A Johann Schumann: Using the Theorem Prover SETHEO for verifying the development of a Communication Protocol in FOCUS - A Case Study -

342/01/95 A Hans-Joachim Bungartz: Higher Order Finite Elements on Sparse Grids

342/02/95 A Tao Zhang, Seonglim Kang, Lester R. Lipsky: The Performance of Parallel Computers: Order Statistics and Amdahl's Law

342/03/95 A Lester R. Lipsky, Appie van de Liefvoort: Transformation of the Kronecker Product of Identical Servers to a Reduced Product Space

342/04/95 A Pierre Fiorini, Lester R. Lipsky, Wen-Jung Hsin, Appie van de Liefvoort: Auto-Correlation of Lag-k For Customers Departing From Semi-Markov Processes

342/05/95 A Sascha Hilgenfeldt, Robert Balder, Christoph Zenger: Sparse Grids: Applications to Multi-dimensional Schrödinger Problems

342/06/95 A Maximilian Fuchs: Formal Design of a Model-N Counter

342/07/95 A Hans-Joachim Bungartz, Stefan Schulte: Coupled Problems in Microsystem Technology

342/08/95 A Alexander Pfaffinger: Parallel Communication on Workstation Networks with Complex Topologies

342/09/95 A Ketil Stølen: Assumption/Commitment Rules for Data-flow Networks - with an Emphasis on Completeness

342/10/95 A Ketil Stølen, Max Fuchs: A Formal Method for Hardware/Software Co-Design

Reihe A

342/11/95 A Thomas Schnekenburger: The ALDY Load Distribution System

342/12/95 A Javier Esparza, Stefan Römer, Walter Vogler: An Improvement of McMillan's Unfolding Algorithm

342/13/95 A Stephan Melzer, Javier Esparza: Checking System Properties via Integer Programming

342/14/95 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Point-to-Point Dataflow Networks

342/15/95 A Andrei Kovalyov, Javier Esparza: A Polynomial Algorithm to Compute the Concurrency Relation of Free-Choice Signal Transition Graphs

342/16/95 A Bernhard Schätz, Katharina Spies: Formale Syntax zur logischen Kernsprache der Focus-Entwicklungsmethodik

342/17/95 A Georg Stellner: Using CoCheck on a Network of Workstations

342/18/95 A Arndt Bode, Thomas Ludwig, Vaidy Sunderam, Roland Wismüller: Workshop on PVM, MPI, Tools and Applications

342/19/95 A Thomas Schnekenburger: Integration of Load Distribution into ParMod-C

342/20/95 A Ketil Stølen: Refinement Principles Supporting the Transition from Asynchronous to Synchronous Communication

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

342/1/90 B Wolfgang Reisig: Petri Nets and Algebraic Specifications

342/2/90 B Jörg Desel: On Abstraction of Nets

342/3/90 B Jörg Desel: Reduction and Design of Well-behaved Free-choice Systems

342/4/90 B Franz Abstreiter, Michael Friedrich, Hans-Jürgen Plewan: Das Werkzeug runtime zur Beobachtung verteilter und paralleler Programme

342/1/91 B Barbara Paech1: Concurrency as a Modality

342/2/91 B Birgit Kandler, Markus Pawlowski: SAM: Eine Sortier- Toolbox - Anwenderbeschreibung

342/3/91 B Erwin Loibl, Hans Obermaier, Markus Pawlowski: 2. Workshop über Parallelisierung von Datenbanksystemen

342/4/91 B Werner Pohlmann: A Limitation of Distributed Simulation Methods

342/5/91 B Dominik Gomm, Ekkart Kindler: A Weakly Coherent Virtually Shared Memory Scheme: Formal Specification and Analysis

342/6/91 B Dominik Gomm, Ekkart Kindler: Causality Based Specification and Correctness Proof of a Virtually Shared Memory Scheme

342/7/91 B W. Reisig: Concurrent Temporal Logic

342/1/92 B Malte Grosse, Christian B. Suttner: A Parallel Algorithm for Set-of-Support

Christian B. Suttner: Parallel Computation of Multiple Sets-of-Support

342/2/92 B Arndt Bode, Hartmut Wedekind: Parallelrechner: Theorie, Hardware, Software, Anwendungen

342/1/93 B Max Fuchs: Funktionale Spezifikation einer Geschwindigkeitsregelung

342/2/93 B Ekkart Kindler: Sicherheits- und Lebendigkeitseigenschaften: Ein Literaturüberblick

342/1/94 B Andreas Listl; Thomas Schnekenburger; Michael Friedrich: Zum Entwurf eines Prototypen für MIDAS