A compositional proof theory for real-time distributed message passing

Universiteit
Faculteit der Wiskunde
en Informatica

A Compositional Proof Theory for Real-Time Distributed Message Passing

March 1987

J. Hooman

A Compositional Proof Theory for Real-Time Distributed Message Passing

March 1987

J. Hooman

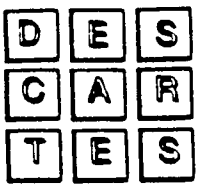
86.10

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science of Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author or the editor.

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands

editor: F.A.J. van Neerven



European Strategic Programme of Research and Development in Information Technology

Project 937 : Debugging and Specification of Ada Real-Time Embedded Systems
Package 4 : Formal Semantics and Proof Systems for Real-Time Languages

Mail to :
Doc. No. : TR. 4-1-1(1)
Type : TR
Title : A Compositional Proof Theory for Real-Time Distributed Message Passing
Author : J. Hooman
Date : 12-1-86 Version : 0
Replaces:
Document Status : Submitted
Confidentiality Level : Public-domain

GSI-TECSI
SYSTEAM KG
FOXBORO Netherlands NV
ELECTRONIQUE SERGE DASSAULT
EINDHOVEN UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF STIRLING
ADCAD Ltd

CONTENTS

A COMPOSITIONAL PROOF THEORY FOR REAL-TIME DISTRIBUTED MESSAGE PASSING

Jozef Hooman *

Department of Mathematics &
Computing Science
Eindhoven University of Technology
P.O. box 513
5600 MB Eindhoven
The Netherlands

March 1987

ABSTRACT

A compositional proof system is given for an OCCAM-like real-time programming language for distributed computing with communication via synchronous message passing. This proof system is based on specifications of processes which are independent of the program text of these processes. These specifications state (1) the assumptions of a process about the behaviour of its environment, and (2) the commitments of that process towards that environment provided these assumptions are met. The proof system is sound w.r.t a denotational semantics which incorporates assumptions regarding actions of the environment, thereby closely approximating the assumption/commitment style of reasoning on which the proof system is based. Concurrency is modelled as "maximal parallelism"; that is, if a process can proceed it will do so immediately. A process only waits when no local action is possible and no partner is available for communication. This maximality property is imposed on the domain of interpretation of assertions by postulating it as separate axiom. The timing behaviour of a system is expressed from the viewpoint of a global external observer, so there is a global notion of time. Time is not necessarily discrete.

---

* supported by Esprit Project 937: Debugging and Specification of Ada Real-Time Embedded Systems (DESCARTES).
Electronic-mail address: mcvax!eutrc3!wsinjh.UUCP or wsdcjh@heithe5.BITNET .

1. INTRODUCTION

Recently attention has been drawn to the discrepancy between the growing number of real-time applications - industrial process control, telecommunication, life support systems in hospitals, avionics systems used for guidance and control, to mention but a few - and the existing theoretical background for such systems. For concurrency and hard time limits make the design and development of real-time embedded systems very complex, and certainly testing is not sufficient to validate a program. Also, in many real-time applications failure is very expensive and can have disastrous consequences. So, especially in this area of real-time systems, there is a growing need for formal specification and verification techniques in order to provide assistance in the "lost world" of real-time software development (see [Glass]).

The ESPRIT project DESCARTES provides a context for investigating these problems. A simple language akin to OCCAM ([OCC]) is considered for capturing the essential features of real-time in the context of distributed message passing. It is based on CSP (Communicating Sequential Processes [Hoare]), a language for concurrent programs with communication via synchronous message-passing. Contrary to CSP, where communicating partners explicitly name each other, here communication occurs along unidirectional channels between pairs of processes. Added is the real-time statement *DELAY d*, which suspends the execution for the specified number of time units. Such a DELAY-statement may occur in the guard of an alternative command. Together with the underlying execution model this gives the opportunity to program a time-out. The execution model is that of "*maximal parallelism*". That is, if a process can proceed it will do so immediately. A process only waits when no local action is possible and no partner is available for communication. As soon as an action becomes possible execution must proceed.

New in this paper is a compositional Hoare-style proof system for safety properties of real-time distributed processes. The maximal parallelism constraint is modeled as an axiom for the domain of interpretation of assertions which may be used throughout the proof system. To obtain specifications of processes which are independent of their program text, Hoare triples are extended with invariants which should hold throughout program execution. This is needed in particular when specifying the communication and timing behaviour of nonterminating processes - the usual kind of processes when considering real-time - independent of their text. The invariants do not refer to any internal state of the process during execution.

What should be the form of such an invariant?

In general, the behaviour of a process depends on its environment; for instance on the values sent by the environment. Incorporating real-time makes this dependency even greater. The timing behaviour of a process will now also depend on the time at which the environment is ready to communicate, on how long a communication is enabled by the environment, etc. Consequently, knowledge about the environment is an important factor in the design of a real-time process. Therefore, we aim at *specifying processes within their environment*, and in the resulting specifications *the knowledge about that environment should be reflected by imposing suitable assumptions.*

To allow process behaviour to be specified relative to such assumptions, we adopt the assumption/commitment-style of reasoning as described in [ZBR,ZRE84], which is based on [MC]. Using this formalism, the invariant of a process in our specifications consists of two parts:
an *assumption* describing the expected behaviour of the environment, and
a *commitment* which is guaranteed by the process itself, as long as the environment does not violate the assumption.
When two processes are composed in parallel, we then have to verify that the assumptions of one process about joint communications correspond to the commitments of the other process for these joint communications.

How can we adapt this assumption/commitment based formalism to deal with real-time? In the formalism of [ZBR] an assumption describes the communication behaviour of a process. Note that the communication behaviour of environment and process is identical when restricting to joint communications, since a channel connects exactly two processes and communication is synchronous. This simple picture changes when dealing with real-time. In our proof system we must be able to make assumptions concerning "wait actions" of the environment, e.g.:

- when is the environment ready to start a communication, when does it start waiting,
- how long will the environment wait for a particular communication,
- when does the environment stop waiting for a communication.

Next observe that such wait actions concerning joint communications are different for environment and process. For instance, regardless of maximal parallelism the waiting period for the same communication, will in general differ for process and environment. Consequently, we distinguish between wait actions of the process and wait actions of the environment. This distinction is reflected in the proof system as follows. The assumption of a process refers to the wait actions of the environment, whereas the commitment refers to the wait actions of that process itself.

In our semantics wait actions are represented by so called *wait records*, which denote the waiting period of a process for a communication. Because the assertions in the specification will refer to wait actions of the environment, *environment* wait records are included in the semantics, too. By means of these environment records the maximal parallelism constraint is imposed on *every* element of the semantic domain by requiring that, for a particular channel, the waiting period denoted by a wait record does not overlap with the waiting period denoted by an environment wait record. Consequently, when processes are composed in parallel no explicit check on maximality is needed. At parallel composition we only have to check additionally that the assumptions made by one process concerning the wait records of the environment must be fulfilled by the other process as far as it concerns their joint channels.

Characteristic of compositional proof systems for concurrency is the conjunctive nature of their parallel composition rules, i.e. the parallel composition of two processes satisfies the conjunction of their specifications. Within our proof system this conjunctive character is preserved, since the commitment of a network is, in principle, the conjunction of the commitments of the components. This is the other reason why environment wait records have been incorporated within our semantics. For their presence allows the essentially complementary character of the maximal parallelism constraint - when I wait you don't - to become internalised within the specification of a process by imposing maximal parallelism as a separate axiom. Therefore our parallel composition rule requires no separate clause for checking maximal parallelism.

The introduction of wait records raises the question whether it is possible to characterise real-time distributed message passing in a compositional fashion *without* such records. If termination, communication along channels, and the time communication takes place are the observables of a process, the answer to this question is no. Specifically, the full abstraction result of [HGR] implies that if wait records - or something equivalent - are not included in the denotational semantics, then it is possible to give two programs with the same semantics, but observably different behaviour. So, given our specific observables, without wait records the semantics would be unsound.
The semantics given in [KSRGA] served as starting point for our semantics, and it has been changed to come as close as possible to that of [ZRE]. The global notion of time used in [KSRGA] is maintained in our semantics. This is justified because we want to express the timing behaviour of a system from the viewpoint of a global external observer with his own clock. So, at the level of reasoning there is a conceptual global clock. New is that, in deviation of [KSRGA], time is not necessarily discrete.

This paper is structured as follows. Chapter 2 contains the syntax of the language considered and its intuitive semantics. In chapter 3 a denotational semantics is defined. The correctness formula and the assertion language are described in chapter 4. The

main chapter is chapter 5, where a compositional proof system is given for our real-time programming language for distributed computing with communication via synchronous message passing. The conclusion can be found in chapter 6, together with a discussion of future work. Finally, in the appendix the proof system of chapter 5 is proven sound w.r.t. the semantics of chapter 3.

ACKNOWLEDGEMENTS

The author thanks the members of the EUT-team involved in ESPRIT project Descartes for clarifying discussions. Especially Willem-Paul de Roever and Rob Gerth provided many useful comments and valuable advice; in fact they rewrote the paper after having been presented with my draft. Amir Pnueli provided stimulation by his interest and suggestions. All this, however, would have been of no use hadn't it been for the work of Job Zwiers on compositionality of proof systems for concurrent networks, and the insight in their intricacies which he shared with the author.

2. SYNTAX

In this chapter we give the syntax of a real-time programming language for distributed synchronous message-passing. This language is essentially OCCAM ([OCC]). Communication takes place through unidirectional channels which connect exactly two processes. There is a delay-statement, which may appear in the guard of an alternative statement, too. Such a delay-branch causes a time-out if no communications were offered during the delay period. We separate the concepts of parallel composition and hiding of internal communications by introducing an explicit hiding operator [..].

In the syntax below D will stand for a channel name, d and e for expressions, b for a boolean expression, and x for a program variable.

Language construction

$L ::= S \mid N$

Statement

$S ::= x := e \mid SKIP \mid IO \mid DELAY\ d \mid S_1 ; S_2 \mid [N] \mid A \mid {}^*A$

Alternative

$$A ::= [\ \Box_{i=1}^{n_1} b_i \rightarrow S_i \ \Box_{i=1}^{n_2} b_i'; DELAY\ d_i \rightarrow S_i' \ \Box_{i=1}^{n_3} b_i''; IO_i \rightarrow S_i'']$$

Input/Output

IO ::= $D!e \mid D?x$

Network

N ::= $S_1 \| S_2$

A boolean expression b_i' or b_i'' is omitted if it is *TRUE*.

2.1 *Informal semantics*

- *SKIP* skip: only affects the execution time.

- $x := e$ assignment: the value of expression e is assigned to the variable x.

- $D!e$ output: send the value of expression e through channel D; this action synchronizes with a corresponding input command.

- $D?x$ input: receive via channel D a value and assign this value to the variable x; this action synchronizes with a corresponding output command.

- *DELAY* d delay: suspends the execution for (the value of) d time units. A delay statement with a negative value is equivalent to a delay statement with a zero value.

- $S_1; S_2$ sequential composition: execute S_2 after having executed S_1.

- $[N]$ hiding: the internal communications of network N are no longer visible.

- A alternative:
  A guard is open if the boolean part evaluates to true. Following [KSRGA] we give priority to purely boolean guards. So if at least one of the b_i is true then select non-deterministically one of the open purely boolean guards and execute the corresponding branch. If none of the purely boolean guards is open and none of the other guards is open execution aborts. Otherwise, let *mindelay* be the minimum of the delay-values of the open delay-guards (infinite if there are no open delay-guards). If within *mindelay* time units at least one IO-command of the open IO-guards can be executed, select non-deterministically one of them and execute the guard and the corresponding branch. Otherwise, if no IO-guard can be taken within *mindelay* time units, one of the open delay-guards with delay value equal to *mindelay* is selected.

- $*A$ iteration: repeated execution of alternative A as long as at least one of the guards is open. When none of the guards is open execution terminates.

- $S_1 \| S_2$ network: parallel execution of S_1 and S_2, based on the maximal parallelism model; no process ever waits unnecessarily, if execution can proceed it will do so immediately.

2.2 Syntactic restrictions

First some definitions:
var (L) denotes the program variables occurring in language construction L,
chan (L) denotes the set of channel names in language construction L, and
type (IO) denotes the channel of the IO-command.

In a network $S_1 \| S_2$ the concurrent processes S_1 and S_2 are not allowed to have shared variables. Thus $var(S_1) \cap var(S_2) = \varnothing$.
Channels are unidirectional and connect exactly two processes.
For $S_1 \| S_2$ we require that S_1 and S_2 do not have joint input channels or joint output channels. So the joint channels of $S_1 \| S_2$, i.e. $chan(S_1) \cap chan(S_2)$, are exactly those channels through which S_1 and S_2 may communicate with each other.

Throughout this paper we use $\equiv$ to denote syntactic equality.

3. SEMANTICS

In [KSRGA] a denotational semantics has been given for CSP-R, a language similar to that of the previous chapter but with communication by means of process naming instead of channels. That semantics is based on the linear history semantics for CSP of [FLP]. The basic domain consists of non-empty prefix-closed sets of pairs of states and (finite) histories. To characterise maximal parallelism, such a history contains besides "communication records", which denote actual communications, also "no-match records" to denote that a process is waiting for a communication. Furthermore, the length of a trace represents the time. In view of the desired proof system, which should be based on the assumption/commitment type of correctness formula from [ZBR,ZRE84], we

reformulate this semantics. The new semantics should be as close as possible to the semantics described in [ZRE], which is formulated in terms of trace-state pairs, where a *trace* is defined as a sequence of communication records only.

We extend a trace-state pair to a 5-tuple, consisting of components for the communication trace, the set of wait records of the process, the set of wait records of the environment, the state and the time. *Wait records* denote the waiting of a process for a communication. In our proof system we want to express assumptions concerning wait actions of the environment, so the semantics contains also a set of *environment* wait records. These environment wait records are used to model maximal parallelism, by requiring that for every tuple in the semantic domain the set of wait records and the set of environment wait records satisfy this maximality constraint. That is, for a particular channel there is no overlap of the waiting periods denoted by a wait record and an environment wait record.
We take the same global notion of time as in [KSRGA]; however, we do not assume discreteness of time.

In the next section we describe 5-tuples, which form the basis of our semantic domain of denotations. In section 3.2 an ordering on these tuples is defined, which is used for a formal definition of correctness formulae in chapter 4, and which is needed to obtain, in section 3.3, a complete partial order as domain of denotations. Finally, the particular function defining the semantics is given in section 3.4.

3.1 Our basic 5-tuples

In this section we define our basic 5-tuples, which form the basis of the semantic domain.
Assume a given time domain $TIME$, and a domain VAL for values of identifiers. To avoid an elaborate distinction between the types $TIME$ and VAL, e.g. the distinction between $TIME$-expressions and VAL-expressions, we choose VAL such that $VAL = TIME$. Furthermore we assume that $0 \in VAL$, and $v+w$, $v<w$, $v=w$ are defined in VAL.

The basic domain of denotations for the semantics of a process consists of sets of tuples $(\tau, W, W^e, \sigma, \alpha)$, where:

- τ is a communication trace; a sequence of communication records (t, D, v), with $t \in TIME$, D a channel name and $v \in VAL$. Informal meaning: at time t a communication via channel D starts and v is the communicated value.

- W is a set of wait records of that process; a wait record has the form (l,u,D), with $l,u \in TIME$ and D a channel name. Informal meaning: wait from time l up to time u for a communication via channel D.

- W^e is a set of wait records of the *environment*.

- σ is a state; a mapping from identifiers to values ($\sigma \in STATE$) or $\perp$, indicating an unfinished computation.

- $\alpha \in TIME \cup \{\perp\}$.

Such a 5-tuple indicates a "point" in a computation, i.e., it reflects the state of affairs in a computation at a certain point of time.
A tuple $(\tau,W',W^e,\sigma,\alpha)$ with $\sigma \neq \perp$, $\alpha \neq \perp$ models a finished computation, which terminates at time α in state σ, with trace τ and set of wait records W' produced during the computation. W^e represents the assumption about the wait actions performed by the environment up to and including termination time α.
Tuples $(\tau,W',W^e,\sigma,\alpha)$ with $\sigma = \perp$ and $\alpha = \perp$, modeling unfinished computations, are needed to obtain prefix closed sets of 5-tuples, and to model infinite computations through an infinite chain of approximations.

3.2 Ordering on tuples

In this section we extend the usual prefix ordering for sequences to our 5-tuples. In the sequel s will stand for the tuple $(\tau,W',W^e,\sigma,\alpha)$, and similar $s' = (\tau',W',W^{e\prime},\sigma',\alpha')$, $\hat{s} = (\hat{\tau},\hat{W},\hat{W}^e,\hat{\sigma},\hat{\alpha})$, etc.

We define the ordering $\leqslant$ on tuples as follows. Let $s' \leqslant s$ denote that either $s' = s$, or s' precedes s in a computation. In the latter case, s' represents an unfinished computation, thus $\sigma' = \perp$ and $\alpha' = \perp$. Moreover, if s' precedes s in a computation then trace τ' should be a prefix of τ, W' a subset of W, and $W^{e\prime}$ a subset of W^e. The following example shows that we have to take care that s' really represents a point of time in a computation leading to s.

ex.
$(<(3,..,..),(9,..,..)>,\{(1,4,..)\},\varnothing,\perp,\perp) \not\leqslant (<(3,..,..),(9,..,..)>,\{(1,4,..),(7,8,..)\},\varnothing,\perp,\perp)$,
because the left tuple can not represent a point of time in a computation leading to the right tuple; the wait record (7,8,..) has not yet been added to the left tuple, although the communication record (9,..,..), which corresponds to a later point of time, is already present. So if we remove this record (9,..,..) from the left tuple, we obtain
$(<(3,..,..)>,\{(1,4,..)\},\varnothing,\perp,\perp) \leqslant (<(3,..,..),(9,..,..)>,\{(1,4,..),(7,8,..)\},\varnothing,\perp,\perp)$.

Furthermore $(<>,\{(1,4,..)\},\varnothing,\perp,\perp) \not\leqslant (<(3,..,..),(9,.....)>,\{(1,4,..),(7,8,..)\},\varnothing,\perp,\perp)$, since the left tuple contains wait record (1,4,..), whereas communication record (3,..,..) has not yet been added. But then then the left tuple can not represent a tuple in a computation leading to the right tuple, since wait records are added in the semantics when the waiting finishes. In this example (1,4,..) is added at time 4, so also (3,..,..) should be included in the left tuple.
□

Let $<>$ denote the empty trace, then $(<>,\varnothing,\varnothing,\perp,\perp)$ represents the situation where nothing has happened yet. It denotes the start of every computation, so $(<>,\varnothing,\varnothing,\perp,\perp) \leqslant s$ for every tuple s.

These considerations lead to the following, informal, definition of $s' \leqslant s$:
s' is equal to s, or
s' represents an unfinished computation at a certain point of time, say $\hat{\alpha}$, where, τ', W' and W'^{e} are the *restriction* of τ, W and W^{e}, resp., to $\hat{\alpha}$, or
s' denotes the initial tuple of a computation $(<>,\varnothing,\varnothing,\perp,\perp)$.

To formalise this, define the restriction, $\tau\downarrow\alpha$, of a trace τ to a time α as the initial prefix of τ for which the following holds: $(t,D,v)\in\tau\downarrow\alpha \rightleftarrows (t,D,v)\in\tau \wedge t\leqslant\alpha$.
The restriction, $W\downarrow\alpha$, of a set of wait records W to time α is defined as follows:
$W\downarrow\alpha = \{(l,u,D)\in W \mid u\leqslant\alpha\}$.

Then the ordering on tuples, $s'\leqslant s$, is defined by
$s' = s \vee (\alpha'=\perp \wedge \sigma'=\perp \wedge \exists\hat{\alpha}[\,\tau'=\tau\downarrow\hat{\alpha} \wedge W'=W\downarrow\hat{\alpha} \wedge W'^{e}=W^{e}\downarrow\hat{\alpha}\,]) \vee$
$s' = (<>,\varnothing,\varnothing,\perp,\perp)$.

3.3 Domain of denotations

In this section the tuples and their ordering are used to define the semantic domain of denotations. (We assume the reader to be familiar with complete partial orderings, see [deB].) This semantic domain $\mathbb{D}$ is restricted to those tuples that satisfy the **maximal parallelism constraint,** that is, never two processes both wait for the same communication. For the wait records in a tuple s this means that W and W^{e} never contain wait records for the same communication that overlap in time.
Let $[.....>$ denote a left closed, right open interval, and let W and W' be sets of wait records. Then we formulate this constraint as follows:

$\mathbf{MP}(W, W') \rightleftarrows \forall(l,u,D)\in W\ \forall(l',u',D)\in W'\,[\,[l,u> \cap [l',u'> = \varnothing\,]$.

Furthermore, traces occurring in tuples of the semantic domain will always be *time-ordered :* for a trace τ, predicate $time-ordered(\tau)$ is true iff the sequence of time stamps in the records of τ is non-decreasing.
So in the sequel we restrict us to the following set of tuples:

$\mathbb{B} = \{(\tau, W, W^e, \sigma, \alpha) \mid MP(W, W^e) \wedge time-ordered(\tau)\}$.

Let U be a set of tuples. The prefix closure of U is defined as
PFC(U) = $\{ s' \mid s' \leqslant s , s \in U \}$.
U is called prefix closed iff $PFC(U) = U$.

The basic domain of denotations is the set of all nonempty, prefix closed subsets of $\mathbb{B}$,
$\mathbb{D} = \{D \mid D \subseteq \mathbb{B} \wedge D \neq \emptyset \wedge PFC(D) = D\}$.

Next we define the, so called, Hoare order on $\mathbb{D}$ (let $V, W \in \mathbb{D}$):
$V \leqslant_H W \rightleftarrows \forall s \in V \; \exists s' \in W \; [s \leqslant s']$,
which corresponds to the usual set inclusion order:
$V \leqslant_H W \rightleftarrows V \subseteq W$, for all $V, W \in \mathbb{D}$.

So $(\mathbb{D}, \subseteq)$ is a complete partial order, with the singleton set $\{(<>, \emptyset, \emptyset, \perp, \perp)\}$ as least element.

3.4 The function defining the semantics

Finally the particular function defining the semantics is given.
Assume a function T has been given, which assigns to every *atomic* statement S (i.e. skip, assignment, io, delay) and state σ an interval $T_\sigma(S)$, such that the execution time of this statement in this state is an element of the given interval. For the alternative statement A, $T_\sigma(A)$ denotes the overhead needed to execute this statement (e.g. evaluation of boolean guards, selection of an open guard, etc.). We assume that there is no overhead for the other composite constructs.

Assume the existence of semantic functions $[\![..]\!]$ for VAL expressions e and boolean expressions b : $[\![e]\!]\sigma$, $[\![b]\!]\sigma$.

Let $WAIT = \{(l, u, D) \mid l, u \in TIME, l \leqslant u\}$ and
$WAIT_t = \{(l, u, D) \in WAIT \mid u \leqslant t\}$, for $t \in TIME$.

The variant of a state $\sigma \neq \perp$, $\sigma[v/x]$, is defined as

$$\begin{cases} \sigma[v/x](x) = v \\ \sigma[v/x](y) = \sigma(y), \text{ if } y \neq x. \end{cases}$$

The semantics is now defined as a function M which maps a language construction L, given an initial state ($\neq \perp$) and starting time, to an element of $\mathbb{D}$:
$M : L \rightarrow (STATE \times TIME \rightarrow \mathbb{D})$.

skip

The semantics of the *skip* statement shows that the time component is updated with the execution time of this statement; all possible execution times between the bounds given by the T-function are included. Furthermore the environment may add a set of wait records E. Again all possibilities are included with the restriction that the upper bound of these records should be less then or equal to the actual time, i.e. $\alpha+t$. When processes are composed in parallel it is checked that for joint communications the set of environment wait records of one process equals the actual wait records of the other process.

By taking the prefix closure we obtain an element of $\mathbb{D}$.

$$M(SKIP)(\sigma,\alpha) = PFC\ (\{(<>, \varnothing, E, \sigma, \alpha+t) \mid E \subseteq WAIT_{\alpha+t} \wedge t \in T_\sigma(SKIP)\})$$

assignment

The *assignment* statement has a similar semantics, now also the state is updated.

$$M(x:=e)(\sigma,\alpha) = PFC\ (\{(<>, \varnothing, E, \sigma[^{[\![e]\!]\sigma}/_{x}], \alpha+t) \mid E \subseteq WAIT_{\alpha+t} \wedge t \in T_\sigma(x:=e)\})$$

delay

The *delay* statement updates the time component α with the specified time given by the T-function. This T-function should be such that $t \in T_\sigma(DELAY\ d)$ implies $t \geqslant [\![d]\!]\sigma$. Since a negative delay value yields a zero delay, the function *nonneg*, defined below, is applied to the delay value.

$$nonneg(v) = \begin{cases} 0 & \text{if } v<0, \\ v & \text{if } v \geqslant 0. \end{cases}$$

$$M(DELAY\ d)(\sigma,\alpha) = PFC\ (\{(<>, \varnothing, E, \sigma, \alpha+nonneg(t) \mid E \subseteq WAIT_{\alpha+nonneg(t)} \wedge t \in T_\sigma(DELAY\ d)\})$$

output

For the *output* command we include a communication record in the semantics. Assume the process has to wait w time units, then the actual communication starts at point of time $\alpha+w$.

Waiting for w time units is denoted by wait record $(\alpha,\alpha+w,D)$. Since waiting time w depends on the other process, we take all possible values for w.

The maximal parallelism constraint imposes a restriction on the wait records of the environment. These environment wait records must not overlap with the just added wait record of the process itself, so these overlapping records are excluded.

$$M(D!e)(\sigma,\alpha) = PFC\ (\ \{(<(\alpha+w,D,[\![e]\!]\sigma)>, \{(\alpha,\alpha+w,D)\}, E, \sigma, \alpha+w+t)\ |$$
$$w \in TIME\ \wedge w \geq 0\ \wedge t \in T_\sigma(D!e)\ \wedge$$
$$E \subseteq WAIT_{\alpha+w+t}\ \wedge MP(E,\{(\alpha,\alpha+w,D)\})\ \}\ )$$

input

The semantics of the *input* statement is similar to the output command, now the value received is not known, and we include all possible values. Again environment wait records which overlap with the waiting time are excluded.

$$M(D?x)(\sigma,\alpha) = PFC\ (\ \{(<(\alpha+w,D,v)>, \{(\alpha,\alpha+w,D)\}, E, \sigma[^v/_x], \alpha+w+t)\ |$$
$$w \in TIME\ \wedge w \geq 0\ \wedge v \in VAL\ \wedge t \in T_\sigma(D?x)\ \wedge$$
$$E \subseteq WAIT_{\alpha+w+t}\ \wedge MP(E,\{(\alpha,\alpha+w,D)\})\ \}\ )$$

sequential composition

In order to define the semantics of sequential composition, the semantic function is extended to initial tuples by defining $\overline{M(L)} : \{\, s \in \mathbb{B} \mid \sigma \neq \perp \,\} \rightarrow \mathbb{D}$.

First the concatenation of two tuples s_1 and s_2 is defined by

$s_1 s_2 = (\tau_1\tau_2, W_1 \cup W_2, W_1^e \cup W_2^e, \sigma_2, \alpha_2)$.

Then $\overline{M(L)}\hat{s} = \{\, \hat{s}\, s \mid s \in M(L)(\hat{\sigma},\hat{\alpha}) \wedge MP(W^e,\hat{W}) \wedge MP(\hat{W}^e,W)\,\}$.

Note that there is an explicit check to guarantee that the concatenation satisfies the maximal parallelism constraint.

The semantics of $S_1;S_2$ is defined as the union of two sets:

- the result of computing S_2 starting in a tuple representing a terminated computation of S_1.
- the tuples representing the unfinished computations of S_1.

$$M(S_1;S_2)(\hat{\sigma},\hat{\alpha}) = \{\, s \mid \exists s_1 [\, s_1 \in M(S_1)(\hat{\sigma},\hat{\alpha}) \wedge \sigma_1 \neq \perp \wedge s \in \overline{M(S_2)}s_1\,]\,\}$$
$$\cup \{\, s_1 \mid s_1 \in M(S_1)(\hat{\sigma},\hat{\alpha}) \wedge \sigma_1 = \perp\,\}$$

Note that $M(S_1;S_2)$ is prefix closed if S_1 and S_2 have a prefix closed semantics.

hiding

Hiding of internal communications just means the projection on external channels:

$$M([N])(\sigma,\alpha) = [\, M(N)(\sigma,\alpha)\,]_{chan([N])}$$

with for $U \in \mathbb{D}$ projection on a set *cset* is defined as follows:

$[U]_{cset} = \{([\tau]_{cset}, [W]_{cset}, [W^e]_{cset}, \sigma, \alpha) \mid (\tau,W,W^e,\sigma,\alpha) \in U\ \}$ where

$[\tau]_{cset}$ denotes the restriction of τ to records with channel name in *cset*, and

$[A]_{cset} = \{(l,u,D) \mid (l,u,D) \in A\ \wedge D \in cset\ \}$, for $A \subseteq WAIT$.

alternative

For the semantics of the *alternative* construction consider two cases:

- at least one of the purely boolean guards is true; then, because of priority for these branches, take the union of the semantics of all branches with a true purely boolean guard.
- none of the purely boolean guards is true:
  - then we take one of the open delay branches with minimal delay if there was no communication available for the open communication guards within this delay period. This last restriction is denoted by wait records for the channels of open i/o-guards, with interval length equal to the minimal delay period.
  - another possiblity is a communication before the minimal delay period has elapsed. Then we include the usual communication record and wait records for all open i/o-guards.

  Again the wait records of the environment are restricted in order to satisfy the maximal parallelism constraint.

$T_\sigma(A)$ represents the time needed to decide which i/o-branches are open, to compute delays, to select a branch, etc.

First define the extension of a function $X : STATE \times TIME \to \mathbb{D}$ to a set $U \in \mathbb{D}$, $X^* : \mathbb{D} \to \mathbb{D}$:

(remember the definition of $\overline{X} : \{ s \in \mathbb{B} \mid \sigma \neq \bot \} \to \mathbb{D}$ at sequential composition)

$$X^*(U) = \{ s \mid \exists s_u [s_u \in U \wedge \sigma_u \neq \bot \wedge s \in \overline{X} s_u] \}$$

$$\cup \{ s_u \mid s_u \in U \wedge \sigma_u = \bot \}.$$

Note that $M(S_1 ; S_2)(\hat{\sigma}, \hat{\alpha}) = M(S_2)^*(M(S_1)(\hat{\sigma}, \hat{\alpha}))$.

Let $A \equiv [\overset{n_1}{\underset{i=1}{\Box}} b_i \to S_i \overset{n_2}{\underset{i=1}{\Box}} b_i' ; DELAY\ d_i \to S_i' \overset{n_3}{\underset{i=1}{\Box}} b_i'' ; IO_i \to S_i'']$,

define

$mindelay = min\{ nonneg([\![d_i]\!]\sigma) \mid [\![b_i]\!]\sigma \}$ $\quad (min(\varnothing) = \infty)$

$ioset = \{ type(IO_i) \mid [\![b_i'']\!]\sigma \}$ and abbreviate

$\{(l, u, cset)\} = \{ (l, u, D) \mid D \in cset \}$.

$M(A)(\sigma, \alpha) =$

$$\bigcup_{i=1}^{n} \{ M(S_i)(\sigma, \alpha + t) \mid [\![b_i]\!]\sigma \wedge t \in T_\sigma(A) \}, \quad \text{if } \bigvee_{i=1}^{n_1} [\![b_i]\!]\sigma,$$

and otherwise

$$\bigcup_{i=1}^{n_2} M(S_i')^*(PFC\{(<>, \{(\alpha + t, \alpha + t + mindelay, ioset)\}, E, \sigma, \alpha + t + nonneg(t')) \mid [\![b_i']\!]\sigma \wedge$$

$$nonneg([\![d_i]\!]\sigma) = mindelay \wedge t \in T_\sigma(A) \wedge t' \in T_\sigma(DELAY\ d_i) \wedge$$

$$E \subseteq WAIT_{\alpha+t+nonneg(t')} \wedge MP(E,\{(\alpha+t\ ,\ \alpha+t+mindelay\ ,\ ioset\ )\})\ \})$$

$$\cup \bigcup_{i=1}^{n_3} M(S_i'')^{*}(PFC\{(<(\alpha+t+w\ ,\ D\ ,\ [\![e]\!]\sigma\ )>\ ,\ \{(\alpha+t\ ,\ \alpha+t+w\ ,\ ioset\ )\},$$

$$E\ ,\ \sigma\ ,\ \alpha+t+w+t')\mid [\![b_i'']\!]\sigma \ \wedge IO_i \equiv D!e \ \wedge$$

$$t \in T_\sigma(A\ ) \ \wedge t' \in T_\sigma(D!e\ ) \ \wedge w \in TIME \ \wedge 0 \leqslant w < mindelay \ \wedge$$

$$E \subseteq WAIT_{\alpha+t+w+t'} \ \wedge MP(E,\{(\alpha+t\ ,\ \alpha+t+w\ ,\ ioset\ )\})\ \})$$

$$\cup \bigcup_{i=1}^{n_3} M(S_i'')^{*}(PFC\{(<(\alpha+t+w\ ,\ D\ ,\ v\ )>\ ,\ \{(\alpha+t\ ,\ \alpha+t+w\ ,\ ioset\ )\},$$

$$E\ ,\ \sigma[{}^{v}/_{x}],\ \alpha+t+w+t')\mid [\![b_i'']\!]\sigma \ \wedge IO_i \equiv D?x \ \wedge v \in VAL \ \wedge$$

$$t \in T_\sigma(A\ ) \ \wedge t' \in T_\sigma(D?x\ ) \ \wedge w \in TIME \ \wedge 0 \leqslant w < mindelay \ \wedge$$

$$E \subseteq WAIT_{\alpha+t+w+t'} \ \wedge MP(E,\{(\alpha+t\ ,\ \alpha+t+w\ ,\ ioset\ )\})\ \})$$

iteration

The semantics of the *iteration* statement is defined as the limit of a chain of approximations. The extension of a function $X : STATE \times TIME \rightarrow \mathbb{D}$ to sets of tuples, $X^{*} : \mathbb{D} \rightarrow \mathbb{D}$, has been defined already at the alternative statement above. Then we define

$$M(*A\ )(\sigma\ ,\alpha) = \bigcup_{i=0}^{\infty} \phi_i\ (\sigma\ ,\alpha),$$

where ϕ_i are functions from $STATE \times TIME$ to $\mathbb{D}$ defined inductively by

$$\phi_0(\sigma\ ,\alpha) = \{(<>\ ,\varnothing\ ,\varnothing\ ,\perp\ ,\perp\ )\},$$

$$\phi_{i+1}(\sigma\ ,\alpha) =$$

$$\begin{cases} \phi_i^{*}(M(A\ )(\sigma\ ,\alpha)), & \text{if } \bigvee_{i=1}^{n_1} [\![b_i]\!]\sigma \ \vee \bigvee_{i=1}^{n_2} [\![b_i']\!]\sigma \ \vee \bigvee_{i=1}^{n_3} [\![b_i'']\!]\sigma \\ PFC\{(<>\ ,\varnothing\ ,E\ ,\sigma\ ,\alpha+t\ )\mid E \subseteq WAIT_{\alpha+t} \ \wedge t \in T_\sigma(A\ )\}, & \text{otherwise.} \end{cases}$$

An equivalent definition of the semantics of the *iteration* statement is given by the following fixed point equation.

$$M(^*A) = \mu X.\, \lambda_{\sigma,\alpha}.\ \text{if} \quad \bigvee_{i=1}^{n_1} [\![b_i]\!]\sigma \ \vee \bigvee_{i=1}^{n_2} [\![b_i']\!]\sigma \ \vee \bigvee_{i=1}^{n_3} [\![b_i'']\!]\sigma$$

$$\text{then } X^*(M(A)(\sigma,\alpha))$$

$$\text{else } PFC\{(<>,\emptyset,E,\sigma,\alpha+t) \mid E \subseteq WAIT_{\alpha+t} \ \wedge t \in T_\sigma(A)\},$$

with μ the least fixed point operator.

parallel composition

For the *parallel composition* $S_1 \parallel S_2$ the semantics includes:

- synchronized merge of the traces of both processes.
- union of sets of wait records. For the environment wait records we discharge the wait records concerning the joint channels. Note that the tuples of $M(S_1 \parallel S_2)$ satisfy the maximal parallelism constraint provided the tuples in $M(S_1)$ and $M(S_2)$ satisfy this constraint. The assumptions made by one process concerning the wait records of the environment must be fulfilled by the other process as far as it concerns joint communications.
- combination of the states. Remember that there are no shared variables.
- maximum of the time components.

Given that S_1 and S_2 have a prefix closed semantics we again obtain a prefix closed semantics for $S_1 \parallel S_2$.

Let $jchan = chan(S_1) \cap chan(S_2)$ and

define $max(\alpha_1,\alpha_2) = \perp$ if $\alpha_1 = \perp \ \vee \ \alpha_2 = \perp$.

$$M(S_1 \parallel S_2)(\hat{\sigma},\hat{\alpha}) = \{(\tau, W_1 \cup W_2, W_1^e \cup W_2^e - [\, W_1^e \cup W_2^e \,]_{jchan}, \sigma, max(\alpha_1,\alpha_2)) \mid$$

$$(\tau_i, W_i, W_i^e, \sigma_i, \alpha_i) \in M(S_i)(\hat{\sigma},\hat{\alpha}) \ \wedge i \in \{1,2\} \ \wedge$$

$$[\tau]_{chan(S_i)} = \tau_i \ \wedge (D \notin chan(S_1,S_2) \rightarrow [\tau]_D = <>) \ \wedge$$

$$time\text{-}ordered(\tau) \ \wedge$$

$$[W_1]_{jchan} = [W_2^e]_{jchan} \ \wedge [W_2]_{jchan} = [W_1^e]_{jchan} \ \wedge$$

$$(\sigma_1 \neq \perp \ \wedge \sigma_2 \neq \perp \rightarrow \sigma(x) = \begin{cases} \sigma_i(x) & , x \in var(S_i) \\ \hat{\sigma}(x) & , x \notin var(S_1,S_2)) \end{cases} \wedge$$

$$(\sigma_1 = \perp \ \vee \sigma_2 = \perp \rightarrow \sigma = \perp)\}$$

4. SPECIFICATION LANGUAGE

In this chapter our specification language is defined. First we give an informal introduction to correctness formulae in section 4.1. Section 4.2 lists the basic primitives of the assertion language, and the examples of section 4.3 should give an impression of the type of specifications intended. Section 4.4 contains the syntax of the assertion language. Restrictions on assertions are formulated in section 4.5. Section 4.6 concerns the formal interpretation of assertions, and finally in section 4.7 a formal definition of a correctness formula is given.

4.1 Correctness formulae

In this section the correctness formulae used in the proof system are introduced. Our aim is a compositional proof theory for safety properties, in which it is possible to specify the behaviour of a process relative to assumptions about the behaviour of its environment. Therefore we extend Hoare triples with two parts, an

- **assumption** specifying the expected communication behaviour of the environment (the waiting for a communication included), and a
- **commitment**, which is guaranteed to hold by the process itself, as long as the assumption concerning earlier behaviour has not been violated by the environment.

Important is that assumption and commitment reflect, respectively, the externally visible behaviours of environment and process. That is, they refer to a communication trace of externally visible channels and to wait records concerning these channels. Consequently, assumption and commitment must not contain program variables or internal channels. Clearly the assumption refers to *environment* wait records, whereas the commitment refers to wait records of the process itself. In addition we require that assumption and commitment do not refer to the time component.

We use the following notation: $(A,C):\{p\}\ L\ \{q\}$, meaning informally:
assume that p holds for the initial tuple (in $\mathbb{B}$) in which L starts executing, then:

(1) C holds for the initial tuple of L,

(2) C holds after every communication and wait action of L, provided A held after all communications and wait actions of L before this particular one,

(3) q holds for the final tuple if and when L terminates, provided A held after all communications and wait actions of L, up to and including the moment of termination.

Observe that the coupling between A and C is checked whenever the set of wait records or the trace of L changes. This is justified, since A and C do not refer to the program variables or to the time component. Furthermore, assertions are restricted (see section 4.5) such that their validity is not changed by adding *environment* wait records.

4.2 Assertion language

In this section we list the basic primitives of our assertion language which will be used in the examples of the next section. A complete syntax is given in section 4.4.

In our assertions it is possible to refer to the components of a tuple; to the
- trace of communication records by π,
- set of wait records by W,
- set of environment wait records by W^e,
- program variables,
- time component by means of the special variable *time*.

In the sequel assertions are restricted to those where π, W and W^e occur only *projected*, that is, within the scope of a projection $[...]_{cset}$;
$[\pi]_{cset}$ denotes the maximal subtrace of π with channel names in *cset*
(in the sequel denoted as π_{cset}).
$[W]_{cset}$ denotes the maximal subset of W with channel names in *cset*
(denoted as W_{cset}). Similar for $[W^e]_{cset}$.
We often omit brackets and commas in *cset*, e.g. W_D, π_{BD}.
The precise restrictions on the assertion language are formulated in section 4.5.

Because a trace is a sequence of records, we use an index to refer to a particular record, e.g. $\pi_B[i]$ refers to the i-th communication record in trace projection π_B.

Furthermore, we can select the fields of a communication record:
- *tim* selects the time stamp,
- *comm* selects the channel name, and
- *val* selects the communicated value.

|..| denotes the length of a trace expression.

4.3 Examples of specifications

The examples below should give an impression of the type of specifications intended.

ex. 1 Take the following T-function: $T(x:=x+1)=[3,4]$, $T(IO)=[1.5,3.5]$. Then

$$(TRUE,TRUE):\{time=v\}\ \mathbf{B?x;x:=x+1;D!x}\ \{time-v\in[6,11]\}$$

where v is a logical VAL variable (see next section).

□

Assume for the following examples:
$T(DELAY\ d)=[d,d]$, $T(IO)=[1,1]$, and $T(A)=[1,1]$.

ex. 2 Consider the following informal specification:

(env. waits for the first comm. via D from time 2 up to the actual comm., TRUE):
{execution starts at time 0 and the initial trace of channel D is empty}
DELAY 5 ; D!3
{termination at time 6}.

This can be expressed formally as follows:

$$(\pi_D\neq<>\rightarrow(2,tim(\pi_D[1]),D)\in W_D^e,TRUE):$$
$$\{time=0\ \wedge\ \pi_D=<>\}$$
DELAY 5 ; D!3
$$\{time=6\}.$$

□

ex. 3 The correctness formula below contains an informal assumption:

(the environment does not communicate via channel D in time interval [1,6], TRUE):

$$\{\pi_D=<>\ \wedge time=0\}$$
$$[\mathbf{DELAY\ 5}\rightarrow \mathbf{x:=5}\ \square\ \mathbf{D!3}\rightarrow \mathbf{x:=6}]$$
$$\{x=5\}.$$

This assumption can be formalised as follows: $|\pi_D|\geq 1\rightarrow tim(\pi_D[1])\notin[1,6]$.

□

ex. 4 This example demonstrates how two concurrent processes mutually make assumptions about the waiting period for a communication of the other. Consider assumption

$A_1\equiv(|\pi_D|\geq 1\rightarrow(2,tim(\pi_D[1]),D)\in W_D^e)\ \wedge\ (|\pi_D|\geq 2\rightarrow(13,tim(\pi_D[2]),D)\in W_D^e)$

and commitment

$C_1\equiv(|\pi_D|\geq 1\rightarrow(5,tim(\pi_D[1]),D)\in W_D)\ \wedge\ (|\pi_D|\geq 2\rightarrow(8,tim(\pi_D[2]),D)\in W_D)$

then

$$(A_1,C_1):\{\pi_D = <> \ \wedge time = 0\}\ \text{DELAY 5 ; D!3 ; DELAY 2 ; D!6}\ \{time = 14\}.$$

Note that the commitment C_1 of this process expresses that the waiting period for the second D-communication starts at time 8, which depends on the assumption in A_1 about when the first D-communication of the environment is enabled.
The second concurrent process has an "inverted" assumption/commitment pair,
let $A_2 \equiv C_1[{}^{W^e}/_{W}]$ and $C_2 \equiv A_1[{}^{W}/_{W^e}]$ then

$$(A_2,C_2):\{\pi_D = <> \ \wedge time = 0\}\ \text{DELAY 2 ; D?x ; DELAY 7 ; D?x}\ \{time = 14\}.$$

□

4.4 Syntax of the assertion language

In section 4.2 a number of basic primitives of the assertion language were presented. In this section the complete syntax is given.
In assertions we use logical variables to relate assumption, commitment, precondition and postcondition. These variables do not occur in the program text, so the value they denote is not affected by program execution. In order to apply correct substitutions distinguish between three types of logical variables:

- logical trace variables : t ,
- logical wait variables : w ,
- logical VAL variables : v .

Quantification is only allowed over logical variables.
In the following syntax of the assertion language we denote by *elt* an element of VAL, by D a channel name, by x a program variable, and *cset* denotes a set of channel names.

trace expressions :

$te \ ::= \ \pi \mid t \mid [te]_{cset}$

wait expressions :

$we ::= W \mid W^e \mid w \mid [we]_{cset}$

wait records :

$wr ::= (e_1,e_2,c)$

channels :

$c \ ::= \ D \mid comm(te[e])$

VAL expressions :

$e \ ::= elt \mid v \mid x \mid time \mid |te| \mid |we| \mid e_1+e_2 \mid val(te[e]) \mid tim(te[e])$

assertions :

$$p ::= true \mid te_1 = te_2 \mid we_1 = we_2 \mid c_1 = c_2 \mid e_1 = e_2 \mid c \in cset \mid wr \in we \mid$$
$$\neg p \mid p_1 \rightarrow p_2 \mid \exists v\,[p] \mid \exists t\,[p] \mid \exists w\,[p]$$

Let $var(p)$ be the set of program variables occurring in assertion p.
$chan(p)$ is defined as the set of channel names occurring in projections. Remember that we restrict us to assertions in which π, W and W^e only occur *projected*.

The following abbreviations are often used:
$\pi_{cset} \equiv [\pi]_{cset}$, $W_{cset} \equiv [W]_{cset}$, $W^e_{cset} \equiv [W^e]_{cset}$.
A lot of other abbreviations will be used which are expressible in the formal syntax as given above, e.g.

$(5,D,8) \in \pi_{cset} \equiv \exists v\,[time(\pi_{cset}[v]) = 5 \wedge comm(\pi_{cset}[v]) = D \wedge val(\pi_{cset}[v]) = 8]$.

To denote that a trace expression te_1 is an initial prefix of trace expression te_2, we use the abbreviation $te_1 \leqslant te_2$, defined as follows

$$|te_1| \leqslant |te_2| \wedge \forall v\,[v \leqslant |te_1| \rightarrow tim(te_1[v]) = tim(te_2[v]) \wedge$$
$$comm(te_1[v]) = comm(te_2[v]) \wedge$$
$$val(te_1[v]) = val(te_2[v])\,].$$

4.5 Restrictions on the assertion language

For a correctness formula $(A, C) : \{p\}\ L\ \{q\}$ the following restrictions are imposed upon the assertions A, C, p and q:

- $var(A, C) = \varnothing$; program variables must not occur in A and C, since A and C should express the communication interface only.

- W does not occur in A; an assumption must only mention the wait records of the environment and the trace.

- W^e does not occur in C; a commitment must only mention the wait records of the process itself and the trace.

- the special variable *time* does not occur in A and C.
  By imposing this constraint (and the first restriction), the validity of A and C depends on the trace and the wait records only, and not on the time component. Consequently, we have to check preservation of the validity of A and C, and their coupling, only after an occurrence of a communication or wait action, and not when merely time passes. Future research will investigate the consequences of allowing the special variable *time* to occur in A and C.

- π, W' and W'^e must occur *projected*, that is within the scope of a projection $[..]_{cset}$.

- W'^e is allowed in p and q, but all assertions must be monotone in W'^e:
  an assertion p is called *monotone in* W'^e iff
  $p \rightarrow \forall E \subseteq WAIT\ [\, p[{}^{W'^e \cup E}/_{W'^e}]\,]$.
  Also assumption A must be monotone in W'^e.
  ex. Examples of nonmonotone assertions are:
  $W'^e_D = \varnothing$, $\quad (5,7,B) \notin W'^e_B$, $\quad W'^e_D \subseteq \{(0,5,D),(9,10,D)\}$.
  The following assertions are monotone:
  $W'^e_D \neq \varnothing$, $\quad (5,7,B) \in W'^e_B$, $\quad W'^e_D \supseteq \{(0,5,D),(9,10,D)\}$.
  □

The last two restrictions are imposed because we aim at a *compositional* proof system, that is, the specification of a program should be verifiable in terms of the specifications of its syntactic subprograms. For the parallel composition rule the goal is, in principle, a simple conjunction of commitments, and similar for pre and post conditions. This is achieved by imposing maximal parallelism as a separate axiom on the domain of interpretations of assertions, and furthermore by the above mentioned constraints and restrictions in the proof system; assertions of a process remain valid under the execution of environment actions:
w.r.t. wait actions of the environment (because of monotonicity), and
w.r.t. communications of the environment (because of the use of projections and the restriction in the proof system that at parallel composition the assertions of one process do not refer to external channels of the other process).
(See [HdeR] for a comprehensive discussion of compositionality and how to achieve it by means of projections.)

4.6 Interpretation of assertions

This section concerns the interpretation of the assertion language.
An assertion p is interpreted in a logical variable environment γ, which assigns values to logical variables, and a tuple $s = (\tau, W', W'^e, \sigma, \alpha) \in \mathbb{B}$, notation: $[\![p]\!]\gamma s$.
If p contains free program variables ($var(p) \neq \varnothing$) or the special variable *time*, then p is only interpreted in tuples s with $\sigma \neq \bot$ and $\alpha \neq \bot$. The interpretation is straightforward, some examples:

$[\![t]\!]\gamma s = \gamma(t)$, $\quad [\![w]\!]\gamma s = \gamma(w)$, $\quad [\![v]\!]\gamma s = \gamma(v)$,

$[\![\pi]\!]\gamma s = \tau$, $\quad [\![[te]_{cset}]\!]\gamma s = [\,[\![te]\!]\gamma s\,]_{cset}$,

$[\![W']\!]\gamma s = W$, $\quad [\![[we]_{cset}]\!]\gamma s = [\,[\![we]\!]\gamma s\,]_{cset}$,

if $\sigma \neq \bot$ and $\alpha \neq \bot$ then $[\![x]\!]\gamma s = \sigma(x)$, $[\![time]\!]\gamma s = \alpha$.

An assertion p is called valid, denoted by $\models p$, iff

$\forall \gamma \, \forall s \in \mathcal{IB} \, [\sigma \neq \bot \ \wedge \alpha \neq \bot \rightarrow [\![p]\!]\gamma s]$.

4.7 Formal definition of a correctness formula

Finally we are able to give a formal definition of the interpretation of a correctness formula.

Again we use the abbreviation $s = (\tau, W, W^e, \sigma, \alpha)$, $\hat{s} = (\hat{\tau}, \hat{W}, \hat{W}^e, \hat{\sigma}, \hat{\alpha})$ etc.

The concatenation of two tuples, $s_1 s_2$, was defined in section 3.4 as follows:

$s_1 s_2 = (\tau_1 \tau_2, W_1 \cup W_2, W_1^e \cup W_2^e, \sigma_2, \alpha_2)$.

Also recall the extension of the semantic function to initial tuples:

$\overline{M(L)}\hat{s} = \{\hat{s} s \mid s \in M(L)(\hat{\alpha}, \hat{\sigma}) \wedge MP(\hat{W}, W^e) \wedge MP(\hat{W}^e, W)\}$.

Note that, by the explicit check on maximal parallelism, a tuple from $\mathcal{IB}$ is obtained.

For the formal interpretation of a correctness formula we need the $<$ relation on tuples, defined as:

$s' < s \ \rightleftarrows \ s' \leqslant s \ \wedge \ (\tau' \neq \tau \ \vee W' \neq W)$.

Notation: $\hat{s}_\perp = (\hat{\tau}, \hat{W}, \hat{W}^e, \perp, \perp)$.

Now a correctness formula is called valid, denoted by $\models (A, C) : \{p\} \, L \, \{q\}$, iff

$\forall \gamma \forall \hat{s} \in \mathcal{IB}, \hat{\sigma} \neq \bot, \hat{\alpha} \neq \bot \, [[\![p]\!]\gamma \hat{s} \rightarrow \forall s \in \overline{M(L)}\hat{s} \, [(\forall s' [\hat{s}_\perp \leqslant s' < s \rightarrow [\![A]\!]\gamma s'] \rightarrow [\![C]\!]\gamma s) \wedge$

$(\sigma \neq \bot \rightarrow (\forall s' [\hat{s}_\perp \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s'] \rightarrow [\![q]\!]\gamma s))]]$.

Observe that this formal definition of $(A, C) : \{p\} \, L \, \{q\}$ corresponds to the informal meaning of section 4.1, since $\overline{M(L)}$ is prefix closed and the definitions of prefix closed and "$<$" are such that the validity of C is checked whenever W or τ changes.

Furthermore C holds initially, because $(<>, \varnothing, \varnothing, \perp, \perp) \in M(L)(\hat{\sigma}, \hat{\alpha})$ (since $M(L)(\hat{\sigma}, \hat{\alpha})$ is prefix closed), and thus $(\hat{\tau}, \hat{W}, \hat{W}^e, \perp, \perp) \in \overline{M(L)}\hat{s}$ (since $MP(\hat{W}, \varnothing)$ and $MP(\hat{W}^e, \varnothing)$).

Then $\forall s' [\hat{s}_\perp \leqslant s' < (\hat{\tau}, \hat{W}, \hat{W}^e, \perp, \perp) \rightarrow [\![A]\!]\gamma \hat{s} s']$, because there is no s' such that $\hat{s}_\perp \leqslant s' < (\hat{\tau}, \hat{W}, \hat{W}^e, \perp, \perp)$.

Hence, by the formal definition above, $[\![C]\!]\gamma(\hat{\tau}, \hat{W}, \hat{W}^e, \perp, \perp) = [\![C]\!]\gamma \hat{s}$ has to hold (remember that C does not refer to the state or the time).

5. PROOF SYSTEM

Important in the proof system, which will be formulated in this chapter, is how we deal with maximal parallelism. In the assertion language the **maximal parallelism constraint** is formulated as follows:

(MP) $\forall v_1 v_2 v_3 v_4 [\ (v_1,v_2,D) \in W_D \wedge (v_3,v_4,D) \in W_D^e \rightarrow [v_1,v_2> \cap [v_3,v_4> = \varnothing\ ]$,

where D is a channel name, and v_1,v_2,v_3,v_4 are logical VAL variables.

Observe that it is allowed to use axiom scheme *MP* for every implication in the assertion language, since assertions are only interpreted in tuples from $\mathbb{B}$ (remember that every tuple in $\mathbb{B}$ satisfies the maximal parallelism constraint w.r.t. W and W^e (see chapter 3)).

Conclusion: Maximal parallelism is modeled as the axiom *MP* which is imposed on the domain of interpretation of assertions in our system. That is, this axiom can be used to prove implications between assertions, for instance, when applying the consequence rule.

The rules and axioms of our proof system are given in three groups. In section 5.1 the rules and axioms related to atomic statements of our language are presented. In section 5.2, those related to composite constructs, and in section 5.3 general axioms and rules related to all language constructions are given. In section 5.4 soundness of the system is stated (which is proved in the appendix). Section 5.5 contains an example demonstrating the use of assumptions and commitments in combination with parallel composition and hiding.

5.1 Rules and axioms for atomic statements

First we give rules and axioms for skip, assignment, delay and i/o-commands. These rules and axioms have in common that in order to prove $(A, C): \{p\}\ S\ \{q\}$ the implication $p \rightarrow C$ has to hold (C should hold initially). Following [ZRE84] these implications are avoided by proving $(A, C): \{p \wedge C\}\ S\ \{q \wedge C\}$.

For an arbitrary T-function the skip axiom would have the following form (note that *time* does not occur in C):

$$(A, C): \{\forall t \in T(SKIP)[\ q[^{time+t}/_{time}]\ ] \wedge C\}\ SKIP\ \{q \wedge C\}$$

In order to avoid explicit mentioning of the T-function in every rule of the proof system we take one specific T-function. It represents assumptions about the execution time which are similar to those in [KSRGA], where atomic actions take one time unit, except for the *DELAY d* statement which takes exactly d time units.

To be precise: in the sequel we adopt the following T-function:
$T(SKIP) = T(x := e) = T(D!e) = T(D?x) = T(A) = [1,1]$ (closed interval), and
$T(DELAY\ d) = [d, d]$.

skip

This leads to the following skip axiom:

(skip) $$(A, C): \{q[^{time+1}/_{time}] \wedge C\}\ SKIP\ \{q \wedge C\}$$

The assignment and delay axiom are similar to the skip axiom:

assignment

(assignment) $$(A, C): \{q[^{time+1}/_{time}, {}^{e}/_{x}] \wedge C\}\ x := e\ \{q \wedge C\}$$

delay

Remember that a negative delay value yields a zero delay, so the function *nonneg* is applied, which is defined as follows:

$$nonneg(v) = \begin{cases} 0 & \text{if } v < 0, \\ v & \text{if } v \geqslant 0. \end{cases}$$

(delay) $$(A, C): \{q[^{time+nonneg(d)}/_{time}] \wedge C\}\ DELAY\ d\ \{q \wedge C\}$$

output

For the output command we have to prove that given the precondition:

- commitment C holds for the final state (which is represented by the substitution), and
- the postcondition holds in the final state (also the time is updated), provided assumption A holds in the final state.

Note that in general we do not know the length of the waiting period for this communication, thus we have to prove commitment and postcondition for all possible wait values w.

Let $subst \equiv {}^{W \cup \{(time, time+w, D)\}}/_{W}, {}^{\pi \cup \{(time+w, D, e)\}}/_{\pi}$.

(output)

$$\frac{p \wedge C \rightarrow \forall w \in TIME.\, w \geqslant 0\,[\,C[subst] \wedge (A[subst] \rightarrow q[subst, {}^{time+w+1}/_{time}])\,]}{(A, C): \{p \wedge C\}\ D!e\ \{q \wedge C\}}$$

As observed above, it is allowed to use axiom *MP* for every implication between assertions. This will be used in the following example, where assumption A is strong enough to determine the waiting period.

ex. We want to prove the following formula:

$$(A \equiv \pi_D \neq <> \rightarrow (7,tim(\pi_D[1]),D) \in W'^{e}_{D}\ ,\ C \equiv TRUE):$$

$$\{p \equiv \pi_D = <> \wedge time = 4\}\ \mathbf{D!3}\ \{time = 8\}.$$

First take the following auxiliary postcondition:

$q \equiv (7,time-1,D) \in W'^{e}_{D} \wedge (4,time-1,D) \in W_D$.

By using the output rule we can prove $(A\ ,\ C): \{p\}\ \mathbf{D!e}\ \{q\}$,

since $p \wedge C \wedge A[subst] \rightarrow (7,time+w,D) \in W'^{e}_{D} \wedge time = 4$, and

$q[subst, {}^{time+w+1}/_{time}] \rightleftarrows$

$(7,time+w,D) \in W'^{e}_{D} \wedge (4,time+w,D) \in W_D \cup \{(time,time+w,D)\}$,

thus $p \wedge C \wedge A[subst] \rightarrow q[subst, {}^{time+w+1}/_{time}]$, for all $w \in TIME$, $w \geqslant 0$.

By using the maximal parallelism axiom MP for channel D we can derive from the post condition q:

$[7,time-1> \cap [4,time-1> = \varnothing$.

Since $l \leqslant u$ for a wait record $(l,u,....)$ we can derive: $time-1=7$, and thus: $time=8$.

Then the consequence rule, which will be formulated later, leads to the desired result.
□

input

The input rule has the same structure as the output rule. Since the received value is not known in general, we have to prove commitment and postcondition for all possible input values.

Let $subst \equiv {}^{W' \cup \{(time,time+w,D)\}}/_{W'}, {}^{\pi'(time+w,D,v)}/_{\pi}$.

(input)

$$\frac{p \wedge C \rightarrow \forall w \in TIME, w \geqslant 0\ \forall v \in VAL[\ C[subst] \wedge (A[subst] \rightarrow q[subst, {}^{time+w+1}/_{time}, {}^{v}/_{x}])\ ]}{(A\ ,\ C): \{p\ \wedge C\}\ D?x\ \{q\ \wedge C\}}$$

As we saw above, A may contain enough information to be more specific about the waiting period for the communication. In addition, A can specify the value that will be received by the input.

ex. Using the input rule, we can prove

$(\pi_D \leqslant <(..,D,5)>, TRUE): \{\pi_D = <>\}\ \mathbf{D?x}\ \{x = 5\}$.

Note that it is not allowed to use the assumption directly for the commitment. So we can not prove:

$(\pi_D \leqslant <(..,D,5)>, \pi_D \leqslant <(..,D,5)>): \{\pi_D = <>\}\ \mathbf{D?x}\ \{x = 5\}$.

The reason is that we have to avoid circular reasoning in assumptions and commitments, e.g. consider the following example:

using the assumption directly for the commitment, we could prove

$$(\pi_D \leqslant <(2,D,..)>, \pi_D \leqslant <(2,D,..)>) : \{\pi_D = <> \wedge time = 0\}\ \mathrm{D?x}\ \{time = 3\}$$

and

$$(\pi_D \leqslant <(2,D,..)>, \pi_D \leqslant <(2,D,..)>) : \{\pi_D = <> \wedge time = 0\}\ \mathrm{D!8}\ \{time = 3\}.$$

Clearly the assumption of one process is implied by the commitment of the other, so by the parallel composition rule, which will be given later, this would lead to:

$$(TRUE, TRUE) : \{\pi_D = <> \wedge time = 0\}\ \mathrm{D?x} \parallel \mathrm{D!8}\ \{time = 3\},$$

but this formula is not valid.

The work of Pandya [PJ] shows that it is possible to obtain a sound proof system in which for an *input* command the assumption may be used directly for a proof of the commitment. Circular reasoning is avoided by requiring for every *output* command a proof of the commitment without using the assumption.

□

5.2 Rules for composite constructs

Next we give rules for sequential composition, hiding, alternative, iteration and parallel composition. Since we give a compositional proof system, to each composite construct corresponds a rule in which a specification of the construct can be derived from its constituents without any further knowledge of the structure of these components (see [HdeR] for more details).

sequential composition

(sequential composition)
$$\frac{(A, C) : \{p\}\, S_1\, \{r\},\ (A, C) : \{r\}\, S_2\, \{q\}}{(A, C) : \{p\}\, S_1 ; S_2\, \{q\}}$$

hiding

The hiding rule allows us to encapsulate internal communications.

(hiding)
$$\frac{(A, C) : \{p \wedge \pi_{jchan} = <> \wedge W_{jchan} = \varnothing\}\, S_1 \parallel S_2\, \{q\}}{(A, C) : \{p\}\, [S_1 \parallel S_2]\, \{q\}}$$

where $jchan = chan(S_1) \cap chan(S_2)$, and provided $chan(A, C, p, q) \cap jchan = \varnothing$.

alternative

For the alternative construct we have two rules; a consequence of purely boolean guards having priority.

Let $A \equiv [\ \Box_{i=1}^{n_1} b_i \rightarrow S_i\ \Box_{i=1}^{n_2} b_i' ; DELAY\ d_i \rightarrow S_i'\ \Box_{i=1}^{n_3} b_i'' ; IO_i \rightarrow S_i''\ ]$.

The first rule is applied if one of the purely boolean guards evaluates to true.

Assertion $\bar{p}$ holds after evaluation of the purely boolean guards (which takes one time unit) and before execution of a S_i-branch.

(alt1)
$$\frac{p \rightarrow \bar{p}[^{time+1}/_{time}]\ ,\ (A\ ,\ C):\{\bar{p}\ \wedge b_i\}\ S_i\ \{q\}\ ,\ i=1,..,n_1}{(A\ ,C):\{p\ \wedge \bigvee_{i=1}^{n_1} b_i\}\ A\ \{q\}}$$

In the second rule none of the purely boolean guards is true. In order to define the minimal delay period and the set of "open" IO-guards, we have to know which booleans are true. So we have to guess the set of true boolean guards:
S is the set of indices of b_i' which are true,
T is the set of indices of b_i'' which are true.

Define for sets $S \subseteq \{1,...,n_2\}$ and $T \subseteq \{1,...,n_3\}$:
$mindelay = min\{\ nonneg(d_i) \mid i \in S\ \}, \qquad (min(\varnothing) = \infty)$
$ioset = \{\ type(IO_i) \mid i \in T\ \}$, and abbreviate
$\{(l\ ,u\ ,cset\ )\} = \{\ (l\ ,u\ ,D\ ) \mid D \in cset\ \}$.

Expression $B_{S,T}$ checks the guess, represented by S and T, for booleans:
$B_{S,T} \equiv \bigwedge_{k \in S} b_k' \ \wedge \bigwedge_{k \notin S} \neg\, b_k' \ \wedge \bigwedge_{k \in T} b_k'' \ \wedge \bigwedge_{k \notin T} \neg\, b_k''$.
For a wrong guess $B_{S,T}$ yields FALSE in the premiss of an implication in the rule, thus satisfying this implication trivially.

Assertion $\bar{p}$ holds after a *DELAY*'-guard and before a S_i'-branch,
assertion p_i holds after the IO_i-guard and before the S_i''-branch.

Let $subst_1 \equiv {}^{W' \cup \{(time+1,\,time+1+mindelay,\,ioset)\}}/_{W'}$

$subst_2 \equiv {}^{W' \cup \{(time+1,\,time+1+w,\,ioset)\}}/_{W'},\ {}^{\pi^\frown(time+1+w,D,e)}/_{\pi}$

$subst_3 \equiv {}^{W' \cup \{(time+1,\,time+1+w,\,ioset)\}}/_{W'},\ {}^{\pi^\frown(time+1+w,D,v)}/_{\pi}$.

(alt2) for all $S \subseteq \{1,..,n_2\}$, $T \subseteq \{1,..,n_3\}$:

$p \wedge C \rightarrow \neg \bigvee_{i=1}^{n_1} b_i$

$B_{S,T} \wedge p \wedge C \rightarrow C[subst_1] \wedge (A \rightarrow \bar{p}[subst_1, {}^{time+mindelay+1}/_{time}])$

$(A , C) : \{\bar{p} \wedge nonneg(d_i) = mindelay \wedge b_i'\} S_i' \{q\}$, $i = 1,..,n_2$

$B_{S,T} \wedge p \wedge C \rightarrow \forall w \in TIME\ [0 \leqslant w < mindelay \rightarrow C[subst_2] \wedge$

$(A[subst_2] \rightarrow p_i[subst_2, {}^{time+mindelay+2}/_{time}])]$ if $IO_i = D!e$, $i = 1,..,n_3$

$B_{S,T} \wedge p \wedge C \rightarrow \forall w \in TIME\ \forall v \in VAL\ [0 \leqslant w < mindelay \rightarrow C[subst_3] \wedge$

$(A[subst_3] \rightarrow p_i[subst_3, {}^{time+mindelay+2}/_{time}], {}^{v}/_{x}])]$ if $IO_i = D?x$, $i = 1,..,n_3$

$$\frac{(A , C) : \{p_i \wedge b_i''\} S_i'' \{q\} , \ i = 1,..,n_3}{(A , C) : \{p \wedge C\} A \{q \wedge C\}}$$

iteration

Define $b \equiv \bigvee_{i=1}^{n_1} b_i \vee \bigvee_{i=1}^{n_2} b_i' \vee \bigvee_{i=1}^{n_3} b_i''$, then

(iteration)
$$\frac{(A , C) : \{p \wedge b\} A \{p\} \qquad p \wedge \neg b \rightarrow q[{}^{time+1}/_{time}]}{(A , C) : \{p\} {}^{*}A \{q\}}$$

parallel composition

In [ZBR,ZRE84] the rule for parallel composition has the following form:
given specifications $(A_i , C_i) : \{p_i\} S_i \{q_i\}$ for both components,
choose a network assumption A for $S_1 \| S_2$, and check $A \wedge C_i \rightarrow A_j$,
for $(i , j) \in \{(1,2),(2,1)\}$.
This results in a specification $(A , C_1 \wedge C_2) : \{p_1 \wedge p_2\} S_1 \| S_2 \{q_1 \wedge q_2\}$,
provided certain restrictions on the assertions are met.
Typically, in $A \wedge C_i \rightarrow A_j$ assumptions concerning joint channels are verified; the remaining assumptions about external communications are maintained in A.
A straightforward adaptation of this rule is not possible; suppose we try a rule of the following form:

$$(A_i, C_i): \{p_i\}\ S_i\ \{q_i\},\quad i = 1,2$$

$$\ldots$$

$$A \wedge C_1[{}^{W'^e}/_{W'}] \rightarrow A_2$$

$$\frac{A \wedge C_2[{}^{W'^e}/_{W'}] \rightarrow A_1}{(A, \cdots): \{\cdots\}\ S_1 \| S_2\ \{\cdots\}}$$

Then consider the valid formula:

$$(A_1 \equiv (7, tim(\pi_D[1]), D) \in W_D^e,\ C_1 \equiv (7, tim(\pi_D[1]), D) \notin W_D):$$

$$\{\pi_D = <> \wedge W_D = \varnothing \wedge time = 0\}\ \mathbf{S}_1 \equiv \mathbf{D!3}\ \{time = 8\}.$$

If we take the parallel composition of S_1 and an other process S_2 with $D \notin chan(S_2)$, the new assumption of $S_1 \| S_2$ should be $A \equiv A_1$, since D is an external channel of this network. But then $A \wedge C_1[{}^{W'^e}/_{W'}] \rightarrow FALSE$ and we can prove every arbitrary assumption A_2 for S_2.

These problems can be avoided by taking care that the sets of channel names occurring in projections of W'^e in both components of the conjunction $A \wedge C_i[{}^{W'^e}/_{W'}]$ are disjoint.

Let $wchan(p)$ denote the set of channel names occurring in projections enclosing W' or W'^e in assertion p, and

let $jchan = chan(S_1) \cap chan(S_2)$.

For the network assumption A we require that W'^e does <u>not</u> occur projected on joint channels: $wchan(A) \cap jchan = \varnothing$.

Furthermore <u>project</u> the commitments C_1 and C_2 such that W' occurs only inside projections on joint channels in C_1 and C_2.

The <u>W-projection</u> of an assertion p on a set of channel names $cset$ is defined as follows:

$$p\,|^{W'} cset \equiv \exists w'\ [\ p[{}^{[W']_{cset} \cup [w']_{comp}}/_{W'}]\ ],$$

where $comp = chan(p) - cset$, (convention: $[w']_c = \varnothing$)

and w' a logical wait variable not occurring in p.

<u>ex.</u> $(7,8,D) \in W'_{BD}\ |^{W'}\ \{D\} \rightleftarrows$

$(7,8,D) \in [W']_{BD}\ |^{W'}\ \{D\} \rightleftarrows$

$\exists w'\ [(7,8,D) \in [[W']_D \cup [w']_B]_{BD}] \rightleftarrows$

$\exists w'\ [(7,8,D) \in [W']_D \cup [w']_B] \rightleftarrows$

$(7,8,D) \in W_D$

$(7,8,B) \in W'_{BD}\ |^{W'}\ \{D\} \rightleftarrows$

$\exists w'\ [(7,8,B) \in [W']_D \cup [w']_B] \rightleftarrows$

$\exists w'\ [(7,8,B) \in [w']_B] \rightleftarrows$

TRUE

□

(parallel composition)

$$(A_i, C_i): \{p_i \wedge W_{jchan} = \varnothing\}\ S_i\ \{q_i\},\quad i = 1,2$$

$$q_1[v_1/time, w_1/w, w_3/w^e] \wedge q_2[v_2/time, w_2/w, w_4/w^e] \wedge time = max(v_1, v_2) \wedge$$

$$W = w_1 \cup w_2 \wedge W^e = w_3 \cup w_4 - [w_3 \cup w_4]_{jchan} \rightarrow q$$

$$C_1[w_1/w] \wedge C_2[w_2/w] \wedge W = w_1 \cup w_2 \rightarrow C$$

$$(C_1 |^W jchan)[w^e/w] \wedge A \rightarrow A_2$$

$$(C_2 |^W jchan)[w^e/w] \wedge A \rightarrow A_1$$

$$\overline{(A, C): \{p_1 \wedge p_2 \wedge W_{jchan} = \varnothing\}\ S_1 \| S_2\ \{q\}}$$

with v_1, v_2, w_1, w_2, w_3 and w_4 logical VAL variables not occurring free in C or q, and provided
$wchan(A) \cap jchan = \varnothing$,
$chan(p_i, q_i, A_i, C_i) \cap chan(S_j) \subseteq chan(S_i)$, and
$var(p_i, q_i) \cap var(S_j) = \varnothing$, for $(i, j) \in \{(1,2),(2,1)\}$.

The last two restrictions denote that assertions of one process are not allowed to refer to program variables or external channels of the other process.

The clause $W_{jchan} = \varnothing$ in the precondition is necessary as the following example shows.
ex. Consider the correctness formula

$$((0,3,D) \in W_D^e, TRUE): \{\pi_D = <> \wedge time = 2\}\ \mathbf{D!7}\ \{\pi_D = <(2, D, 7)>\}.$$

which is valid (to derive it use the output rule with postcondition
$\pi_D = <(time - 1, D, 7)> \wedge (2, time - 1, D) \in W_D \wedge (0,3,D) \in W_D^e$) and also

$$(TRUE, (0,3,D) \in W_D): \{(0,3,D) \in W_D\}\ \mathbf{DELAY\ 5\ ;\ D?x}\ \{TRUE\}.$$

Without the above mentioned clause the rule would lead to (take $A \equiv TRUE$)

$$(TRUE, TRUE): \{\pi_D = <> \wedge time = 2 \wedge (0,3,D) \in W_D\}\ \mathbf{D!7} \| \mathbf{DELAY\ 5; D?x}\ \{\pi_D = <(2, D, 7)>\}$$

which is not true. So this would lead to an unsound rule.

□

5.3 General rules and axioms

The following rules and axioms are applicable to every language construction. The **consequence** rule is a straightforward extension of the usual rule in Hoare logic: one can strengthen the assumption and weaken the commitment.

(consequence) $$(A', C'):\{p'\}\ L\ \{q'\}$$

$$\frac{A \rightarrow A',\ C' \rightarrow C,\ p \rightarrow p',\ q' \rightarrow q}{(A, C):\{p\}\ L\ \{q\}}$$

As already observed, it is allowed to use maximal parallelism in the form of axiom *MP* for every implication between assertions.

(substitution) $$\frac{(A, C):\{p\}\ L\ \{q\}}{(A, C):\{p[^{e}/_{v}, ^{f}/_{t}, ^{g}/_{w}]\}\ L\ \{q\}}$$

where v, t and w are a logical VAL variable, a logical trace variable and a logical wait variable, resp., and where e, f and g are an arbitrary VAL expression, trace expression and wait expression, resp., and provided v, t and w do not occur free in A, C or q.

(conjunction) $$\frac{(A_i, C_i):\{p_i\}\ L\ \{q_i\},\quad i = 1,2}{(A_1 \wedge A_2, C_1 \wedge C_2):\{p_1 \wedge p_2\}\ L\ \{q_1 \wedge q_2\}}$$

(invariance) $$(A, C):\{p \wedge C\}\ L\ \{p \wedge C\}$$

provided $var(p) \cap var(L) = \varnothing$ and $chan(A, C, p) \cap chan(L) = \varnothing$.

According to [ZRE84] the following two axioms are needed for relative completeness of the proof system.

(prefix invariance) $$(TRUE, \pi_{cset} \geqslant t):\{\pi_{cset} = t\}\ L\ \{\pi_{cset} \geqslant t\}$$

where $cset = chan(L)$, and t is a trace variable.

(strengthen) $$(A, \forall t\ [\, t_0 \leqslant t < \pi_{cset} \rightarrow A[^{t}/_{\pi}]\,]):\{\pi_{cset} = t_0\}\ L\ \{A\}$$

where $cset = chan(L)$, t a trace variable not occurring free in A, and provided $chan(A) \subseteq cset$ and $wchan(A) = \varnothing$.

5.4 Soundness

Soundness of the proof system above is stated in the following theorem.

Theorem:

All rules and axioms of the above given proof system are sound w.r.t. the given semantics.

See the appendix for a proof of this theorem.

5.5 Example

The following example demonstrates the use of assumptions and commitments in combination with parallel composition and hiding. Consider the following processes:

$S_{11} \equiv D_1?x\ ; x := x+1\ ; B!x$

$S_{12} \equiv D_2?y_1\ ; y_1 := y_1+1\ ; B?y_2\ ; y_2 := y_2+y_1\ ; F!y_2$

$S_2 \equiv [D_1!0 \parallel D_2!0]\ ; [F?z \rightarrow SKIP$
$\qquad\qquad \Box\ DELAY\ 5 \rightarrow error := 1\ ].$

Suppose we want to prove

$(TRUE, TRUE) : \{error = 0\}\ [\ [\mathbf{S}_{11} \parallel \mathbf{S}_{12}] \parallel \mathbf{S}_2\ ]\ \{error = 0 \wedge z = 2\}.$

For an easy formulation we define the following predicates:
"*env waits for D from v* " denotes the assumption that the environment waits for the first communication via channel D starting at point of time v.
"*env waits for D from v and sends* $\bar{v}$" expresses the waiting for the first communication along channel D starting at point of time v, until the communication takes place with communicated value $\bar{v}$.
"*wait for D from v* " and "*wait for D from v and send* $\bar{v}$" express similar commitments for the process itself. Formal definitions:

env waits for D from v $\equiv \pi_D \neq <> \rightarrow (v, tim(\pi_D[1]), D) \in W_D^{\varepsilon}$

env waits for D from v and sends $\bar{v}$
$\equiv \pi_D \neq <> \rightarrow (v, tim(\pi_D[1]), D) \in W_D^{\varepsilon} \wedge val(\pi_D[1]) = \bar{v}$

wait for D from v $\equiv \pi_D \neq <> \rightarrow (v, tim(\pi_D[1]), D) \in W_D$

wait for D from v and send $\bar{v}$
$\equiv \pi_D \neq <> \rightarrow (v, tim(\pi_D[1]), D) \in W_D \wedge val(\pi_D[1]) = \bar{v}.$

Note: *wait for D from v and send* $\bar{v}$ $\equiv$ (*wait for D from v and receive* $\bar{v}$) $[^{W}/_{W^{\varepsilon}}]$.

We use abbreviates like "*wait for* D_1, D_2 *from v* " instead of "*wait for* D_1 *from v* $\wedge$ *wait for* D_2 *from v* ".

Then we can prove:

$$(env\ waits\ for\ D_1\ from\ v\ and\ sends\ 0,$$
$$wait\ for\ D_1\ from\ v\ \wedge\ wait\ for\ B\ from\ v+2\ and\ send\ 1):$$
$$\{time = v\ \wedge\ \pi_B = <>\ \wedge\ \pi_{D_1} = <>\}\ \mathbf{S}_{11}\ \{TRUE\}$$

and

$(env\ waits\ for\ D_2\ from\ v\ and\ sends\ 0 \wedge env\ waits\ for\ B\ from\ v+2\ and\ sends\ 1,$
$$wait\ for\ D_2\ from\ v\ \wedge\ wait\ for\ F\ from\ v+4\ and\ send\ 2):$$
$$\{time = v\ \wedge\ \pi_B = <>\ \wedge\ \pi_{FD_2} = <>\}\ \mathbf{S}_{12}\ \{TRUE\}.$$

With the parallel composition rule (and the consequence rule for the precondition) this leads to (observe that the commitment of S_{11} about channel B justifies the assumption of S_{12} about this channel)

$(env\ waits\ for\ D_1, D_2\ from\ v\ and\ sends\ 0,$
$$wait\ for\ D_1, D_2\ from\ v\ \wedge\ wait\ for\ F\ from\ v+4\ and\ send\ 2):$$
$$\{time = v\ \wedge\ \pi_{D_1D_2F} = <>\ \wedge\ \pi_B = <>\}\ \mathbf{S}_{11} \parallel \mathbf{S}_{12}\ \{TRUE\}.$$

Hiding of joint channel B gives

$(env\ waits\ for\ D_1, D_2\ from\ v\ and\ sends\ 0,$
$$wait\ for\ D_1, D_2\ from\ v\ \wedge\ wait\ for\ F\ from\ v+4\ and\ send\ 2):$$
$$\{time = v\ \wedge\ \pi_{D_1D_2F} = <>\}\ [\mathbf{S}_{11} \parallel \mathbf{S}_{12}]\ \{TRUE\}.$$

For S_2 we can prove:

$(env\ waits\ for\ D_1, D_2\ from\ v\ \wedge\ env\ waits\ for\ F\ from\ v+2+\bar{v}\ and\ sends\ 2 \wedge \bar{v} < 5,$
$$wait\ for\ D_1, D_2\ from\ v\ and\ send\ 0):$$
$$\{time = v\ \wedge\ \pi_{D_1D_2F} = <>\ \wedge\ error = 0\}\ \mathbf{S}_2\ \{error = 0 \wedge z = 2\}.$$

Using the parallel composition rule we obtain

$$(TRUE, TRUE):\{time = v\ \wedge\ error = 0\ \wedge\ \pi_{D_1D_2F} = <>\}\ [\mathbf{S}_{11} \parallel \mathbf{S}_{12}] \parallel \mathbf{S}_2\ \{error = 0 \wedge z = 2\}.$$

Applying the hiding rule leads to

$$(TRUE, TRUE):\{time = v\ \wedge\ error = 0\}\ [[\mathbf{S}_{11} \parallel \mathbf{S}_{12}] \parallel \mathbf{S}_2]\ \{error = 0 \wedge z = 2\}.$$

Using the substitution rule, with substitution $[^{time}/_v]$ in the precondition, we obtain the desired result:

$$(TRUE, TRUE):\{error = 0\}\ [[\mathbf{S}_{11} \parallel \mathbf{S}_{12}] \parallel \mathbf{S}_2]\ \{error = 0 \wedge z = 2\}.$$

□

6. CONCLUSION AND FUTURE WORK

A compositional proof system has been formulated for a real-time programming language for distributed computing with communication via synchronous message passing. In this proof system it is possible to specify assumptions about the expected behaviour of the environment of a process, and to formulate a commitment concerning the behaviour of the process itself, relative to these assumptions. Assumption and commitment are expressed as invariants, thus allowing the specification of nonterminating processes. Maximal parallelism is modeled by imposing a separate axiom on the domain of interpretation of assertions.

An essential restriction on the assertion language is that the special variable *time* must not occur in assumption or commitment. The next step in our research is to drop this restriction. Thereafter we hope to investigate the relation with liveness properties, in view of Lamport's statement ([La]) that real-time properties can be expressed as safety properties. Also the relation with real-time temporal logic ([KR],[KVR]) will be subject of future research.

Another interesting topic concerns changing the computation model: instead of maximal parallelism, where every process has its own processor, we plan to study the real-time behaviour resulting from the implementation of all processes on one processor together with some scheduling policy. For this seems to be what current practice in real-time programming is about - at least within our Esprit project. Also other communication primitives, such as asynchronous communication and broadcast, will be studied.

We expect the extension of the proof system with recursion in the style of [ZRE] to be straightforward. Finally relative completeness of the proof system is expected to proceed along the same lines as the relative completeness proof of the [ZRE]-system; it will be considered as soon as [Z] becomes available.

A. APPENDIX

A.1 Soundness of the proof system

In this appendix soundness of the proof system, as formulated in chapter 5, is proven. That is, every correctness formula $(A, C):\{p\}\, L\, \{q\}$ which is derivable in the proof system of chapter 5 is also valid: $\models (A, C):\{p\}\, L\, \{q\}$, as defined in section 4.7, using the semantics of chapter 3.
This is achieved by proving the following theorem.

Theorem:

All rules and axioms of the proof system of chapter 5 are sound w.r.t. the semantics given in chapter 3.

proof:

We have to prove that every axiom is valid, and that the conclusion of a rule is valid given the validity of the premisses.
Proving $\models (A, C):\{p\}\, L\, \{q\}$ is by the definition of section 4.7 equivalent to the following:
(again s denotes the tuple $(\tau, W, W^e, \sigma, \alpha)$, $\hat{s} = (\hat{\tau}, \hat{W}, \hat{W}^e, \hat{\sigma}, \hat{\alpha})$, etc, and $s_{\_} = (\tau, W, W^e, \bot, \bot)$)

given γ, $\hat{s} \in \mathbb{B}$, $\hat{\sigma} \neq \bot$, $\hat{\alpha} \neq \bot$ with $[\![p]\!]\gamma\hat{s}$, prove two parts for $s \in \overline{M(L)}\hat{s}$:

i) $\forall s'[\, \hat{s}_{\bot} \leqslant s' < s \rightarrow [\![A]\!]\gamma s'\,] \rightarrow [\![C]\!]\gamma s$, and

ii) $\sigma \neq \bot \rightarrow (\forall s'[\, \hat{s}_{\bot} \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s'\,] \rightarrow [\![q]\!]\gamma s)$.

Observe that all assertions are interpreted in tuples from $\mathbb{B}$, because the initial tuple $\hat{s}$ is in $\mathbb{B}$, and if $s \in \overline{M(L)}\hat{s}$ then $s \in \mathbb{B}$ (see the definitions of M and $\overline{M(L)}$ in chapter 3). Also note that if $s \in \mathbb{B}$, and $s' \leqslant s$ or $s' < s$, then $s' \in \mathbb{B}$.
Remember that $\mathbb{B}$ consists of all tuples $(\tau, W, W^e, \sigma, \alpha)$ with $MP(W, W^e)$, so the following axiom MP holds in every tuple from $\mathbb{B}$:

(MP) $\forall \nu_1 \nu_2 \nu_3 \nu_4 [\ (\nu_1, \nu_2, D) \in W_D \wedge (\nu_3, \nu_4, D) \in W_D^e \rightarrow [\nu_1, \nu_2> \cap [\nu_3, \nu_4> = \varnothing\ ]$,

where D is a channel name, and $\nu_1, \nu_2, \nu_3, \nu_4$ are logical VAL variables.
Thus it is allowed to use this axiom for every implication between assertions.

Note that, because of the restrictions on the assertion language (see section 4.5), $[\![C]\!]\gamma(\tau,W,W^e,\sigma,\alpha)$ depends on τ and W only, so we often write $[\![C]\!]\gamma(\tau,W,...,...,...)$. Similar, often $[\![A]\!]\gamma(\tau,...,W^e,...,...)$ is used.

skip

We have to prove:

$\models (A, C):\{q[^{time+1}/_{time}] \wedge C\}\ SKIP\ \{q \wedge C\}$.

proof:

Choose γ, $\hat{s} \in I\!B$, $\hat{\sigma} \neq \perp$, and $\hat{\alpha} \neq \perp$ arbitrary. Assume

(1) $[\![q[^{time+1}/_{time}] \wedge C]\!]\gamma\hat{s}$.

Let $s \in \overline{M(SKIP)}\,\hat{s}$, then $\tau = \hat{\tau}$ and $W = \hat{W}$.

i) C depends on trace τ and set of wait records W only, thus $[\![C]\!]\gamma s = [\![C]\!]\gamma\hat{s}$, which is valid by (1).

ii) If $\sigma \neq \perp$ then according to the definition of $M(SKIP)$:
$\sigma = \hat{\sigma}$, $\alpha = \hat{\alpha}+1$, and $W^e = \hat{W}^e \cup E$ with $E \subseteq WAIT_{\hat{\alpha}+1}$.
Hence, from (1),
$[\![q[^{time+1}/_{time}] \wedge C]\!]\gamma(\hat{\tau},\hat{W},\hat{W}^e,\hat{\sigma},\hat{\alpha}) =$
$[\![q \wedge C]\!]\gamma(\hat{\tau},\hat{W},\hat{W}^e,\hat{\sigma},\hat{\alpha}+1) =$
$[\![q \wedge C]\!]\gamma(\tau,W,\hat{W}^e,\sigma,\alpha)$ (remember $\tau=\hat{\tau}$ and $W=\hat{W}$).
Since q is *monotone* and W^e does not occur in C, we can add E to the environment wait records:
$[\![q \wedge C]\!]\gamma(\tau,W,\hat{W}^e \cup E,\sigma,\alpha) =$
$[\![q \wedge C]\!]\gamma(\tau,W,W^e,\sigma,\alpha) = [\![q \wedge C]\!]\gamma s$.

□

Soundness of the **assignment** and the **delay** axiom requires a similar proof.

input

Let $subst \equiv {}^{W \cup \{(time, time+w, D)\}}/_{W},\ {}^{\pi^{\cdot}(time+w, D, v)}/_{\pi}$.

Assume

(1) $\models p \wedge C \rightarrow$
$\forall w \in TIME, w \geq 0\ \forall v \in VAL\ [\, C[subst] \wedge (A[subst] \rightarrow q[subst, {}^{time+w+1}/_{time}, {}^{v}/_{x}])\,]$.

We have to prove: $\models (A, C):\{p \wedge C\}\ D?x\ \{q \wedge C\}$.

proof:

Choose γ, $\hat{s} \in I\!B$, $\hat{\sigma} \neq \perp$, and $\hat{\alpha} \neq \perp$ arbitrary. Assume

(2) $[\![p \wedge C]\!]\gamma\hat{s}$.

Let $s \in \overline{M(D?x)}\,\hat{s}$.
Then, by definition of $M(D?x)$, there exists a tuple
$\bar{s} = (\ <(\hat{\alpha}+w\ ,D\ ,v\ )>\ ,\ \{(\hat{\alpha}\ ,\hat{\alpha}+w\ ,D\ )\},\ E\ ,\ \sigma\ ,\ \hat{\alpha}+w+1\ )$
with $w \in TIME$, $v \in VAL$, $E \subseteq WAIT_{\hat{\alpha}+w+1}$, and $MP(E\ ,\{(\hat{\alpha}\ ,\hat{\alpha}+w\ ,D\ )\})$,
such that $\hat{s}_{\perp} \leqslant s \leqslant \hat{s}\,\bar{s}$.
From the definition of $\leqslant$ (see section 3.2) this implies
$s = \hat{s}_{\perp}$, or $s = \hat{s}\,\bar{s}_{\perp}$, or $s = \hat{s}\,\bar{s}$.

i) Assume $\forall s'[\,\hat{s}_{\perp} \leqslant s' < s \rightarrow [\![A]\!]\gamma s'\,]$ and prove $[\![C]\!]\gamma s$ as follows:

- if $\tau = \hat{\tau}$ and $W = \hat{W}$ then $[\![C]\!]\gamma s = [\![C]\!]\gamma\hat{s}$, which is valid by (2).

- if $\tau \neq \hat{\tau}$ or $W \neq \hat{W}$ then $\tau = \hat{\tau}\,\hat{}\,<(\hat{\alpha}+w\ ,D\ ,v\ )>$ and $W = \hat{W} \cup \{(\hat{\alpha}\ ,\hat{\alpha}+w\ ,D\ )\}$.
  So $[\![C]\!]\gamma s = [\![C]\!]\gamma(\ \hat{\tau}\,\hat{}\,(\hat{\alpha}+w\ ,D\ ,v\ )\ ,\hat{W} \cup \{(\hat{\alpha}\ ,\hat{\alpha}+w\ ,D\ )\}\ ,...,...,...)$
  $= [\![C\,[subst\,]]\!]\gamma(\hat{\tau}\ ,\hat{W}\ ,...,...,...)$
  $= [\![C\,[subst\,]]\!]\gamma\hat{s}\ ,$
  which is valid by (1) and (2).

ii) Assume $\sigma \neq \perp$ and
(3) $\forall s'[\,\hat{s}_{\perp} \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s'\,]$.
Prove $[\![q\ \wedge C]\!]\gamma s$ as follows.
First note that in this case $s = \hat{s}\,\bar{s}$.
Furthermore, remember that p is *monotone* in W^e, and W^e does not occur in C, so from (2) we can derive, by adding E to the set of environment wait records:
(4) $[\![p\ \wedge C]\!]\gamma(\hat{\tau}\ ,\hat{W}\ ,\hat{W}^e \cup E\ ,\hat{\sigma}\ ,\hat{\alpha})$.
Now (3) implies
$[\![A]\!]\gamma s = [\![A]\!]\gamma\hat{s}\,\bar{s} =$
$[\![A]\!]\gamma(\hat{\tau}\,\hat{}\,(\hat{\alpha}+w\ ,D\ ,v\ )\ ,\hat{W} \cup \{(\hat{\alpha}\ ,\hat{\alpha}+w\ ,D\ )\}\ ,\hat{W}^e \cup E\ ,\hat{\sigma}\ ,\hat{\alpha}+w+1)$.
Thus $[\![A\,[subst\,]]\!]\gamma(\hat{\tau}\ ,\hat{W}\ ,\hat{W}^e \cup E\ ,\hat{\sigma}\ ,\hat{\alpha}\ )$.
Together with (4) this leads, by using (1), to $[\![q\,[subst\,]]\!]\gamma(\hat{\tau}\ ,\hat{W}\ ,\hat{W}^e \cup E\ ,\hat{\sigma}\ ,\hat{\alpha})$.
Hence $[\![q]\!]\gamma(\hat{\tau}\,\hat{}\,(\hat{\alpha}+w\ ,D\ ,v\ )\ ,\hat{W} \cup \{(\hat{\alpha}\ ,\hat{\alpha}+w\ ,D\ )\}\ ,\hat{W}^e \cup E\ ,\hat{\sigma}\ ,\hat{\alpha}+w+1) = [\![q]\!]\gamma s$.
In part i) we already proved $[\![C]\!]\gamma s$, so $[\![q\ \wedge C]\!]\gamma s$.

□

Soundness of the **output** rule proceeds along the same lines.

sequential composition
Let
(1) $\models (A\ ,C\ ):\{p\}\,S_1\,\{r\}$, and

(2) $\models (A, C): \{r\} S_2 \{q\}$.
Prove $(A, C): \{p\} S_1; S_2 \{q\}$.
proof:
Choose γ, $\hat{s} \in \mathbb{B}$, $\hat{\sigma} \neq \perp$, and $\hat{\alpha} \neq \perp$ arbitrary. Assume
(3) $[\![p]\!]\gamma\hat{s}$.

Let $s \in \overline{M(S_1; S_2)}\hat{s}$. Note that

$$\overline{M(S_1; S_2)}\hat{s} = \{s \mid \exists s_1 [s_1 \in \overline{M(S_1)}\hat{s} \wedge \sigma_1 \neq \perp \wedge s \in \overline{M(S_2)}s_1\}$$
$$\cup \{s \mid s \in \overline{M(S_1)}\hat{s} \wedge \sigma = \perp\}.$$

So there are two possibilities:

> $s \in \overline{M(S_1)}\hat{s}$ and $\sigma = \perp$. Then

i) $\forall s' [\hat{s}_\perp \leqslant s' < s \rightarrow [\![A]\!]\gamma s'] \rightarrow [\![C]\!]\gamma s$, follows from (1), since (3) holds.

ii) $\sigma \neq \perp \rightarrow (\forall s' [\hat{s}_\perp \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s'] \rightarrow [\![q]\!]\gamma s)$, because $\sigma = \perp$.

> there exists an s_1 with $s_1 \in \overline{M(S_1)}\hat{s}$, $\sigma_1 \neq \perp$, and $s \in \overline{M(S_2)}s_1$.
Again consider two cases:

>> $\tau = \tau_1$ and $W' = W'_1$.

i) If we assume
(4) $\forall s' [\hat{s}_\perp \leqslant s' < s \rightarrow [\![A]\!]\gamma s']$,
then certainly $\forall s' [\hat{s}_\perp \leqslant s' < s_1 \rightarrow [\![A]\!]\gamma s']$, since $s' < s_1 \rightarrow s' < s$,
thus by using (1): $[\![C]\!]\gamma s_1$.
So $[\![C]\!]\gamma s$, since $\tau = \tau_1$ and $W'_2 = W'_1$.

ii) Assume $\sigma \neq \perp$ and
(5) $\forall s' [\hat{s}_\perp \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s']$,
then certainly $\forall s' [\hat{s}_\perp \leqslant s' \leqslant s_1 \rightarrow [\![A]\!]\gamma s']$.
In this case $\sigma_1 \neq \perp$ and $s_1 \in \overline{M(S_1)}\hat{s}$, so we obtain by (1): $[\![r]\!]\gamma s_1$.
From (5): $\forall s' [(s_1)_\perp \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s']$.
Since $\sigma \neq \perp$ and $s \in \overline{M(S_2)}s_1$, this leads by (2) to: $[\![q]\!]\gamma s$.

>> $\tau \neq \tau_1$ or $W \neq W_1$.

i) Assume
(6) $\forall s' [\hat{s}_\perp \leqslant s' < s \rightarrow [\![A]\!]\gamma s']$.

Since $s_1 \leqslant s$, we can deduce in this case that $s_1 < s$.
So, if $s' \leqslant s_1$, then also $s' < s$.
Consequently, we obtain from (6) $\forall s' [\hat{s}_\perp \leqslant s' \leqslant s_1 \rightarrow [\![A]\!]\gamma s']$.
So (1) leads to (remember $\sigma_1 \neq \perp$): $[\![r]\!]\gamma s_1$.
Now (6) implies $\forall s' [(s_1)_\perp \leqslant s' < s \rightarrow [\![A]\!]\gamma s']$,
thus, using (2) and $s \in \overline{M(S_2)}s_1$, we derive $[\![C]\!]\gamma s$.

ii) Assume $\sigma \neq \perp$, and $\forall s' [\hat{s}_\perp \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s']$.
Then similar to part i) we infer $[\![r]\!]\gamma s_1$, and
$\forall s' [(s_1)_\perp \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s']$.
Thus, by using (2), we obtain $[\![q]\!]\gamma s$.

□

hiding
Assume
(1) $\models (A, C): \{p \wedge \pi_{jchan} = <> \wedge W_{jchan} = \varnothing\} S_1 \| S_2 \{q\}$,
where $jchan = chan(S_1) \cap chan(S_2)$, and provided
(2) $chan(A, C, p, q) \cap jchan = \varnothing$.
We have to prove:
$\models (A, C): \{p\} [S_1 \| S_2] \{q\}$.
proof:
Choose γ, $\hat{s} \in \mathbb{B}$, $\hat{\sigma} \neq \perp$, and $\hat{\alpha} \neq \perp$ arbitrary. Assume
(3) $[\![p]\!]\gamma \hat{s}$.
Let $s \in \overline{M([S_1 \| S_2])}\hat{s}$.
Then there exists an $\bar{s}$ with $s = \hat{s}[\bar{s}]_{extchan}$,
(4) $\bar{s} \in M(S_1 \| S_2)(\hat{\sigma}, \hat{\alpha})$, and
(5) $MP(\hat{W}, [\overline{W}^e]_{extchan}) \wedge MP([\overline{W}]_{extchan}, \hat{W}^e)$,
where $extchan = chan([S_1 \| S_2])$.

Define the projection of a tuple s on a set of channel names $cset$, notation $[s]_{cset}$, as follows:
$[s]_{cset} = ([\tau]_{cset}, [W]_{cset}, [W^e]_{cset}, \sigma, \alpha)$.
Furthermore $s \backslash cset$ denotes the tuple obtained from s by deleting all records (in τ, W and W^e) with channel name in $cset$.

By using $chan(p) \cap jchan = \varnothing$ (see (2)),
we can deduce from (3): $[\![p]\!]\gamma \hat{s} \backslash jchan$.
Observe that $[\![\pi_{jchan} = <> \wedge W_{jchan} = \varnothing]\!]\gamma \hat{s} \backslash jchan$, so
(6) $[\![p \wedge \pi_{jchan} = <> \wedge W_{jchan} = \varnothing]\!]\gamma \hat{s} \backslash jchan$.

i) Assume

(7) $\forall s'[\hat{s}_{\perp} \leqslant s' < s \rightarrow [\![A]\!]\gamma s']$

and prove $[\![C]\!]\gamma s$ as follows.

First we show that (7), together with $chan(A) \cap jchan = \varnothing$ (see (2)), leads to

(8) $\forall s'[(\hat{s} \backslash jchan)_{\perp} \leqslant s' < (\hat{s} \backslash jchan)\bar{s} \rightarrow [\![A]\!]\gamma s']$.

proof:

If $(\hat{s} \backslash jchan)_{\perp} \leqslant s' < (\hat{s} \backslash jchan)\bar{s}$,

then there exists an $\dot{s}$ with $s' = (\hat{s} \backslash jchan)\dot{s}$ and $\dot{s} < \bar{s}$.

Now take $s'' = \hat{s}[\dot{s}]_{extchan}$, then $\hat{s}_{\perp} \leqslant s'' < \hat{s}[\bar{s}]_{extchan}$,

and thus (remember $s = \hat{s}[\bar{s}]_{extchan}$): $\hat{s}_{\perp} \leqslant s'' < s$.

So (7) leads to $[\![A]\!]\gamma s'' = [\![A]\!]\gamma \hat{s}[\dot{s}]_{extchan} =$

$[\![A]\!]\gamma(\hat{s} \backslash jchan)[\dot{s}]_{extchan}$, since A does refer to $jchan$.

Observe that $\dot{s} < \bar{s}$, so $\dot{s}$ contains channel names from S_1 and S_2 only, and since $extchan = chan(S_1, S_2) - jchan$, we infer

$[\![A]\!]\gamma(\hat{s} \backslash jchan)\dot{s} = [\![A]\!]\gamma s'$.

□

Now $(\hat{s} \backslash jchan)\bar{s} \in \overline{M(S_1 \| S_2)}(\hat{s} \backslash jchan)$

(by (4) and $MP(\hat{W} \backslash jchan, \bar{W}^e)$, $MP(\bar{W}, \hat{W}^e \backslash jchan)$ from (5)),

together with the validity of (6) and (8) this leads, by using (1), to

$[\![C]\!]\gamma(\hat{s} \backslash jchan)\bar{s}$.

Since $chan(C) \cap jchan = \varnothing$ (from (2)), we obtain

$[\![C]\!]\gamma \hat{s} \bar{s}$.

$\bar{s}$ only contains channels from S_1 and S_2, and C does not refer to the joint channels, so

$[\![C]\!]\gamma \hat{s}[\bar{s}]_{extchan} = [\![C]\!]\gamma s$.

ii) Assume $\sigma \neq \perp$ and

(7) $\forall s'[\hat{s}_{\perp} \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s']$.

Note that in case i) we already proved $[\![C]\!]\gamma s$.

Prove $[\![q]\!]\gamma s$ as follows.

Again (7) and $chan(A) \cap jchan = \varnothing$ lead to

$\forall s'[(\hat{s} \backslash jchan)_{\perp} \leqslant s' \leqslant (\hat{s} \backslash jchan)\bar{s} \rightarrow [\![A]\!]\gamma s']$.

Together with (6) and (1) this gives

$[\![q]\!]\gamma(\hat{s} \backslash jchan)\bar{s}$.

Similar as above we can use $chan(q) \cap jchan = \varnothing$ (from(2)) and infer

$[\![q]\!]\gamma \hat{s} \bar{s} = [\![q]\!]\gamma \hat{s}[\bar{s}]_{extchan} = [\![q]\!]\gamma s$.

□

Proving soundness of the alternative rule "alt1" is straightforward and omitted here. The second rule "alt2" requires a rather long and tedious proof based on the same techniques used for soundness of the delay axiom and the rules for i/o and sequential

composition. Also the soundness proof of the iteration rule is omitted, because it is very similar to the usual proof for such a rule.

parallel composition

According to the parallel composition rule for $S_1 \parallel S_2$, we assume the following,

let $jchan = chan(S_1) \cap chan(S_2)$,

(1) $\models (A_i, C_i) : \{p_i \wedge W_{jchan} = \varnothing\}\ S_i\ \{q_i\},\ i = 1,2$

(2) $\models q_1[{}^{v_1}/_{time}, {}^{w_1}/_{W}, {}^{w_3}/_{W^e}] \wedge q_2[{}^{v_2}/_{time}, {}^{w_2}/_{W}, {}^{w_4}/_{W^e}] \wedge time = max(v_1, v_2) \wedge$
$W = w_1 \cup w_2 \wedge W^e = w_3 \cup w_4 - [w_3 \cup w_4]_{jchan} \rightarrow q$

(3) $\models C_1[{}^{w_1}/_{W}] \wedge C_2[{}^{w_2}/_{W}] \wedge W = w_1 \cup w_2 \rightarrow C$

(4) $\models (C_1 |^{W} jchan)[{}^{W^e}/_{W}] \wedge A \rightarrow A_2$

(5) $\models (C_2 |^{W} jchan)[{}^{W^e}/_{W}] \wedge A \rightarrow A_1$

(6) v_1, v_2, w_1, w_2, w_3 and w_4 logical VAL variables not occurring free in C or q,

(7) $wchan(A) \cap jchan = \varnothing$,

(8) $chan(p_i, q_i, A_i, C_i) \cap chan(S_j) \subseteq chan(S_i)$, and

(9) $var(p_i, q_i) \cap var(S_j) = \varnothing$, for $(i, j) \in \{(1,2),(2,1)\}$.

We have to prove:

$\models (A, C) : \{p_1 \wedge p_2 \wedge W_{jchan} = \varnothing\}\ S_1 \parallel S_2\ \{q\}$.

proof: Choose γ, $\hat{s} \in \mathbb{B}$, $\hat{o} \neq \perp$, and $\hat{\alpha} \neq \perp$ arbitrary.

Assume $[\![p_1 \wedge p_2 \wedge W_{jchan} = \varnothing]\!] \gamma \hat{s}$. Thus

(10) $[\![p_i]\!] \gamma \hat{s}$ $,i = 1,2$ and

(11) $\hat{W}_{jchan} = \varnothing$.

Let $s \in \overline{M(S_1 \parallel S_2)}\,\hat{s}$, then $s = \hat{s}\bar{s}$, where

$\bar{s} = (\bar{\tau}, W_1 \cup W_2, W_1^e \cup W_2^e - [W_1^e \cup W_2^e]_{jchan}, \bar{\sigma}, max(\alpha_1, \alpha_2))$ with, for $i = 1,2$:

$s_i = (\tau_i, W_i, W_i^e, \sigma_i, \alpha_i) \in M(S_i)(\hat{o}, \hat{\alpha})$, and

(12) $[\bar{\tau}]_{chan(S_i)} = \tau_i \wedge (D \notin chan(S_1 \parallel S_2) \rightarrow [\bar{\tau}]_D = <>)$,

(13) $[W_1]_{jchan} = [W_2^e]_{jchan} \wedge [W_2]_{jchan} = [W_1^e]_{jchan}$,

(14) If $\sigma_1 \neq \perp \wedge \sigma_2 \neq \perp$ then $\bar{\sigma}(x) = \begin{cases} \sigma_i(x) & , x \in var(S_i) \\ \hat{\sigma}(x) & , x \notin var(S_1, S_2), \end{cases}$

if $\sigma_1 = \perp \vee \sigma_2 = \perp$ then $\bar{\sigma} = \perp$.

i) Assume $\forall s' [\hat{s}_{\perp} \leqslant s' < s \rightarrow [\![A]\!] \gamma s']$, and prove

(15) $[\![C]\!] \gamma s$.

proof: First prove, with induction on $|\tau_1| + |\tau_2| + |W_1 \cup W_2|$:

(16) $\bigwedge_{i=1}^{2} [\![C_i]\!] \gamma \hat{s} s_i$.

- Basic step: $|\tau_1| + |\tau_2| + |W'_1 \cup W'_2| = 0$, thus $\tau_1 = \tau_2 = <>$, $W'_1 = W'_2 = \varnothing$.
  Then there is no s_i' with $\hat{s}_\perp \leqslant s_i' < \hat{s}\, s_i$ (see definition $<$), so
  $\forall s_i' [\, \hat{s}_\perp \leqslant s_i' < \hat{s}\, s_i \rightarrow [\![A_i]\!]\gamma s_i' \,]$. From (1) we obtain $[\![C_i]\!]\gamma \hat{s}\, s_i$, for $i = 1,2$.

- Induction step: let $|\tau_1| + |\tau_2| + |W'_1 \cup W'_2| > 0$.
  Assume, by induction, that (16) holds for smaller values.
  Let $(i,j) \in \{(1,2),(2,1)\}$.
  From (1) we could prove $[\![C_i]\!]\gamma \hat{s}\, s_i$ if $\forall s_i' [\, \hat{s}_\perp \leqslant s_i' < \hat{s}\, s_i \rightarrow [\![A_i]\!]\gamma s_i' \,]$ holds.
  So first the following lemma is proven.
  Lemma: $\forall s_i' [\, \hat{s}_\perp \leqslant s_i' < \hat{s}\, s_i \rightarrow [\![A_i]\!]\gamma s_i' \,]$.
  proof: Let $\hat{s}_\perp \leqslant s_i'' < \hat{s}\, s_i$, then $s_i'' = \hat{s}\, s_i'$ with $s_i' < s_i$.
  Thus $s_i' \in M(S_i)(\hat{o}, \hat{\alpha})$, because $M(S_i)$ is prefix closed.
  Now choose $s_j' \in M(S_j)(\hat{o}, \hat{\alpha})$, such that, if we define
  $s' = (\tau', W'_1 \cup W'_2, W'^e_1 \cup W'^e_2 - [W'^e_1 \cup W'^e_2]_{jchan}, \perp, \perp)$, where
  (17) $[\tau']_{chan(S_i)} = \tau_i' \;\wedge\; [\tau']_{chan(S_j)} = \tau_j' \;\wedge$
  $(D \notin chan(S_1 \parallel S_2) \rightarrow [\tau']_D = <>)$, and
  (18) $[W_i']_{jchan} = [W_j^{e\prime}]_{jchan}$.
  then $s' < \bar{s}$ and $s' \in M(S_1 \parallel S_2)(\hat{o}, \hat{\alpha})$.
  (Observe that for every $s_i' < s_i$ such an s_j' can be found, since s_i and s_j satisfy (12) and (13).)

  In order to prove A_i for tuple s_i'' we want to apply (4) and (5), so we need $C_j|^W_{jchan}$ and A, interpreted in a related tuple.

  Since $|\tau_1'| + |\tau_2'| + |W'_1 \cup W'_2| < |\tau_1| + |\tau_2| + |W'_1 \cup W'_2|$, we can use the induction hypothesis and derive $[\![C_j]\!]\gamma \hat{s}\, s_j' = [\![C_j]\!]\gamma(\hat{\tau}\tau_j', \hat{W} \cup W_j', ..., ..., ...)$.
  Now (8) and (17) lead to $[\![C_j]\!]\gamma(\hat{\tau}\tau', \hat{W} \cup W_j', ..., ..., ...)$.
  Applying projection to this assertion, we obtain
  $[\![\, C_j|^W_{jchan} \,]\!]\gamma(\hat{\tau}\tau', [\hat{W}]_{jchan} \cup [W_j']_{jchan}, ..., ..., ...)$. Thus
  $[\![\, (C_j|^W_{jchan})[^{W^e}/_W] \,]\!]\gamma(\hat{\tau}\tau', ..., [\hat{W}]_{jchan} \cup [W_j']_{jchan}, ..., ...)$.
  So (11), i.e. $[\hat{W}_j]_{jchan} = \varnothing$, leads to
  (19) $[\![\, (C_j|^W_{jchan})[^{W^e}/_W] \,]\!]\gamma(\hat{\tau}\tau', ..., [W_j']_{jchan}, ..., ...)$.
  We assumed $[\![A]\!]\gamma \dot{s}$ for all $\dot{s}$ with $\hat{s}_\perp \leqslant \dot{s} < s$, so
  $[\![A]\!]\gamma \dot{s}$ for all $\dot{s}$ with $\hat{s}_\perp \leqslant \dot{s} < \hat{s}\bar{s}$.
  Note that the s' defined above satisfies $s' < \bar{s}$, so $\hat{s}_\perp \leqslant \hat{s}\, s' < \hat{s}\bar{s}$, and thus
  $[\![A]\!]\gamma(\hat{\tau}\tau', ..., \hat{W}^e \cup W^{e\prime}, ..., ...) =$

$[\![A]\!]\gamma(\hat{\tau}\tau',..,\hat{W}^e \cup W_1^{e'} \cup W_2^{e'} - [W_1^{e'} \cup W_2^{e'}]_{jchan},...,...).$

From (7) we know that A does not refer to joint channels, so

(20) $[\![A]\!]\gamma(\hat{\tau}\tau',..,\hat{W}^e \cup W_1^{e'} \cup W_2^{e'} - [W_1^{e'} \cup W_2^{e'}]_{jchan} - [\hat{W}^e]_{jchan},...,...).$

In order to combine (19) and (20) by using (4) and (5) we define the following union of their environment wait records:

$\overline{W}_j^e = [W_j']_{jchan} \cup \hat{W}^e \cup W_1^{e'} \cup W_2^{e'} - [W_1^{e'} \cup W_2^{e'}]_{jchan} - [\hat{W}^e]_{jchan}$.

Because of the projection on *jchan* in the assertion of (19), we can add wait records with channels not in *jchan*, thus obtaining

$[\![(C_j \restriction^W jchan)[^{W^e}/_W]]\!]\gamma(\hat{\tau}\tau',..,\overline{W}_j^e,...,...).$

Furthermore, A does not refer to *jchan* in environment wait records (by (7)), thus validity of (20) leads to

$[\![A]\!]\gamma(\hat{\tau}\tau',..,\overline{W}_j^e,...,...).$

So by (4) and (5) we infer from the last two formulae:

$[\![A_i]\!]\gamma(\hat{\tau}\tau',..,\overline{W}_j^e,.....).$

Now A_i does not refer to external channels of S_j (see (8)),

so remove $W_j^{e'} - [W_j^{e'}]_{jchan}$ from the set of environment wait records:

$[\![A_i]\!]\gamma(\hat{\tau}\tau',..,[W_j']_{jchan} \cup \hat{W}^e \cup W_i^{e'} - [W_i^{e'}]_{jchan} - [\hat{W}^e]_{jchan},...,...).$

Remember (18): $[W_j']_{jchan} = [W_i^{e'}]_{jchan}$, so

$[\![A_i]\!]\gamma(\hat{\tau}\tau',..,\hat{W}^e \cup W_i^{e'} - [\hat{W}^e]_{jchan},...,...).$

Since A_i is *monotone* in W^e, we can add $[\hat{W}^e]_{jchan}$ to the set of environment wait records, and achieve

$[\![A_i]\!]\gamma(\hat{\tau}\tau',..,\hat{W}^e \cup W_i^{e'},...,...).$

By using (8) and (17):

$[\![A_i]\!]\gamma(\hat{\tau}\tau_i',..,\hat{W}^e \cup W_i^{e'},...,...) = [\![A_i]\!]\gamma\hat{s}s_i' = [\![A_i]\!]\gamma s_i''$,

which proves the lemma.

□

This lemma together with (10) and (11) leads, by using (1), to $[\![C_i]\!]\gamma\hat{s}s_i$. Thus (16).

Remains to prove (15): $[\![C]\!]\gamma s$.

From (16) we infer, by using (8) and (12): $[\![C_i]\!]\gamma(\hat{\tau}\overline{\tau},\hat{W} \cup W_i,...,...,...).$

Take, temporarily, a new environment $\gamma' = \gamma[^{\hat{W} \cup W_1}/_{W_1}, ^{\hat{W} \cup W_2}/_{W_2}]$, then

$[\![\bigwedge_{i=1}^{2} C_i[^{W_i}/_W] \wedge W = w_1 \cup w_2]\!]\gamma'(\hat{\tau}\overline{\tau},\hat{W} \cup W_1 \cup W_2,...,...,...).$

Hence (3) leads to $[\![C]\!]\gamma'(\hat{\tau}\overline{\tau},\hat{W} \cup W_1 \cup W_2,...,...,...).$

Thus from (6): $[\![C]\!]\gamma(\hat{\tau}\overline{\tau},\hat{W} \cup W_1 \cup W_2,...,...,...) = [\![C]\!]\gamma\hat{s}\overline{s} = [\![C]\!]\gamma s$.

□

ii) Assume $\sigma \neq \bot$ and $\forall s'[\hat{s}_\bot \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s']$.
We have to prove $[\![q]\!]\gamma s$.
proof: Let $i = 1,2$.
Similar to part i) we can prove: $\forall s_i'[\hat{s}_\bot \leqslant s_i' \leqslant \hat{s}\, s_i \rightarrow [\![A_i]\!]\gamma s_i']$.
Then (1) and (10) lead to $[\![q_i]\!]\gamma \hat{s}\, s_i = [\![q_i]\!]\gamma(\hat{\tau}\tau_i, \hat{W} \cup W_i, \hat{W}^e \cup W_i^e, \sigma_i, \alpha_i)$.
From (8), (9), (12) and (14) we obtain $[\![q_i]\!]\gamma(\hat{\tau}\bar{\tau}, \hat{W} \cup W_i, \hat{W}^e \cup W_i^e, \bar{\sigma}, \alpha_i)$.
Take $\gamma' = \gamma[{}^{\sigma_1}/_{v_1}, {}^{\sigma_2}/_{v_2}, {}^{\hat{W} \cup W_1}/_{w_1}, {}^{\hat{W} \cup W_2}/_{w_2}, {}^{\hat{W}^e \cup W_1^e}/_{w_3}, {}^{\hat{W}^e \cup W_2^e}/_{w_4}]$, then
$[\![q_i[{}^{v_i}/_{time}, {}^{w_i}/_{W}, {}^{w_{i+2}}/_{W^e}]]\!]\gamma'(\hat{\tau}\bar{\tau}, \hat{W} \cup W_1 \cup W_2,$
$\hat{W}^e \cup W_1^e \cup W_2^e - [W_1^e \cup W_2^e]_{jchan}, \bar{\sigma}, max(\alpha_1, \alpha_2))$.
Thus
$[\![\bigwedge_{i=1}^{2} q_i[{}^{v_i}/_{time}, {}^{w_i}/_{W}, {}^{w_{i+2}}/_{W^e}] \wedge time = max(v_1, v_2) \wedge W = w_1 \cup w_2 \wedge$
$W^e = w_3 \cup w_4 - [w_3 \cup w_4]_{jchan}]\!]\gamma' \hat{s}\, \bar{s}$.
Then (2) leads to $[\![q]\!]\gamma' \hat{s}\, \bar{s}$, so by using (6) $[\![q]\!]\gamma \hat{s}\, \bar{s} = [\![q]\!]\gamma s$.
□

□

Soundness of the **consequence**, the **substitution**, and the **conjunction** rule, and the **invariance** axiom is straightforward.

prefix invariance
Let $cset = chan(L)$, and t some trace variable. We have to prove:
$\models (TRUE, \pi_{cset} \geqslant t) : \{\pi_{cset} = t\}\, S_1 \| S_2\, \{\pi_{cset} \geqslant t\}$.
proof:
For traces τ_1 and τ_2, $\tau_1 \leqslant \tau_2$ denotes that τ_1 is an initial prefix of τ_2.
Observe that $[\![\pi_{cset} \geqslant t]\!]\gamma s \;\rightleftarrows\; [\tau]_{cset} \geqslant \gamma(t)$.
Choose γ, $\hat{s} \in \mathbb{B}$, $\hat{\sigma} \neq \bot$, and $\hat{\alpha} \neq \bot$ arbitrary. Assume $[\![\pi_{cset} = t]\!]\gamma \hat{s}$, then
(1) $\gamma(t) = [\hat{\tau}]_{cset}$.
Let $s \in \overline{M(L)}\hat{s}$, then $s = \hat{s}\,\bar{s}$ with $\bar{s} \in M(L)$. Thus
(2) $\tau \geqslant \hat{\tau}$.

i) By (2) and (1) we see: $[\tau]_{cset} \geqslant [\hat{\tau}]_{cset} = \gamma(t)$,
so $[\![\pi_{cset} \geqslant t]\!]\gamma s$.

ii) Similar for the postcondition $\pi_{cset} \geqslant t$.

□

strengthen

Let $cset = chan(L)$, and t a trace variable not occurring free in A. Assume

(1) $chan(A) \subseteq cset$, and

(2) $wchan(A) = \varnothing$.

We have to prove:

$\models (A, \forall t\ [t_0 \leqslant t < \pi_{cset} \rightarrow A[t/\pi]]) : \{\pi_{cset} = t_0\}\ L\ \{A\}$.

proof:

Choose γ, $\hat{s} \in \mathbb{B}$, $\hat{\sigma} \neq \perp$, and $\hat{\alpha} \neq \perp$ arbitrary. Assume $[\![\pi_{cset} = t_0]\!]\gamma\hat{s}$, then

(3) $\gamma(t_0) = [\hat{\tau}]_{cset}$.

Let $s \in \overline{M(L)}\hat{s}$.

i) Assume (4) $\forall s'[\hat{s}_{\perp} \leqslant s' < s \rightarrow [\![A]\!]\gamma s']$.
We have to prove $[\![\forall t\ [t_0 \leqslant t < \pi_{cset} \rightarrow A[t/\pi]]]\!]\gamma s$.
Choose t arbitrary, and assume $[\![t_0 \leqslant t < \pi_{cset}]\!]\gamma s$, thus
$\gamma(t_0) \leqslant \gamma(t) < [\tau]_{cset}$. By (3) we obtain:
$[\hat{\tau}]_{cset} \leqslant \gamma(t) < [\tau]_{cset}$.
Then there exists an τ' with $\hat{\tau} \leqslant \tau' < \tau$, such that $\gamma(t) = [\tau']_{cset}$.
Thus we can find an s' with $\hat{s}_{\perp} \leqslant s' < s$, hence
$[\![A]\!]\gamma s'$, by (4).
Now (1) leads to $[\![A]\!]\gamma([\tau']_{cset}, W', W^e', \sigma', \alpha') =$
$[\![A[t/\pi]]\!]\gamma([\tau']_{cset}, W', W^e', \sigma', \alpha') =$
$[\![A[t/\pi]]\!]\gamma s$, since π and W^e (see (2)) do not occur in $A[t/\pi]$.

ii) Assume $\sigma \neq \perp$ and $\forall s'[\hat{s}_{\_} \leqslant s' \leqslant s \rightarrow [\![A]\!]\gamma s']$,
then (take $s' = s$) $[\![A]\!]\gamma s$.

□

So all rules and axioms of the proof system as given in chapter 5 are sound w.r.t. the semantics defined in chapter 3, and using the interpretation of correctness formulae from chapter 4.

□

B. REFERENCES

[deB] de Bakker, J.W., Mathematical Theory of Program Correctness, Prentice Hall, London, (1980).

[FLP] Francez, N., Lehman, D., Pnueli, A., A Linear History Semantics for Distributed Programming, TCS 32, (1984), 25-46.

[Glass] Glass, R.L., The "Lost world" of Software Debugging and Testing, CACM 23, (1980), 264-271.

[Hoare] Hoare, C.A.R., Communicating Sequential Processes, CACM 21, (1978), 666-677.

[HdeR] Hooman, J., de Roever, W.P., The quest goes on: a survey of proof systems for partial correctness of CSP, Current Trends in Concurrency, LNCS 224, (1986), 343-395.

[HGR] Huizing, C., Gerth, R., de Roever, W.P., Full Abstraction of a Real-Time Denotational Semantics for an OCCAM-like Language, to appear in POPL 87, (1987).

[KR] Koymans, R., de Roever, W.P., Examples of a Real-Time Temporal Logic Specification, The Analysis of Concurrent Systems, LNCS 207, (1983), 231-252.

[KSRGA] Koymans, R., Shyamasundar, R.K., de Roever, W.P., Gerth, R., Arun-Kumar, S., Compositional Semantics for Real-Time Distributed Computing, Report no. 68, University of Nijmegen, to appear in Information and Control, (1986).

[KVR] Koymans, R., Vytopil, J., de Roever, W.P., Real-Time Programming and Asynchronous Message Passing, Proc. 2nd ACM Symposium on Principles of Distributed Computing, (1983).

[La] Lamport, L., What Good Is Temporal Logic?, Information Processing 83, R.E. Manson (ed.), North Holland, (1983), 190-222.

[MC] Misra, J., Chandy, K.M., Proofs of Networks of Processes, IEEE Transactions on Software Engineering, SE-7, (1981), 417-426.

[OCC] The OCCAM Language Reference Manual, Prentice Hall, (1984).

[PJ] Paritosh Pandya, Mathai Joseph, A-C Logic: A proof system for total correctness of OCCAM-S, Draft Tech. Rep., TIFR, Bombay, India, (1986).

[Z] Zwiers, J., Ph.D. Thesis, to appear, Eindhoven University of Technology, (June 1987).

[ZBR] Zwiers, J., de Bruin, A., de Roever, W.P., A proof system for partial correctness of dynamic networks, Logics of Programs 83, LNCS 164, (1983).

[ZRE84] Zwiers, J., de Roever, W.P., van Emde Boas, P., Compositionality and concurrent networks: soundness and completeness of a proofsystem, Report no. 57, University of Nijmegen, (1984).

[ZRE] Zwiers, J., de Roever, W.P., van Emde Boas, P., Compositionality and concurrent networks: soundness and completeness of a proofsystem, ICALP 85, LNCS 194, (1985).

COMPUTING SCIENCE NOTES

In this series appeared :

No.	Author(s)	Title
85/01	R.H. Mak	The formal specification and derivation of CMOS-circuits
85/02	W.M.C.J. van Overveld	On arithmetic operations with M-out-of-N-codes
85/03	W.J.M. Lemmens	Use of a computer for evaluation of flow films
85/04	T. Verhoeff H.M.J.L. Schols	Delay insensitive directed trace structures satisfy the foam rubber wrapper postulate
86/01	R. Koymans	Specifying message passing and real-time systems
86/02	G.A. Bussing K.M. van Hee M. Voorhoeve	ELISA, A language for formal specifications of information systems
86/03	Rob Hoogerwoord	Some reflections on the implementation of trace structures
86/04	G.J. Houben J. Paredaens K.M. van Hee	The partition of an information system in several parallel systems
86/05	Jan L.G. Dietz Kees M. van Hee	A framework for the conceptual modeling of discrete dynamic systems
86/06	Tom Verhoeff	Nondeterminism and divergence created by concealment in CSP
86/07	R. Gerth L. Shira	On proving communication closedness of distributed layers

86/08	R. Koymans R.K. Shyamasundar W.P. de Roever R. Gerth S. Arun Kumar	Compositional semantics for real-time distributed computing (Inf.&Control 1987)
86/09	C. Huizing R. Gerth W.P. de Roever	Full abstraction of a real-time denotational semantics for an OCCAM-like language
86/10	J. Hooman	A compositional proof theory for real-time distributed message passing
86/11	W.P. de Roever	Questions to Robin Milner - A responder's commentary (IFIP86)
86/12	A. Boucher R. Gerth	A timed failure semantics for communicating processes
86/13	R. Gerth W.P. de Roever	Proving monitors revisited: a first step towards verifying object oriented systems (Fund. Informatica IX-4)
86/14	R. Koymans	Specifying passing systems requires extending temporal logic
87/01	R. Gerth	On the existence of sound and complete axiomatizations of the monitor concept
87/02	Simon J. Klaver Chris F.M. Verberne	Federatieve Databases
87/03	G.J. Houben J.Paredaens	A formal approach distri-buted information systems
87/04	T.Verhoeff	Delay-insensitive codes - An overview

Available Reports from the Theoretical Computing Science Group

	Author(s)	Title	Classification EUT	Classification DESCARTES
TIR83.1	R. Koymans, J. Vytopil, W.P. de Roever	Real-Time Programming and Synchronous Message passing (2nd ACM PODC)		
TIR84.1	R. Gerth, W.P. de Roever	A Proof System for Concurrent Ada Programs (SCP4)		
TIR84.2	R. Gerth	Transition Logic - how to reason about temporal properties in a compositional way (16th ACM FOCS)		
TIR85.1	W.P. de Roever	The Quest for Compositionality - a survey of assertion-based proof systems for concurrent progams, Part I: Concurrency based on shared variables (IFIP85)		
TIR85.2	O. Grünberg, N. Francez, J. Makowsky, W.P. de Roever	A proof-rule for fair termination of guarded commands (Inf.& Control 1986)		
TIR85.3	F.A. Stomp, W.P. de Roever, R. Gerth	The μ-calculus as an assertion language for fairness arguments (Inf.& Control 1987)		
TIR85.4	R. Koymans, W.P. de Roever	Examples of a Real-Time Temporal Logic Specification (LNCS207)		
TIR86.1	R. Koymans	Specifying Message Passing and Real-Time Systems (extended abstract)	CSN86/01	
TIR86.2	J. Hooman, W.P. de Roever	The Quest goes on: A Survey of Proof Systems for Partial Correctness of CSP (LNCS227)	EUT-Report 86-WSK-01	

TIR86.3	R. Gerth, L. Shira	On Proving Communication Closedness of Distributed Layers (LNCS236)	CSN86/07	
TIR86.4	R. Koymans, R.K. Shyamasundar, W.P. de Roever, R. Gerth, S. Arun Kumar	Compositional Semantics for Real-Time Distributed Computing (Inf.&Control 1987)	CSN86/08	
TIR86.5	C. Huizing, R. Gerth, W.P. de Roever	Full Abstraction of a Real-Time Denotational Semantics for an OCCAM-like Language	CSN86/09	PE.01
TIR86.6	J. Hooman	A Compositional Proof Theory for Real-Time Distributed Message Passing	CSN86/10	TR.4-1-1(1)
TIR86.7	W.P. de Roever	Questions to Robin Milner - A Responder's Commentary (IFIP86)	CSN86/11	
TIR86.8	A. Boucher, R. Gerth	A Timed Failure Semantics for Communicating Processes	CSN86/12	TR.4-4(1)
TIR86.9	R. Gerth, W.P. de Roever	Proving Monitors Revisited: a first step towards verifying object oriented systems (Fund. Informatica IX-4)	CSN86/13	
TIR86.10	R.Koymans	Specifying Message Passing Systems Requires Extending Temporal Logic	CSN86/14	PE.02
TIR87.1	R. Gerth	On the existence of sound and complete axiomatizations of the monitor concept	CSN87/01	