T U M

I N S T I T U T F Ü R I N F O R M A T I K

A Denotational Model for Mobile Many-to-Many Data-flow Networks

Radu Grosu
Ketil Stølen

TUM-I9622
Mai 1996

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-05-1996-I9622-350/1.-FI
Alle Rechte vorbehalten
Nachdruck auch auszugsweise verboten

©1996 MATHEMATISCHES INSTITUT UND
INSTITUT FÜR INFORMATIK
TECHNISCHE UNIVERSITÄT MÜNCHEN

Typescript: ---

Druck: Mathematisches Institut und
Institut für Informatik der
Technischen Universität München

TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

A Denotational Model for Mobile Many-to-Many Data-flow Networks

Radu Grosu, Ketil Stølen

TUM-I9622
SFB-Bericht Nr.342/13/96 A
Mai 1996

TUM–INFO–05-96-I22-350/1.–FI

Alle Rechte vorbehalten

Nachdruck auch auszugsweise verboten

©1996 SFB 342 Methoden und Werkzeuge für
die Nutzung paralleler Architekturen

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

Druck: Fakultät für Informatik der
Technischen Universität München

A Denotational Model for Mobile Many-to-Many Data-flow Networks

Radu Grosu, Ketil Stølen

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

June 11, 1996

Abstract

We present a fully abstract, denotational model for mobile, timed, nondeterministic data-flow networks whose components communicate in a many-to-many fashion. In this model components and networks of components are represented by sets of stream processing functions. Each stream processing function is required to be strongly guarded and generic. A stream processing function is strongly guarded if it is contractive with respect to the standard metric on streams. This property guarantees the existence of unique fix-points. The genericity property can be thought of as an invariant, or alternatively, a privacy requirement, that is satisfied by any mobile system. It guarantees that a function never accesses, depends on or forwards a port whose name it does not already know. Our model allows the description of a wide variety of networks — in particular, the description of unbounded nondeterministic networks. We demonstrate some features of our model by specifying a mobile telephone network.

1 Introduction

One of the most prominent theories for interactive computation is the theory of data-flow networks. In this theory, an interactive system is represented by a network of autonomous components which communicate solely by asynchronous transmission of messages via directed channels.

A very elegant model for static, deterministic data-flow networks whose components communicate in a point-to-point fashion, was given by Kahn in [Kah74]. Despite of its elegant foundation, this class of networks is, however, too restrictive for many practical applications. In this paper we extend Kahn's model in a number of ways.

Firstly, contrary to [Kah74], we model nondeterministic behavior. Like Park [Par83], Broy [Bro87] and Russell [Rus90], we represent nondeterministic data-flow networks by sets of stream processing functions. However, in contrast with [Par83] and [Bro87], our model is fully abstract. This is achieved, by considering only sets of functions which are closed with respect to the external observations. The closure idea was used by [Rus90] for the same purpose. However, contrary to [Rus90], we use a timed model and a different notion of observation. This allows us to describe a considerably greater class of networks. In

particular, we can describe all fair merge components discussed in [PS92]. In fact, we can describe any liveness property that can be expressed in standard property oriented specification languages for distributed systems [CM88], [Lam91], [BDD$^+$93]. Moreover, since our model is fully abstract, we obviously avoid the expressiveness problem known as the Brock/Ackermann anomaly [BA81].

Secondly, contrary to [Kah74], and also contrary to [Par83], [Bro87], [Rus90], we handle many-to-many communication. This is achieved by building implicit fair merge components into the network operators — the operators used to build networks from basic components. Despite the fact that several components may have both receive and send access to the same channel, each component is described by a set of functions mapping input streams to output streams. The input streams contain the messages sent by the environment; the output streams contain the messages sent by the component. Thus, we model the shared-state interference caused by the many-to-many communication at a very abstract level — the interference is isolated and placed in the network operators.

Thirdly, contrary to [Kah74] and also contrary to [Par83], [Bro87], [Rus90], we describe dynamically reconfigurable or mobile networks — networks in which every component may change its communication partners on the basis of computation and interaction. The formal modeling of mobility has been a very popular research direction in recent years. However, most models published so far have been formalized mainly in operational terms. Examples of such models are the Actor Model [HBS73, AMST92], the π–Calculus [EN86, MPW92a, MPW92b], the Chemical Abstract Machine [BB90], the Rewriting Logic [Mes91] and the Higher Order CCS [Tho89]. On the contrary, our model gives a denotational formalization of mobility. As in the above models, this formalization is based on two assumptions. Firstly, ports are allowed to be passed between the network components. Secondly, the components preserve privacy: their behavior do not depend on ports they do not know. Although it is well understood how to express privacy operationally, there is less denotational understanding. Our solution is to require each stream processing function to be generic. This requirement can be thought of as an invariant satisfied by any mobile system. Informally speaking, the genericity property makes sure that a function accesses, depends on and forwards only ports it already knows. By "the ports it already knows" we basically mean any port which is in its initial interface, it has already received or it has already created itself. Any port created by the function itself is assigned a "new" name taken from a set that is "private" to the component in question.

Although we could have formulated our semantics in a cpo context, we decided to base it on the topological tradition of metric spaces [Niv82, dBZ82, AdBKR89]. Firstly, we wanted to understand the exact relationship between our approach and those based on metric spaces. Secondly, the use of metric spaces seems more natural since our approach is based on infinite streams, and since our strong guardedness constraint, guaranteeing the existence of a unique fix-point, corresponds straightforwardly to contractivity.

The rest of the paper is split into seven sections. Section 2 introduces basic notions like communication histories and stream processing functions. Section 3 formalizes the genericity constraint. Section 4 introduces mobile components. Section 5 describes the modeling of interference. Section 6 is devoted to the network operators. Section 7 gives an example. Section 8 relates our approach to other approaches known from the literature. Finally, there is an appendix reviewing some basic stuff on metric spaces and streams.

2 Basic Notions

We model an interactive system by a network of autonomous components communicating via directed channels in a time-synchronous and message-asynchronous way. Time-synchrony is achieved by using a global clock splitting the time axis into discrete, equidistant time units. Message-asynchrony is achieved by allowing arbitrary, but finitely many messages to be sent along a channel in each time unit.

2.1 Communication Histories

We model the communication histories of directed channels by infinite streams of finite streams of messages. Each finite stream represents the communication history within a time unit. The first finite stream contains the messages received within the first time unit, the second the messages received within the second time unit, and so on. Since time never halts, any complete communication history is infinite. Let M be the set of all messages. Then $[M^*]$ is the set of all complete communication histories, and $(M^*)^*$ is the set of all partial communication histories[1].

In the introduction we anticipated that components may transmit *ports.* A port is a *channel name* together with an *access right*, which is either a receive right, represented by ?, or a send right, represented by !. Hence, if N is the set of all channel names, then $?N = \{?n \mid n \in N\}$ is the corresponding set of receive ports, and $!N = \{!n \mid n \in N\}$ is the corresponding set of send ports. We also write $?!N$ for $?N \cup !N$. We assume that $?!N \subseteq M$. Let D be the set of all messages not contained in the set of ports, i.e., $D = M \setminus ?!N$.

Since components may exchange ports, each component can potentially access any channel in N. For that reason we model the complete input and output histories of a component by named stream tuples contained in $N \to [M^*]$. The partial ones are modeled by $N \to (M^*)^*$. In the sequel we refer to named stream tuples of these signatures as *named communication histories.* Thus, each named communication history assigns a communication history to each channel name in N.

2.2 Guarded Functions

A *deterministic component* is modeled by a stream processing function

$$f \in (N \to [M^*]) \to (N \to [M^*])$$

mapping complete named communication histories for its input channels to complete named communication histories for its output channels. Note that if no message is communicated along an input channel within a time unit then the empty stream occurs in the communication history for that channel. The lack of this information causes the fair merge anomaly [Kel78].

The functions process their inputs *incrementally* — at any point in time, their outputs do not depend on future inputs. Functions satisfying this constraint are called *weakly*

[1] For an arbitrary set S, by S^* we denote the set of finite streams over S and by $[S]$ the set of infinite streams over S. See also the appendix.

guarded. If the outputs they produce in time unit t are not only independent of future inputs, i.e., the inputs received during time unit $t+1$ or later, but also of the inputs received during time unit t, then they are called *strongly guarded.* Intuitively, the strongly guarded functions introduce a delay of at least one time unit between input and output. The weakly guarded functions allow in addition zero-delay behavior.

Let $\theta\downarrow_j$ represent the prefix of θ of length j, i.e., the result of cutting θ after the j'th time unit. Then weak and strong guardedness can be formalized as below:

Definition 1 (Guarded functions) *A function* $f \in (N \to [M^*]) \to (N \to [M^*])$ *is weakly guarded if*

$$\forall \theta, \varphi, j : \ \theta\downarrow_j = \varphi\downarrow_j \quad \Rightarrow \quad f(\theta)\downarrow_j = f(\varphi)\downarrow_j,$$

and strongly guarded if

$$\forall \theta, \varphi, j : \ \theta\downarrow_j = \varphi\downarrow_j \quad \Rightarrow \quad f(\theta)\downarrow_{j+1} = f(\varphi)\downarrow_{j+1}.$$ □

We use the arrow $\stackrel{w}{\to}$ to characterize sets of weakly guarded functions.

A weakly guarded function is *non-expansive* and a strongly guarded function is *contractive* with respect to the metric on streams (see appendix). As a consequence, by Banach's fix-point theorem, strong guardedness not only replaces the usual monotonicity and continuity constraints of domain theory but also guarantees unique fix-points of feedback loops.

3 Genericity

A stream processing function $f \in (N \to [M^*]) \to (N \to [M^*])$ used to model a component is not only required to be strongly guarded but also to be generic. That it is *generic* means that it *accesses*, *depends on* and *forwards* only ports it already knows. Thus, the genericity property basically characterizes the way the function gains access to ports. In this section we formalize this additional property.

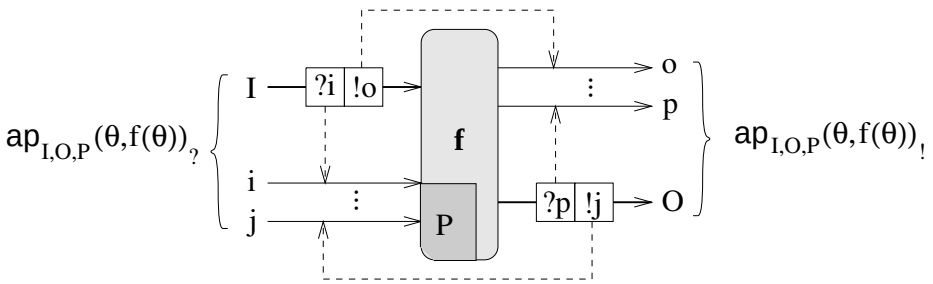

Figure 1: Generic Stream Processing Function

The behavior of a generic function can be described with respect to Figure 1, as follows. Initially, f receives from a designated set of input channels I and sends on a designated set of output channels O. These two sets name the *static* channels or the initial wiring. To make sure that channels *created* by the different components in a network have different

names, each mobile function is assigned a set of *private names* P. Obviously, this set should be disjoint from the static interface. Thus, we require that $(I \cup O) \cap P = \emptyset$.

During the computation, the sets of accessible channels gradually grow. For example, if the function receives a receive port $?i$ ($i \notin P$) then it may receive from the channel i, and if it receives a send port $!o$ ($o \notin P$) then it may send on the channel o. Similarly, whenever the function sends a send port $!j$, whose channel $j \in P$ it has created itself, it may later receive what is sent along j, or whenever it sends a receive port $?p$, whose channel $p \in P$ it has created itself, it may itself send messages along p which eventually are received by the components which received the receive port.

To formally characterize this behavior we introduce some additional notation. For a given point in time n and named input history θ, by $\mathsf{ap}(\theta, f(\theta))(n)$ we denote the set of *active* input and output ports, and by $\mathsf{pp}(\theta, f(\theta))(n)$ we denote the set of *passive* input and output ports. At any point in time n these two sets are disjoint. For any channel p, by $\widetilde{p}$ we denote its complement, i.e., $\widetilde{!p} = ?p$ and $\widetilde{?p} = !p$. A port p in $\mathsf{pp}(\theta, f(\theta))(n)$ remains passive as long as its complement port $\widetilde{p}$ is unknown to the environment. After all, if $\widetilde{p}$ is unknown to the environment then there is no way the environment can receive what f sends along p if p is a send port, and there is no way the environment can influence what f receives on p if p is a receive port. Formally:

Definition 2 (Active and passive ports) *For any n, I, O, P, θ and δ, let*

$$\mathsf{ap}_{I,O,P}(\theta,\delta)(n) \stackrel{\mathsf{def}}{=} \mathsf{ap}_n, \quad \mathsf{pp}_{I,O,P}(\theta,\delta)(n) \stackrel{\mathsf{def}}{=} \mathsf{pp}_n,$$

where ap_n and pp_n are defined recursively as follows:

$$\begin{array}{llll} \mathsf{ap}_1 & \stackrel{\mathsf{def}}{=} & ?I \cup !O, & \mathsf{pp}_1 \stackrel{\mathsf{def}}{=} ?!P, \\ \mathsf{ap}_{n+1} & \stackrel{\mathsf{def}}{=} & \mathsf{ap}_n \cup r_n \cup g_n, & \mathsf{pp}_{n+1} \stackrel{\mathsf{def}}{=} \mathsf{pp}_n \setminus g_n, \end{array}$$

where

$$r_n = \bigcup_{?i \in \mathsf{ap}_n} \{p \mid p \in \overline{\mathsf{pp}_n} \wedge p \in \theta(i)(n)\},$$
$$g_n = \bigcup_{!i \in \mathsf{ap}_n} \{p \mid p \in \mathsf{pp}_n \wedge \widetilde{p} \in \delta(i)(n)\}.$$

□

The sets r_n and g_n are the sets of received and generated ports, respectively.

Theorem 1 *For any n, I, O, P, θ and δ*

$$\mathsf{pp}_{I,O,P}(\theta,\delta)(1) \subseteq \mathsf{ap}_{I,O,P}(\theta,\delta)(n) \cup \mathsf{pp}_{I,O,P}(\theta,\delta)(n)$$
$$\mathsf{pp}_{I,O,P}(\theta,\delta)(n) \subseteq \mathsf{pp}_{I,O,P}(\theta,\delta)(1)$$

Proof: Follows trivially since anything that is removed from the set of passive ports is added to the set of active ports, and since nothing is ever added to the set of passive ports. □

With $\mathsf{dom}_{I,O,P}$ and $\mathsf{rng}_{I,O,P}$ we describe how f dynamically gain access to ports. The expression $\mathsf{dom}_{I,O,P}(\theta, f(\theta))(n)$ characterizes the input history that is actually considered by f in time unit n. Accordingly, $\mathsf{rng}_{I,O,P}(\theta, f(\theta))(n)$ is the output history that is actually produced by f in time unit n. In the definition of genericity below, $\mathsf{dom}_{I,O,P}$ and $\mathsf{rng}_{I,O,P}$ constrain f to maintain the privacy invariant described above. Since the function f runs in an open environment this constraint is of course not sufficient unless also the environment

sticks to the rules of the game. There are basically two ways in which the environment of f can break the rules of the game. Firstly, the environment can send f a port $p \in !?P$ which the environment has not yet received from f, i.e., a port "it does not yet know" because it has not yet been output by f. Secondly, the environment can send along a channel $c \in P$ without first having received $!c$.

There are two ways to deal with this problem. One alternative is to impose an environment assumption in all definitions characterizing exactly those input histories in which the environment sticks to the rules of the game. The other alternative, which is used in this paper, is to constrain the functions to ignore the input messages which do not respect the privacy restrictions. This is okay, because we are only interested in environments that can be understood as mobile components in accordance with the definition below. Such components will never break the rules of the game. For that reason, $\mathsf{dom}_{I,O,P}$ and $\mathsf{rng}_{I,O,P}$ have been defined in such a way that, in addition to their main task of characterizing the actual domain and range of a function, they also correct environment mistakes. Ports sent by the environment that should be unknown to the environment are filtered away. Moreover, messages sent by the environment along channels it does know are removed. Formally:[2]

Definition 3 (Domain and range) *For any n, I, O, P, θ and δ, we define*

$$\mathsf{dom}_{I,O,P}(\theta,\delta)(i)(n) \stackrel{\mathsf{def}}{=} \begin{cases} (\widetilde{\overline{\mathsf{pp}_n}} \cup D) \copyright \theta(i)(n) & \textit{if} \quad ?i \in \mathsf{ap}_n \\ \epsilon & \textit{otherwise} \end{cases}$$

$$\mathsf{rng}_{I,O,P}(\theta,\delta)(i)(n) \stackrel{\mathsf{def}}{=} \begin{cases} (\mathsf{pp}_n \cup \mathsf{ap}_n \cup D) \copyright \delta(i)(n) & \textit{if} \quad !i \in \mathsf{ap}_n \\ \epsilon & \textit{otherwise} \end{cases}$$

where $\mathsf{ap}_n = \mathsf{ap}_{I,O,P}(\theta,\delta)(n)$ *and* $\mathsf{pp}_n = \mathsf{ap}_{I,O,P}(\theta,\delta)(n)$. □

Theorem 2 *The functions* pp *and* ap *are strongly guarded, and the functions* dom *and* rng *are weakly guarded.*

Proof: $\mathsf{pp}_{I,O,P}(\theta,\delta)(n)$ and $\mathsf{ap}_{I,O,P}(\theta,\delta)(n)$ depend only on $\theta{\downarrow_{n-1}}$ and $\delta{\downarrow_{n-1}}$. $\mathsf{dom}_{I,O,P}(\theta,\delta)(n)$ and $\mathsf{rng}_{I,O,P}(\theta,\delta)(n)$ depend only on $\theta{\downarrow_n}$ and $\delta{\downarrow_n}$. □

Theorem 3 *The functions* dom *and* rng *have the following properties:*

$$\begin{aligned} \mathsf{dom}_{I,O,P}(\theta,\delta) &= \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta) = \mathsf{dom}_{I,O,P}(\theta,\mathsf{rng}_{I,O,P}(\theta,\delta)), \\ \mathsf{rng}_{I,O,P}(\theta,\delta) &= \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta) = \mathsf{rng}_{I,O,P}(\theta,\mathsf{rng}_{I,O,P}(\theta,\delta)). \end{aligned}$$

Proof: The proof is based on the inductive definitions of ap and pp.

Induction hypothesis:

$$\begin{aligned} \mathsf{ap}_{I,O,P}(\theta,\delta)(n) &= \mathsf{ap}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta)(n) = \mathsf{ap}_{I,O,P}(\theta,\mathsf{rng}_{I,O,P}(\theta,\delta))(n), \\ \mathsf{pp}_{I,O,P}(\theta,\delta)(n) &= \mathsf{pp}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta)(n) = \mathsf{pp}_{I,O,P}(\theta,\mathsf{rng}_{I,O,P}(\theta,\delta))(n). \end{aligned}$$

To simplify the notation we write:

$$\begin{aligned} \mathsf{ap}_n &= \mathsf{ap}_{I,O,P}(\theta,\delta)(n), \\ \mathsf{ap}'_n &= \mathsf{ap}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta)(n), \quad \mathsf{ap}''_n = \mathsf{ap}_{I,O,P}(\theta,\mathsf{rng}_{I,O,P}(\theta,\delta))(n), \\ \mathsf{pp}_n &= \mathsf{pp}_{I,O,P}(\theta,\delta)(n), \\ \mathsf{pp}'_n &= \mathsf{pp}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta)(n), \quad \mathsf{pp}''_n = \mathsf{pp}_{I,O,P}(\theta,\mathsf{rng}_{I,O,P}(\theta,\delta))(n). \end{aligned}$$

[2] The operator $\in$ is overloaded to test for containment in a list. Moreover, for any set of ports $S \subseteq ?!N$, $\overline{S} \stackrel{\mathsf{def}}{=} ?!N \setminus S$ and $\widetilde{S} \stackrel{\mathsf{def}}{=} \{\widetilde{p} \mid p \in S\}$. When convenient, we also view $\theta(i)(n)$ as a set.

Base case: $\mathsf{ap}_1 = \mathsf{ap}'_1 = \mathsf{ap}''_1 = ?I \cup !O$ and $\mathsf{pp}_1 = \mathsf{pp}'_1 = \mathsf{pp}''_1 = ?!P$.

Induction Step: By induction hypothesis $\mathsf{ap}_n = \mathsf{ap}'_n = \mathsf{ap}''_n$ and $\mathsf{pp}_n = \mathsf{pp}'_n = \mathsf{pp}''_n$. By definition of ap and pp:

$$\begin{aligned}
\mathsf{ap}_{n+1} = \; & (\mathsf{ap}_n \cup \textstyle\bigcup_{?i \in \mathsf{ap}_n} \{c \mid c \in \overline{\mathsf{pp}_n} \wedge c \in \theta(i)(n)\} \cup \\
& \textstyle\bigcup_{!i \in \mathsf{ap}_n} \{c \mid c \in \mathsf{pp}_n \wedge \widetilde{c} \in \delta(i)(n)\}), \\
\mathsf{ap}'_{n+1} = \; & (\mathsf{ap}'_n \cup \textstyle\bigcup_{?i \in \mathsf{ap}'_n} \{c \mid c \in \overline{\mathsf{pp}'_n} \wedge c \in \mathsf{dom}_{I,O,P}(\theta,\delta)(i)(n)\} \cup \\
& \textstyle\bigcup_{!i \in \mathsf{ap}'_n} \{c \mid c \in \mathsf{pp}'_n \wedge \widetilde{c} \in \delta(i)(n)\}), \\
\mathsf{ap}''_{n+1} = \; & (\mathsf{ap}''_n \cup \textstyle\bigcup_{?i \in \mathsf{ap}''_n} \{c \mid c \in \overline{\mathsf{pp}''_n} \wedge c \in \theta(i)(n)\}) \cup \\
& \textstyle\bigcup_{!i \in \mathsf{ap}''_n} \{c \mid c \in \mathsf{pp}''_n \wedge \widetilde{c} \in \mathsf{rng}_{I,O,P}(\theta,\delta)(i)(n)\}), \\
\mathsf{pp}_{n+1} = \; & \mathsf{pp}_n \setminus \textstyle\bigcup_{!i \in \mathsf{ap}_n} \{c \mid c \in \mathsf{pp}_n \wedge \widetilde{c} \in \delta(i)(n)\}, \\
\mathsf{pp}'_{n+1} = \; & \mathsf{pp}'_n \setminus \textstyle\bigcup_{!i \in \mathsf{ap}'_n} \{c \mid c \in \mathsf{pp}'_n \wedge \widetilde{c} \in \delta(i)(n)\}, \\
\mathsf{pp}''_{n+1} = \; & \mathsf{pp}''_n \setminus \textstyle\bigcup_{!i \in \mathsf{ap}''_n} \{c \mid c \in \mathsf{pp}''_n \wedge \widetilde{c} \in \mathsf{rng}_{I,O,P}(\theta,\delta)(i)(n)\}.
\end{aligned}$$

By definition of dom and rng:

$$\begin{aligned}
\mathsf{dom}_{I,O,P}(\theta,\delta)(i)(n) &= (\widetilde{\overline{\mathsf{pp}_n}} \cup D) \mathbin{©} \theta(i)(n) && \text{if} \quad ?i \in \mathsf{ap}_n = \mathsf{ap}'_n = \mathsf{ap}''_n, \\
\mathsf{rng}_{I,O,P}(\theta,\delta)(i)(n) &= (\mathsf{pp}_n \cup \mathsf{ap}_n \cup D) \mathbin{©} \delta(i)(n) && \text{if} \quad !i \in \mathsf{ap}_n = \mathsf{ap}'_n = \mathsf{ap}''_n.
\end{aligned}$$

The first union in the definition of ap_{n+1}, ap'_{n+1} and ap''_{n+1} is taken over $?i \in \mathsf{ap}_n = \mathsf{ap}'_n = \mathsf{ap}''_n$. As a consequence

$$\mathsf{dom}_{I,O,P}(\theta,\delta)(i)(n) = (\widetilde{\overline{\mathsf{pp}_n}} \cup D) \mathbin{©} \theta(i)(n)$$

inside this union. It is enough to show that

$$c \in \theta(i)(n) \quad \Leftrightarrow \quad c \in (\widetilde{\overline{\mathsf{pp}_n}} \cup D) © \theta(i)(n)$$

under the assumption that $c \notin \mathsf{ap}_n$ and $c \in \overline{\mathsf{pp}_n}$. This follows trivially since $\widetilde{c} \in \mathsf{pp}_1 \Leftrightarrow c \in \mathsf{pp}_1$, Theorem 1 and the two assumptions imply that $c \notin \widetilde{\overline{\mathsf{pp}_n}}$.

The second union in the definition of ap_{n+1}, ap'_{n+1} and ap''_{n+1} is taken over $!i \in \mathsf{ap}_n = \mathsf{ap}'_n = \mathsf{ap}''_n$. As a consequence

$$\mathsf{rng}_{I,O,P}(\theta,\delta)(i)(n) = (\mathsf{pp}_n \cup \mathsf{ap}_n \cup D) \mathbin{©} \delta(i)(n)$$

inside this union. It is enough to show that

$$\widetilde{c} \in \delta(i)(n) \quad \Leftrightarrow \quad \widetilde{c} \in (\mathsf{pp}_n \cup \mathsf{ap}_n \cup D) © \delta(i)(n)$$

under the assumption that $c \in \mathsf{pp}_n$. This follows trivially since $\widetilde{c} \in \mathsf{pp}_1 \Leftrightarrow c \in \mathsf{pp}_1$, Theorem 1 and the assumption imply that $\widetilde{c} \in \mathsf{pp}_n \cup \mathsf{ap}_n$. This proves that $\mathsf{ap}_{n+1} = \mathsf{ap}'_{n+1} = \mathsf{ap}''_{n+1}$. That $\mathsf{pp}_{n+1} = \mathsf{pp}'_{n+1} = \mathsf{pp}''_{n+1}$ follows accordingly.

Finally, because of these equalities, $\mathsf{dom}_{I,O,P}(\theta,\delta)(i)(n)$ simplifies to $\theta(i)(n)$ inside the definition of $\mathsf{dom}_{I,O,P}$ and $\mathsf{rng}_{I,O,P}(\theta,\delta)(i)(n)$ simplifies to $\delta(i)(n)$ inside the definition of $\mathsf{rng}_{I,O,P}$. This immediately proves the theorem. □

We can now characterize what it means for a function to be generic.

Definition 4 (Generic functions) *A function $f \in (N \to [M^*]) \to (N \to [M^*])$ is called* generic *with respect to the initial wiring (I, O) and the private names P iff:*

$$\forall \theta : f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta))) = \mathsf{rng}_{I,O,P}(\theta, f(\theta)).$$

□

We use the $Mob(I, O, P)$ to characterize the set of all strongly guarded functions that are generic with respect to (I, O, P). In the sequel we refer to these functions as mobile.

4 Mobile Components

We model a nondeterministic component by a set of mobile functions F. Any pair $(\theta, f(\theta))$, where $f \in F$, is a possible *behavior* of the component. Intuitively, for any input history each mobile function $f \in F$ represents one possible nondeterministic behavior. For any set of functions F we define $\mathcal{O}(F)$ to be the set of all behaviors of F, i.e., $\mathcal{O}(F) = \{(x, f(x)) \mid f \in F\}$.

Different sets of mobile functions may have the same set of behaviors. The reason is that for some sets of mobile functions we may find additional mobile functions which can be understood as combinations of the functions already in the set. For example, we may find a mobile function g which for one input history behaves as the function $f \in F$ and for another input history behaves as the function $f' \in F$, and so on. This means, a model in which a nondeterministic component is represented by an arbitrary set of mobile functions, is too distinguishing, and consequently, not *fully abstract*. To achieve full abstraction we consider only closed sets, i.e., sets F, where each combination of functions in F, which gives a mobile function, is also in F.

Definition 5 (Mobile components) *A mobile component, with initial wiring (I, O) and private names P, where $P \cap (I \cup O) = \emptyset$, is modeled by a nonempty set of stream processing functions*

$$F \subseteq Mob(I, O, P)$$

that is closed *in the sense that for any $f \in Mob(I, O, P)$*

$$(\forall \theta \in (N \to [M^*]) : \exists f' \in F : f(\theta) = f'(\theta)) \Rightarrow f \in F. \qquad \square$$

It follows straightforwardly that if F_1 and F_2 are mobile components then $F_1 = F_2$ iff $\mathcal{O}(F_1) = \mathcal{O}(F_2)$. Thus, our notion of a component is fully abstract with respect to the corresponding set of behaviors. Note the relationship to [Rus90]. That our semantics is fully abstract with respect to $\mathcal{O}$ is of course trivial. Nevertheless, this notion of observation characterizes the expectations one has to a semantics dealing with time.

Note that if $c \in N$ and $(\theta, \delta) \in \mathcal{O}(F)$ then the communication history $\theta(c)$ contains the history of all messages sent by the environment along the channel c. On the other hand, the communication history $\delta(c)$ contains the history of all messages sent by F along the channel c. Thus, although we model many-to-many communication, each component can be understood as a pure relation between input and output histories where each input history contains only messages sent by the environment, and each output history contains only messages sent by the component.

5 Interference and Implicit Merge

Interference occurs when two mobile components send on the same channel. As should be clear from the discussion in the previous section, interference is not considered at the

component level. This allows us to describe each component in a very abstract and, in our opinion, intuitive way. Instead, interference is modeled by building implicit merge components into the network operators. Each such merge component MC takes two named communication histories as input and yields their merge as output. Since MC is hidden in the semantics, it should neither add nor reduce delay. Remember, we want to be able to express timing constraints. This means that its output history during time unit k should be a merge of the two finite streams characterizing the input histories in time unit k. Moreover, MC should not fix the interleaving. Thus, any interleaving of the messages received within a time unit should be allowed. This means that MC is nondeterministic. MC is now formally defined in two steps.

We first characterize what it means for a finite stream to be a merge of two finite streams. To do so, we introduce some operators. For any tuple t we use $\pi_1(t)$ to denote the first component of t; for any stream s we use $\#s$ to denote the length of s. For a set of messages A and a stream of messages s, $A©s$ denotes the stream we obtain by removing any message in s that is not contained in A. This operator is overloaded to sets of pairs of messages $A \times B$ and pairs of streams of messages (r, s) in a straightforward way: for each j, $(r(j), s(j))$ is filtered away iff it is not in $A \times B$.

Definition 6 (The merge relation on finite streams) *Let FM be the function such that:*

$$FM \in M^* \times M^* \to \mathcal{P}(M^*)$$
$$FM(s_1, s_2) = \{s \mid \exists p \in \{1,2\}^* : \#p = \#s \wedge \forall i \in \{1,2\} : s_i = \pi_1((M \times \{i\}) \copyright (s, p))\}$$

□

It is now straightforward to define the implicit merge component.

Definition 7 (The merge component) *The* merge component *is a set of weakly guarded functions defined as follows:*

$$MC \subseteq (N \to [M^*]) \times (N \to [M^*]) \xrightarrow{w} (N \to [M^*])$$
$$MC = \{f \in (N \to [M^*]) \times (N \to [M^*]) \xrightarrow{w} (N \to [M^*]) \mid \forall \varphi, \psi, i : N, n \in \mathbb{N} : \; f(\varphi, \psi)(i)(n) \in FM(\varphi(i)(n), \psi(i)(n))\}$$

□

Note that the functions contained in MC are only required to be weakly guarded. Nevertheless, due to the way this implicit component is used in the definitions of network operators, this does not lead to any problem.

Theorem 4 *MC is a nonempty set.*

Proof: Trivial □

6 Network Operators

In this section we introduce three network operators, namely an operator for *parallel composition with mutual feedback*, a *feedback* operator and a *hiding* operator.

6.1 Many-to-Many Composition

The parallel composition with mutual feedback of two components F_1 and F_2 is characterized by the network in Figure 2.

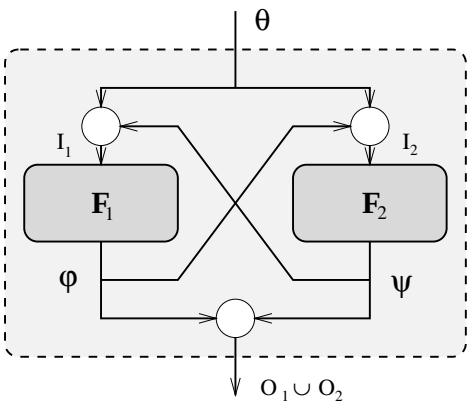

Figure 2: Many-to-many communication

Definition 8 (Many-to-many composition) *Given two mobile components*

$$F_1 \subseteq Mob(I_1, O_1, P_1), \quad F_2 \subseteq Mob(I_2, O_2, P_2),$$

where $P_1 \cap (P_2 \cup I_2 \cup O_2) = P_2 \cap (P_1 \cup I_1 \cup O_1) = \emptyset$. *Let*

$$I = I_1 \cup I_2, \quad O = O_1 \cup O_2, \quad P = P_1 \cup P_2,$$

then

$$F_1 \oplus F_2 \subseteq Mob(I, O, P)$$

$$F_1 \oplus F_2 = \{f \in Mob(I, O, P) \mid \forall \theta : \exists f_1 \in F_1, f_2 \in F_2, m_1, m_2, m_3 \in MC :$$
$$f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \delta) \text{ where}$$
$$\delta = m_3(\varphi, \psi), \ \vartheta = \mathsf{dom}_{I,O,P}(\theta, \delta)$$
$$\varphi = f_1(m_1(\vartheta, \psi)), \ \psi = f_2(m_2(\vartheta, \varphi))\}$$

□

Note the role of $\mathsf{dom}_{I,O,P}$ and $\mathsf{rng}_{I,O,P}$ in maintaining privacy. If F_1 sends a private send port $!p \in !P_1$ on a feedback channel (and it does not send $?p$), then both the environment and F_2 can send along p, but only F_1 is allowed to receive on p. As a consequence, the output of F_2 along p should not be observed by the environment. This is ensured by $\mathsf{rng}_{I,O,P}$. Similarly, if F_1 sends a private receive port $?p \in ?P_1$ on a feedback channel (and it does not send $!p$), then both the environment and F_2 can receive on p, but only F_1 is allowed to send along p. As a consequence, the input of F_2 on p should contain only messages sent by F_1. This is ensured by $\mathsf{dom}_{I,O,P}$.

Theorem 5 $F_1 \oplus F_2 \neq \emptyset$.

Proof: Since F_1 and F_2 are mobile components and MC is not empty we may find functions $f_1 \in F_1, f_2 \in F_2$ and $m_1, m_2, m_3 \in MC$. Based on these functions we construct a function f which is strongly guarded, generic and satisfies the recursive definition above.

Let g be defined as follows:

$$g \in ((N \to [M^*]) \times (N \to [M^*])) \times (N \to [M^*]) \to (N \to [M^*]) \times (N \to [M^*])$$
$$g((\varphi, \psi), \theta) = (f_1(m_1(\vartheta, \psi)), f_2(m_2(\vartheta, \varphi)), \quad \text{where} \quad \vartheta = \mathsf{dom}_{I,O,P}(\theta, \delta),\ \delta = m_3(\varphi, \psi).$$

Theorem 12 and the way g is defined in terms of strongly and weakly guarded functions imply that g is strongly guarded. Thus $\mu\, g$ is well-defined, in which case Theorem 14 implies that $\mu\, g$ is strongly guarded. That the function f defined below is also strongly guarded follows by a similar argument:

$$f \in (N \to [M^*]) \to (N \to [M^*])$$
$$f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \delta) \quad \text{where} \quad \delta = m_3(\varphi, \psi), \quad (\varphi, \psi) = (\mu\, g)(\theta)$$

That f is generic is a consequence of the next two lemmas.

Lemma 1 $f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$.

Proof: The idea of the proof is to transform $f(\theta)$ to $f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$ by using the equalities from Theorem 3. By definition

$$f(\mathsf{dom}_{I,O,P}(\theta, f(\theta))) = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta') \ \text{ where}$$
$$\delta' = m_3(\varphi', \psi'), \quad \vartheta' = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta'),$$
$$\varphi' = f_1(m_1(\vartheta', \psi')), \quad \psi' = f_2(m_2(\vartheta', \varphi')).$$

By Theorem 3 and definition of f

$$\begin{array}{lll}
\mathsf{dom}_{I,O,P}(\theta, \delta) & = \mathsf{dom}_{I,O,P}(\theta, \mathsf{rng}_{I,O,P}(\theta, \delta)) & = \mathsf{dom}_{I,O,P}(\theta, f(\theta)), \\
\mathsf{dom}_{I,O,P}(\theta, \delta) & = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, \delta), \delta) & = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta), \\
\mathsf{rng}_{I,O,P}(\theta, \delta) & = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, \delta), \delta) & = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta).
\end{array}$$

Replacing $\mathsf{dom}_{I,O,P}(\theta, \delta)$ and $\mathsf{rng}_{I,O,P}(\theta, \delta)$ in the definition of f we obtain

$$f(\theta) = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta) \ \text{ where}$$
$$\delta = m_3(\varphi, \psi), \quad \vartheta = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta),$$
$$\varphi = f_1(m_1(\vartheta, \psi)), \quad \psi = f_2(m_2(\vartheta, \varphi)).$$

Now the fact that we have unique fix-points implies that $\delta = \delta'$, $\varphi = \varphi'$, $\psi = \psi'$ and $\vartheta = \vartheta'$ and consequently that $f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$. □

Lemma 2 $f(\theta) = \mathsf{rng}_{I,O,P}(\theta, f(\theta))$.

Proof:

$$\begin{array}{ll}
\mathsf{rng}_{I,O,P}(\theta, f(\theta)) = & \\
\mathsf{rng}_{I,O,P}(\theta, \mathsf{rng}_{I,O,P}(\theta, \delta)) = & \{\text{by definition of } f\} \\
\mathsf{rng}_{I,O,P}(\theta, \delta) = & \{\text{by Theorem 3}\} \\
f(\theta) & \{\text{by definition of } f\}
\end{array}$$ □

Finally, since $\exists f_1, f_2, m_1, m_2, m_3 : \forall \theta : P$ implies $\forall \theta : \exists f_1, f_2, m_1, m_2, m_3 : P$ it follows that $f \in F_1 \oplus F_2$. □

Theorem 6 *$F_1 \oplus F_2$ is a mobile component.*

Proof: That $F_1 \oplus F_2 \neq \emptyset$ follows from Theorem 5. To see that $F_1 \oplus F_2$ is closed, let $f \in Mob(I, O, P)$, and assume that

$\forall\theta : \exists f' \in F_1 \oplus F_2 :\ f(\theta) = f'(\theta).$

The definition of $\oplus$ implies that for any θ there are $f' \in F_1 \oplus F_2$, $f_1 \in F_1, f_2 \in F_2$ and $m_1, m_2, m_3 \in MC$ such that:

$f(\theta) = f'(\theta) = \mathsf{rng}_{I,O,P}(\theta, \delta)$ where
$\varphi = f_1(m_1(\vartheta, \psi)),\quad \psi = f_2(m_2(\vartheta, \varphi)),\quad \vartheta = \mathsf{dom}_{I,O,P}(\theta, \delta),\quad \delta = m_3(\varphi, \psi)$

By the definition of $\oplus$, it follows that $f \in F_1 \oplus F_2$. □

6.2 Feedback

As we have already pointed out, if (θ, δ) is a behavior of a mobile component F then the communication history $\theta(c)$ contains the messages sent by the environment along the channel c, and $\delta(c)$ contains the messages sent by the component along the channel c. Clearly, the behavior of F has only indirect influence on its input via the environment. Thus, there is no direct interference. To allow the component's output to interfere with its input a simple feedback operator can be used. We then get the network pictured in Figure 3. Formally, the feedback operator can be defined as below:

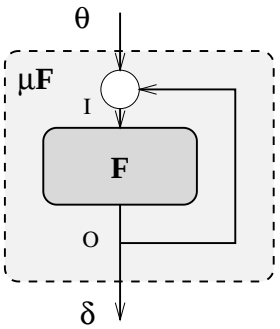

Figure 3: Feedback

Definition 9 (Feedback) *Given a mobile component $F \subseteq Mob(I, O, P)$. Then*

$\mu F \subseteq Mob(I, O, P)$

$\mu F = \{f \in Mob(I, O, P) \mid \forall\theta : \exists g \in F, m \in MC :$
$f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \delta) \quad \text{where} \quad \delta = g(m(\mathsf{dom}_{I,O,P}(\theta, \delta), \delta))\}.$ □

Note the close relationship between Figure 3 and the definition. If $I \cap O \neq \emptyset$ it is completely clear that interference will occur. However, interference can take place also if $I \cap O = \emptyset$. For example, this is the case if F receives a receive port $?c$ to a channel in O.

Theorem 7 $\mu F \neq \emptyset$.

Proof: Since F is a mobile component and MC is not empty we may find functions $g \in F$ and $m \in MC$. Based on these functions we construct a function f which is strongly guarded, generic and satisfies the recursive definition above. Let h be defined as follows:

$h \in (N \to [M^*]) \times (N \to [M^*]) \to (N \to [M^*])$

$h(\delta, \theta) = g(m(\mathsf{dom}_{I,O,P}(\theta, \delta), \delta))$

Theorem 12 and the way h is defined in terms of strongly and weakly guarded functions imply that h is strongly guarded. Thus $\mu\, h$ is well-defined, in which case Theorem 14 implies that $\mu\, h$ is strongly guarded. By a similar argument, f defined below is also strongly guarded:

$f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \delta) \quad \mathsf{where} \quad \delta = \mu\, h(\theta)$

That f is generic is a consequence of the next two lemmas.

Lemma 3 $f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$.

Proof: The idea of the proof is to transform $f(\theta)$ to $f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$ by using the equalities from Theorem 3. By definition

$f(\mathsf{dom}_{I,O,P}(\theta, f(\theta))) = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta')$ where
$\quad \delta' = g(m(\mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta'), \delta'))$.

By Theorem 3 and definition of f

$$\begin{array}{lll}
\mathsf{dom}_{I,O,P}(\theta, \delta) & = \mathsf{dom}_{I,O,P}(\theta, \mathsf{rng}_{I,O,P}(\theta, \delta)) & = \mathsf{dom}_{I,O,P}(\theta, f(\theta)) \\
\mathsf{dom}_{I,O,P}(\theta, \delta) & = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, \delta), \delta) & = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta) \\
\mathsf{rng}_{I,O,P}(\theta, \delta) & = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, \delta), \delta) & = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta).
\end{array}$$

Replacing $\mathsf{dom}_{I,O,P}(\theta, \delta)$ and $\mathsf{rng}_{I,O,P}(\theta, \delta)$ in the definition of f we obtain

$f(\theta) = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta)$ where
$\quad \delta = g(m(\mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta), \delta))$.

Now the fact that we have unique fix-points implies that $\delta = \delta'$ and consequently that $f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$. □

Lemma 4 $f(\theta) = \mathsf{rng}_{I,O,P}(\theta, f(\theta))$.

Proof:

$$\begin{array}{lll}
\mathsf{rng}_{I,O,P}(\theta, f(\theta)) = & & \\
\mathsf{rng}_{I,O,P}(\theta, \mathsf{rng}_{I,O,P}(\theta, \delta)) = & \{\text{by definition of } f\} & \\
\mathsf{rng}_{I,O,P}(\theta, \delta) = & \{\text{by Theorem 3}\} & \\
f(\theta) & \{\text{by definition of } f\} &
\end{array}$$ □

Finally, since $\exists g, m : \forall \theta : P$ implies $\forall \theta : \exists g, m : P$ it follows that $f \in \mu F$. □

Theorem 8 *μF is a mobile component.*

Proof: That $\mu F \neq \emptyset$ follows from Theorem 7. To see that μF is closed, let $f \in Mob(I, O, P)$, and assume that

$\forall \theta : \exists f' \in \mu F : \ f(\theta) = f'(\theta)$.

The definition of feedback implies that for any θ there are $f' \in \mu F$, $g \in F$ and $m \in MC$ such that:

$f(\theta) = f'(\theta) = \mathsf{rng}_{I,O,P}(\theta, \delta) \quad \mathsf{where} \quad \delta = g(m(\mathsf{dom}_{I,O,P}(\theta, \delta), \delta))\}$

By the definition of μ, it follows that $f \in \mu F$. □

6.3 Hiding

The privacy of a non-static port is guaranteed by the genericity property. The privacy of a static port, i.e., a port from the initial wiring, can be ensured by using a hiding operator. If x is a port from the initial wiring of the component F, then $\nu x : F$ is a component where x is added to the set of private channel names and deleted from the static interface. The domain and range of the functions modeling $\nu x : F$ are changed accordingly. As a consequence, only components receiving $!x$ or $?x$ as a message can access this channel later on.

Definition 10 (Hiding) *Given a mobile component $F \subseteq Mob(I', O', P')$ and a channel name x such that $x \notin P'$. Let:*

$$I = I' \setminus \{x\}, \quad O = O' \setminus \{x\}, \quad P = P' \cup \{x\}.$$

Then:

$$\begin{aligned} \nu x : F &\subseteq Mob(I, O, P) \\ \nu x : F &= \{f \in Mob(I, O, P) \mid \forall \theta : \exists g \in F : \\ &\quad f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \delta) \quad \mathsf{where} \quad \delta = g(\mathsf{dom}_{I,O,P}(\theta, \delta))\} \end{aligned}$$

□

Note the role of $\mathsf{dom}_{I,O,P}$ and $\mathsf{rng}_{I,O,P}$ in maintaining privacy: $\mathsf{dom}_{I,O,P}$ makes sure that the behavior of $\nu x : F$ is independent of what the environment sends along x before the environment has received the send port $!x$; $\mathsf{rng}_{I,O,P}$ makes sure that $\nu x : F$ does not send messages along x before it has sent the receive port $?x$.

Theorem 9 $\nu\, x : F \neq \emptyset$.

Proof: Since F is a mobile component we may find a strongly guarded function $g \in F$. Based on g we construct a function f which is strongly guarded, generic and satisfies the recursive definition above.

Let h be defined as follows:

$$h(\delta, \theta) = g(\mathsf{dom}_{I,O,P}(\theta, \delta))$$

Theorem 12 and the way h is defined in terms of strongly and weakly guarded functions imply that h is strongly guarded. Thus $\mu\, h$ is well-defined, in which case Theorem 14 implies that $\mu\, h$ is strongly guarded. By a similar argument, f defined below is also strongly guarded:

$$f(\theta) = \mathsf{rng}_{I,O,P}(\theta, \delta) \quad \mathsf{where} \quad \delta = \mu\, g(\theta)$$

That f is generic is a consequence of the next two lemmas.

Lemma 5 $f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$.

Proof: The idea of the proof is to transform $f(\theta)$ to $f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$ by using the equalities from Theorem 3. By definition

$$\begin{aligned} &f(\mathsf{dom}_{I,O,P}(\theta, f(\theta))) = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta') \quad \mathsf{where} \\ &\quad \delta' = g(\mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)), \delta')). \end{aligned}$$

By Theorem 3 and definition of f

$$\begin{array}{lll}
\mathsf{dom}_{I,O,P}(\theta,\delta) & = \mathsf{dom}_{I,O,P}(\theta, \mathsf{rng}_{I,O,P}(\theta,\delta)) & = \mathsf{dom}_{I,O,P}(\theta, f(\theta)) \\
\mathsf{dom}_{I,O,P}(\theta,\delta) & = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta) & = \mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)),\delta) \\
\mathsf{rng}_{I,O,P}(\theta,\delta) & = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta,\delta),\delta) & = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)),\delta).
\end{array}$$

Replacing $\mathsf{dom}_{I,O,P}(\theta,\delta)$ and $\mathsf{rng}_{I,O,P}(\theta,\delta)$ in the definition of f we obtain

$$f(\theta) = \mathsf{rng}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)),\delta) \quad \mathsf{where}$$
$$\delta = g(\mathsf{dom}_{I,O,P}(\mathsf{dom}_{I,O,P}(\theta, f(\theta)),\delta)).$$

Now the fact that we have unique fix-points implies that $\delta = \delta'$ and consequently that $f(\theta) = f(\mathsf{dom}_{I,O,P}(\theta, f(\theta)))$. □

Lemma 6 $f(\theta) = \mathsf{rng}_{I,O,P}(\theta, f(\theta))$.

Proof:

$$\begin{array}{ll}
\mathsf{rng}_{I,O,P}(\theta, f(\theta)) = & \\
\mathsf{rng}_{I,O,P}(\theta, \mathsf{rng}_{I,O,P}(\theta,\delta)) = & \{\text{by definition of } f\} \\
\mathsf{rng}_{I,O,P}(\theta,\delta) = & \{\text{by Theorem 3}\} \\
f(\theta) & \{\text{by definition of } f\}
\end{array}$$ □

Finally, since $\exists g : \forall\theta : P$ implies $\forall\theta : \exists g : P$ it follows that $f \in \nu x : F$. □

Theorem 10 *$\nu x : F$ is a mobile component.*

Proof: That $\nu x : F \neq \emptyset$ follows from Theorem 9. To see that $\nu x : F$ is closed, let $f \in Mob(I,O,P)$, and assume that

$$\forall\theta : \exists f' \in \nu x : F : \ f(\theta) = f'(\theta).$$

The definition of hiding implies that for any θ there are $f' \in \nu x : F$ and $g \in F$ such that:

$$f(\theta) = f'(\theta) = \mathsf{rng}_{I,O,P}(\theta,\delta) \quad \mathsf{where} \quad \delta = g(\mathsf{dom}_{I,O,P}(\theta,\delta))\}$$

By the definition of ν, it follows that $f \in \nu x : F$. □

7 Mobile Telephones

Our denotational model is very expressive. In this section we show how we can deal with the mobile telephone example discussed in [Mil91]. A center is in permanent contact with two base stations; each in a different part of the country. A car with a mobile telephone moves about the country; it should always be in contact with a base. If it gets rather far from its current base contact, then a hand-over procedure is initiated, and as a result the car relinquishes contact with one base and assumes contact with another. The initial configuration of the network is illustrated by Figure 4.

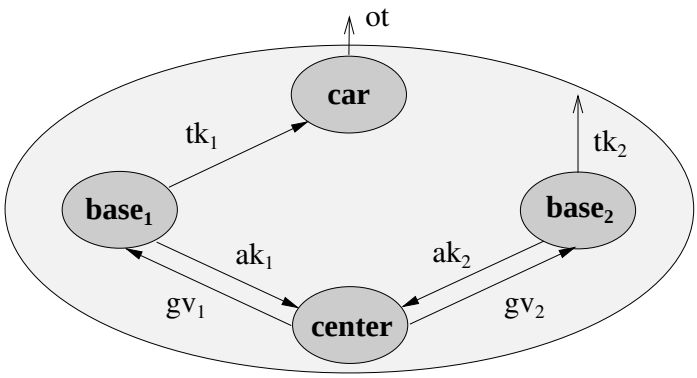

Figure 4: The initial system configuration

In this example there are no real-time constraints to be imposed. We therefore introduce a *time abstraction* operator: the untimed stream tuple $\widehat{\theta}$ is obtained from the timed stream tuple θ by removing time information. This is achieved by concatenating for each $i \in N$ all the finite sequences in $\theta(i)$. The operation $\{c \to m\} \,\&\, \varphi$ appends the message m to the head of the stream $\varphi(c)$. The other streams in φ remain unchanged. This operator is overloaded to streams in the obvious way.

Figure 4 shows the initial configuration of system. car is in contact with base_1. However, there is no connection between car and base_2. The system is specified as follows:

$$\mathsf{system}(\triangleright ot) \stackrel{\mathsf{def}}{=} \nu\, tk_1, tk_2, gv_1, gv_2, ak_1, ak_2 :$$
$$\mathsf{car}(tk_1 \triangleright ot) \oplus \mathsf{base}(gv_1 \triangleright tk_1, ak_1) \oplus \mathsf{base}(gv_2 \triangleright tk_2, ak_2) \oplus$$
$$\mathsf{center}(tk_1, tk_2)(ak_1, ak_2 \triangleright gv_1, gv_2)$$

The car is parametric upon a receive port tk and a send port ot. On tk it can either receive talk messages m or switch messages $?x$. Any talk message is forwarded along ot. The arrival of a receive port $?c$ forces the component to receive on $?c$ instead of on tk.

$$\mathsf{car}(tk \triangleright ot) \stackrel{\mathsf{def}}{=} \{\ f \in Mob(\{tk\}, \{ot\}, \emptyset) \mid$$
$$\forall \theta :\ \widehat{f(\theta)} = g(tk)(\widehat{\theta}) \quad \textbf{where} \quad \forall \vartheta, m, c :$$
$$g(tk)(\{tk \mapsto m\} \,\&\, \vartheta) = \quad \{ot \mapsto m\} \,\&\, g(tk)(\vartheta)$$
$$g(tk)(\{tk \mapsto ?c\} \,\&\, \vartheta) = \quad g(c)(\vartheta) \quad \}$$

A base can talk repeatedly with the car; but at any time it can receive on its give channel gv a new port which it should communicate to the car and then become idle itself. Whether it ignores the give channel or not is controlled with a prophecy stream. An idle base is reactivated upon receipt of an act message on its give channel.

$$\mathsf{base}(gv \triangleright tk, ak) \stackrel{\mathsf{def}}{=} \{\ f \in Mob(\{gv\}, \{tk, ak\}, \emptyset) \mid$$
$$\forall \theta : \exists p \in [\{1,2\}] :\ \widehat{f(\theta)} = h(p)(\widehat{\theta}) \quad \textbf{where} \quad \forall p, \vartheta, x :$$
$$h(p)(\{gv \mapsto \mathsf{act}\} \,\&\, \vartheta) = \quad g(p)(\vartheta)$$
$$g(1 \,\&\, p)(\vartheta) = \quad \{tk \mapsto m\} \,\&\, g(p)(\vartheta)$$
$$g(2 \,\&\, p)(\{gv \mapsto ?x\} \,\&\, \vartheta) = \quad \{tk \mapsto ?x, ak \mapsto \mathsf{ok}\} \,\&\, h(p)(\vartheta) \quad \}$$

The center which initially knows that the car is in connection with base_1, can decide (according to information which we do not model) to transmit the receive port $?tk_2$ to the car via base_1. Upon receipt of an acknowledgment on ak_1 from base_1 it alerts base_2 of this fact.

$$\begin{aligned}
&\mathsf{center}(tk_1, tk_2)(ak_1, ak_2 \rhd gv_1, gv_2) \stackrel{\mathsf{def}}{=} \{\ f \in Mob(\{ak_1, ak_2\}, \{gv_1, gv_2\}, \emptyset) \mid \\
&\quad \forall \theta :\ \widehat{f(\theta)} = g(1)(\widehat{\theta}) \quad \textbf{where} \quad \forall i \in \{1,2\}, \vartheta \\
&\qquad g(i)(\vartheta) = \qquad\qquad \{gv_i \mapsto \mathsf{act}\} \,\&\, \{gv_i \mapsto ?tk_{\neg i}\} \,\&\, h(i)(\vartheta) \\
&\qquad h(i)(\{ak_i \mapsto \mathsf{ok}\} \,\&\, \vartheta) = \quad g(\neg i)(\vartheta)\ \}
\end{aligned}$$

It is here assumed that $\neg 1 = 2$ and that $\neg 2 = 1$.

8 Discussion

Our main contribution is that we have extended a denotational model for timed, many-to-many, nondeterministic data-flow networks to handle *mobility*. Our model is fully compositional. It allows us to reason about mobility at a very abstract level. In fact, we believe our semantics is well-suited as a foundation for a method for the specification and development of mobile systems. The exact relationship between our model and other models like for instance the π-calculus [Mil91] and actor-based approaches [AMST92] is an interesting area for future research. For example, we believe that our model can be used to give a denotational semantics for the asynchronous π-calculus. We also believe that the actor languages can be smoothly integrated within our formalism.

Our semantics can easily be adapted to model other kinds of communication. For example, in [GS96b, GS96a] we show how two different kinds of point-to-point communication can be modeled in our approach.

Our approach is related to the work of Kok [Kok87, Kok89]. The major difference is that Kok does not deal with mobility. Moreover, his handling of nondeterminism differs from ours. In [Kok89], where he uses a metric on relations, he can basically handle only bounded nondeterminism. In [Kok87], which is not based on metric spaces, an automaton is used to generate the behavior of basic agents. This guarantees the existence of fix-points. We use sets of strongly guarded functions for the same purpose.

References

[AdBKR89] P. America, J. de Bakker, J. N. Kok, and J. Rutten. Denotational semantics of a parallel object-oriented language. *Information and Computation*, 83:152–205, 1989.

[AMST92] G. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. Towards a theory of actor computation. In *Proc. CONCUR'92, Lecture Notes in Computer Science 630*, pages 565–579, 1992.

[BA81] J. D. Brock and W. B. Ackermann. Scenarios: A model of non-determinate computation. In *Proc. Formalization of Programming Concepts, Lecture Notes in Computer Science 107*, pages 252–259, 1981.

[BB90] G. Berry and G. Boudol. The chemical abstract machine. In *Proc. POPL'90*, pages 81–94, 1990.

[BDD+93] M. Broy, F. Dederichs, C. Dendorfer, M. Fuchs, T. F. Gritzner, and R. Weber. The design of distributed systems — an introduction to Focus (revised version). Technical Report SFB 342/2/92 A, Technische Universität München, 1993.

[Bro87] M. Broy. Semantics of finite and infinite networks of concurrent communicating agents. *Distributed Computing*, 2:13–31, 1987.

[CM88] K. M. Chandy and J. Misra. *Parallel Program Design, A Foundation.* Addison-Wesley, 1988.

[dBZ82] J. W. de Bakker and J. I. Zucker. Denotational semantics of concurrency. In *Proc. 14 ACM Symposium on Theory of Computing*, pages 153–158, 1982.

[EN86] U. Engberg and M Nielsen. A calculus of communicating systems with label-passing. Technical Report DAIMI PB-208, University of Aarhus, 1986.

[Eng77] R. Engelking. *General Topology.* PWN — Polish Scientific Publishers, 1977.

[GS96a] R. Grosu and K. Stølen. A denotational model for mobile point-to-point dataflow networks with channel sharing. To appear as technical report, Technische Universität München, 1996.

[GS96b] R. Grosu and K. Stølen. A model for mobile point-to-point dataflow networks without channel sharing. To appear in the Proc. of AMAST'96, 1996.

[HBS73] C. Hewitt, P. Bishop, and R. Steiger. A universal modular actor formalism for artificial intelligence. In *Proc. IJCAI'73*, pages 235–245, 1973.

[Kah74] G. Kahn. The semantics of a simple language for parallel programming. In *Proc. Information Processing 74*, pages 471–475. North-Holland, 1974.

[Kel78] R. M. Keller. Denotational models for parallel programs with indeterminate operators. In *Proc. Formal Description of Programming Concepts*, pages 337–366. North-Holland, 1978.

[Kok87] J. N. Kok. A fully abstract semantics for data flow nets. In *Proc. PARLE'87, Lecture Notes in Computer Science 259*, pages 351–368, 1987.

[Kok89] J. N. Kok. An iterative metric fully abstract semantics for nondeterministic dataflow. In *Proc. MFCS'89, Lecture Notes in Computer Science 379*, pages 321–331, 1989.

[Lam91] L. Lamport. The temporal logic of actions. Technical Report 79, Digital, SRC, Palo Alto, 1991.

[Mes91] J. Meseguer. Conditional rewriting logic as a unified model of concurrency. Technical Report SRI-CSL-91-05, SRI, 1991.

[Mil91] R. Milner. The polyadic π-calculus: A tutorial. Technical Report ECS-LFCS-91-180, University of Edinburgh, 1991.

[MPS86] D. MacQueen, G. Plotkin, and R. Sethi. An ideal model for recursive polymorphic types. *Information and Control*, 71:95–130, 1986.

[MPW92a] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part I. *Information and Computation*, 100:1–40, 1992.

[MPW92b] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part II. *Information and Computation*, 100:41–77, 1992.

[Niv82] M Nivat. Behaviours of processes and synchronized systems of processes. In *Proc. Theoretical Foundations of Programming Methodology*, pages 473–551, 1982.

[Par83] D. Park. The "fairness" problem and nondeterministic computing networks. In *Proc. 4th Foundations of Computer Science, Mathematical Centre Tracts 159*, pages 133–161. Mathematisch Centrum Amsterdam, 1983.

[PS92] P. Panangaden and V. Shanbhogue. The expressive power of indeterminate dataflow primitives. *Information and Computation*, 98:99–131, 1992.

[Rus90] J. R. Russell. On oraclizable networks and Kahn's principle. In *Proc. POPL'90*, pages 320–328, 1990.

[Tho89] B. Thomsen. A calculus of higher order communicating systems. In *Proc. POPL'89*, 1989.

A Metric Space Definitions

A.1 Metric Space Basics

The fundamental concept in metric spaces is the concept of distance.

Definition 11 (Metric Space) *A* metric space *is a pair (D, d) consisting of a nonempty set D and a mapping $d \in D \times D \to R$, called a* metric *or* distance, *which has the following properties:*

(1) $\forall x, y \in D: \quad d(x, y) = 0 \quad \Leftrightarrow \quad x = y$
(2) $\forall x, y \in D: \quad d(x, y) = d(y, x)$
(3) $\forall x, y, z \in D: \quad d(x, y) \leq d(x, z) + d(z, y)$ □

A very simple example of a metric is the discrete metric.

Definition 12 (The discrete metric) *The* discrete metric (D, d) *over a set* D *is defined as follows:*

$$d(x,y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$$ □

Measuring the distance between the elements of a sequence $(x_i)_{i\in\mathbb{N}}$ in D we obtain the familiar definitions for convergence and limits.

Definition 13 (Convergence and limits) *Let* (D, d) *be a metric space and let* $(x_i)_{i\in\mathbb{N}}$ *be a sequence in* D*.*

(1) *We say that* $(x_i)_{i\in\mathbb{N}}$ *is a* Cauchy sequence *whenever we have:*

$\forall \epsilon > 0 : \exists N \in \mathbb{N} : \forall n, m > N : d(x_n, x_m) < \epsilon.$

(2) *We say that* $(x_i)_{i\in\mathbb{N}}$ converges to $x \in D$ *denoted by* $x = lim_{n\to\infty} x_i$ *and call* x *the* limit *of* $(x_i)_{i\in\mathbb{N}}$ *whenever we have:*

$\forall \epsilon > 0 : \exists N \in \mathbb{N} : \forall n > N : d(x_n, x) < \epsilon.$

(3) *The metric space* (D, d) *is called* complete *whenever each Cauchy sequence converges to an element of* D*.*

□

Theorem 11 *The discrete metric is complete.*

Proof: Each Cauchy sequence is constant from a given N. □

A very important class of functions over metric spaces is the class of *Lipschitz functions*.

Definition 14 (Lipschitz functions) *Let* (D_1, d_1) *and* (D_2, d_2) *be metric spaces and let* $f \in D_1 \to D_2$ *be a function. We call* f Lipschitz function *with constant* c *if there is a constant* $c \geq 0$ *such that the following condition is satisfied:*

$d(f(x), f(y)) \leq c \cdot d(x, y)$

For a function f *with arity* n *the above condition generalizes to:*

$d(f(x_1, \ldots, x_n), f(y_1, \ldots, y_n)) \leq c \cdot max\{d(x_i, y_i) \mid i \in [1..n]\}$

If $c = 1$ *we call* f non-expansive. *If* $c < 1$ *we call* f contractive. □

Theorem 12 *The composition of two Lipschitz functions* $f \in D_1 \to D_2$ *and* $g \in D_2 \to D_3$ *is a Lipschitz function with constant* $c_1 \cdot c_2$*.*

Proof: $d(g(f(x_1)), g(f(x_2)) \leq c_2 \cdot d(f(x_1), f(x_2)) \leq c_2 \cdot c_1 \cdot d(x_1, x_2)$ □

Lemma 1 *The composition of a contractive and a non-expansive function is contractive. The composition of two non-expansive functions is non-expansive. Identity is non-expansive.* □

The main tool for handling recursion in metric spaces is the Banach's fixed point theorem. It guarantees the existence of a unique fixed point for every contractive function.

Theorem 13 (Banach's fixed point theorem) *Let (D, d) be a complete metric space and $f \in D \to D$ a contractive function. Then there exists an $x \in D$, such that the following holds:*

(1) $x = f(x)$ *(x is a fixed point of f)*
(2) $\forall y \in D: \; y = f(y) \Rightarrow y = x$ *(x is unique)*
(3) $\forall z \in D: \; x = lim_{n\to\infty} f^n(z)$ *where*
$f^0(z) = z$
$f^{n+1}(z) = f(f^n(z))$

Proof: See [Eng77]. □

Usually we want to use a parameterized version of this theorem.

Definition 15 (Parameterized fixed point) *Let $f \in D \times D_1 \times \ldots \times D_n \to D$ be a function of non-empty complete metric spaces that is contractive in its first argument. We define the* parameterized fixed point *function μf as follows:*

$(\mu f) \in D_1 \times \ldots \times D_n \to D$
$(\mu f)(y_1, \ldots, y_n) = x$

where x is the unique element of D such that $x = f(x, y_1, \ldots, y_n)$ as guaranteed by Banach's fixed point theorem. □

Theorem 14 *If f is contractive (non-expansive) so is μf.*

Proof: See for example [MPS86] pages 114–115. □

A.2 Streams and Named Stream Tuples

A stream is a finite or infinite sequence of elements. For any set of elements E, we use E^* to denote the set of all finite streams over E, and $[E]$ to denote the set of all infinite streams over E. For any infinite stream s, we use $s{\downarrow_j}$ to denote the prefix of s containing exactly j elements. We use ϵ to denote the empty stream.

We define the metric of streams generically with respect to an arbitrary discrete metric (E, ρ).

Definition 16 (The metric of streams) *The metric of streams $([E], d)$ over a discrete metric (E, ρ) is defined as follows:*

$[E] = \prod_{i\in\mathbb{N}} E$
$d(s,t) = inf\{2^{-j} \mid s{\downarrow_j} = t{\downarrow_j}\}$

□

This metric is also known as the Baire metric [Eng77].

Theorem 15 *The metric space of streams $([E], d)$ is complete.*

Proof: See for example [Eng77]. □

A *named stream tuple* is a mapping $\theta \in (I \to [E])$ from a set of names to infinite streams. $\downarrow$ is overloaded to named stream tuples in a point-wise style, i.e., $\theta{\downarrow_j}$ denotes the result of applying $\downarrow_j$ to each component of θ.

Definition 17 (The metric of named stream tuples) *The metric of* named stream tuples $(I \to [E], d)$ *with names in* I *and elements in* (E, ρ) *is defined as follows:*

$I \to [E]$ *is the set of functions from the countable set* I *to the metric* $[E]$,

$d(s,t) = inf\{2^{-j} \mid s{\downarrow_j} = t{\downarrow_j}\}$

□

Theorem 16 *The metric space of named stream tuples* $(I \to [E], d)$ *is complete.*

Proof: This metric is equivalent to the Cartesian product metric $\prod_{i \in I}[E]$ which is complete because $[E]$ is [Eng77]. □

SFB 342: Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

bisher erschienen :

Reihe A

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

Reihe A

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Reihe A

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Reihe A

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

Reihe A

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Reihe A

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Reihe A

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Reihe A

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

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

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

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

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

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

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

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

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

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

342/21/95 A Andreas Listl, Giannis Bozas: Performance Gains Using Subpages for Cache Coherency Control

342/22/95 A Volker Heun, Ernst W. Mayr: Embedding Graphs with Bounded Treewidth into Optimal Hypercubes

342/23/95 A Petr Jančar, Javier Esparza: Deciding Finiteness of Petri Nets up to Bisimulation

342/24/95 A M. Jung, U. Rüde: Implicit Extrapolation Methods for Variable Coefficient Problems

342/01/96 A Michael Griebel, Tilman Neunhoeffer, Hans Regler: Algebraic Multigrid Methods for the Solution of the Navier-Stokes Equations in Complicated Geometries

342/02/96 A Thomas Grauschopf, Michael Griebel, Hans Regler: Additive Multilevel-Preconditioners based on Bilinear Interpolation, Matrix Dependent Geometric Coarsening and Algebraic-Multigrid Coarsening for Second Order Elliptic PDEs

342/03/96 A Volker Heun, Ernst W. Mayr: Optimal Dynamic Edge-Disjoint Embeddings of Complete Binary Trees into Hypercubes

342/04/96 A Thomas Huckle: Efficient Computation of Sparse Approximate Inverses

342/05/96 A Thomas Ludwig, Roland Wismüller, Vaidy Sunderam, Arndt Bode: OMIS — On-line Monitoring Interface Specification

342/06/96 A Ekkart Kindler: A Compositional Partial Order Semantics for Petri Net Components

342/07/96 A Richard Mayr: Some Results on Basic Parallel Processes

342/08/96 A Ralph Radermacher, Frank Weimer: INSEL Syntax-Bericht

Reihe A

342/09/96 A P.P. Spies, C. Eckert, M. Lange, D. Marek, R. Radermacher, F. Weimer, H.-M. Windisch: Sprachkonzepte zur Konstruktion verteilter Systeme

342/10/96 A Stefan Lamberts, Thomas Ludwig, Christian Röder, Arndt Bode: PFS-Lib – A File System for Parallel Programming Environments

342/11/96 A Manfred Broy, Gheorghe Ştefănescu: The Algebra of Stream Processing Functions

342/12/96 A Javier Esparza: Reachability in Live and Safe Free-Choice Petri Nets is NP-complete

342/13/96 A Radu Grosu, Ketil Stølen: A Denotational Model for Mobile Many-to-Many Data-flow Networks

SFB 342 : Methoden und Werkzeuge für die Nutzung paralleler Rechnerarchitekturen

Reihe B

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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