TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

A Denotational Model for Mobile Point-to-Point Dataflow Networks

Radu Grosu, Ketil Stølen

TUM-I9527
SFB-Bericht Nr.342/14/95 A
Oktober 1995

TUM–INFO–10-95-I27-0/1.–FI

Anforderungen an: Prof. Dr. A. Bode
Sprecher SFB 342
Institut für Informatik
Technische Universität München
Arcisstr. 21 / Postfach 20 24 20
D-80290 München, Germany

Druck: Fakultät für Informatik der
Technischen Universität München

A Denotational Model for Mobile Point-to-Point Dataflow Networks[*]

Radu Grosu, Ketil Stølen

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

October 10, 1995

---

[*] The first author has been financially supported by the DF

Abstract

We present a denotational model for mobile, timed, unbounded nondeterministic dataflow networks whose components communicate in a point-to-point fashion. We first introduce a model for static, point-to-point dataflow networks. In this model components and networks of components are represented by sets of strongly pulse-driven stream processing functions. A stream processing function is strongly pulse-driven if its output until time $j + 1$ is completely determined by its input until time j. This model is then extended to support mobility by allowing the components to communicate ports. In the mobile case, the functions are not only required to be strongly pulse-driven, but also to be generic in the sense that they do not read or write on channels whose ports they have not received or generated themselves. The model is shown to be fully abstract. We demonstrate the power and applicability of our model by specifying a mobile communication central.

Contents

Chapter 1

Introduction

Mobile systems are systems in which every component may change its communication partners on the basis of computation and interaction. In the last few years the formal modeling of mobile systems has received considerable interest. Examples of models expressing mobility are the Actor Model [HBS73, AMST92], the π–Calculus [EN86, MPW92a, MPW92b], the Chemical Abstract Machine [BB90], the Rewriting Logic [Mes91], the Higher Order CCS [Tho89] and the Concurrent Constraint Programming Languages [SRP91].

Most approaches published so far are operational. For example, with the exception of [SRP91], all the approaches mentioned above are of a very operational nature. In this paper we give a denotational model for timed, unbounded nondeterministic, mobile dataflow networks whose components communicate in a point-to-point fashion. Our model can be seen as an extension of the model for static deterministic dataflow networks proposed by Kahn [Kah74], and the models for static nondeterministic dataflow networks proposed by Park [Par83], Kok [Kok87] and Broy [Bro87].

In our model, a complete communication history for a channel is represented by an infinite stream of finite streams of messages. A complete communication history for a set of channels is represented by a named communication history. In the style of [BD92], a named communication history is a function mapping channel names to complete communication histories. We represent components by sets of stream processing functions mapping named communication histories for the input channels to named communication histories for the output channels.

These functions are required to satisfy two properties. First of all, they have to be strongly pulse-driven in the sense that the input until time j completely determines the output until time $j+1$. Due to this constraint, any network of deterministic components has a unique fixed point with respect to a complete communication history for its external input channels. Secondly, since mobility is achieved by allowing the functions to communicate ports, we have to make sure that the functions do not write or read on channels whose ports they do not know or have not created themselves. More explicitly, the functions are only allowed to read and write on channels dynamically generated on the basis of their initial wiring. A function which is well-behaved in this sense is said to be generic.

Our model is compositional with respect to the operator for network composition. We also prove that our model is fully abstract with respect to a notion of observation.

As we see it, the contribution of this paper is two-fold. Firstly, we provide a denotational understanding of mobility. This in contrast to the very operational nature of most other models. Secondly, point-to-point communication is imposed semantically. Thus, contrary to for example [SRP91], we do not need additional syntactic constraints to avoid inconsistencies.

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 pulse-drivenness constraints — strong and weak pulse-drivenness — correspond straightforwardly to contractiveness and non-expansiveness.

The rest of the paper is split into six chapters. Chapter 2 introduces our basic formalism and briefly shows how it can be used to model static point-to-point dataflow networks. Then, in Chapter 3 we extend our static model to deal with mobility. An example is given in Chapter 4, and in Chapter 5 we define a notion of observation and prove full abstractness with respect to this notion. Chapter 6 relates our approach to other approaches known from the literature. Finally, there is an appendix reviewing some basic stuff on metric spaces and streams.

Chapter 2

Static Data

channels of some component, then θ gives a complete communication history for its input channels.

2.2 Operations on Named Communication Histories

In this section we define two non-expansive operators on named communication histories, namely *summation* and *projection.* The summation operator constructs the union of two named communication histories with disjoint domains. The projection operator projects a named communication history on a given set of channel names.

Definition 1 (Summation) *For any* I, J *such that* $I \cap J = \emptyset$, *we define*

$$+ \in (I \to [D^*]) \times (J \to [D^*]) \to ((I \cup J) \to [D^*])$$

as follows:

$$(\theta + \varphi)(i) = \begin{cases} \theta(i) & \textit{if } i \in I, \\ \varphi(i) & \textit{if } i \in J. \end{cases}$$

□

Theorem 1 *The summation operator is is non-expansive.*

Proof: Given $\theta_1, \theta_2 \in I \to [D^*]$ and $\varphi_1, \varphi_2 \in J \to [D^*]$ with $I \cap J = \emptyset$. Then:

$$d(\theta_1 + \varphi_1, \theta_2 + \varphi_2) = inf\,\{2^{-j} \mid \theta_1{\downarrow_j} = \theta_2{\downarrow_j} \wedge \varphi_1{\downarrow_j} = \varphi_2{\downarrow_j}\} \leq max\{d(\theta_1, \theta_2), d(\varphi_1, \varphi_2)\}.$$

□

Definition 2 (Projection) *We define*

$$| \in (I \to [D^*]) \times O \to ((I \cap O) \to [D^*])$$

as follows:

$$(\theta|_O)(i) = \theta(i) \quad \textit{if } i \in (I \cap O).$$

□

Theorem 2 *The projection operator is non-expansive in its first argument.*

Proof: Given $\theta, \varphi \in I \to [D^*]$ and some set of channel names J. Then:

$$d(\theta|_J, \varphi|_J) = inf\,\{2^{-j} \mid (\theta|_J){\downarrow_j} = (\varphi|_J){\downarrow_j}\} \leq inf\,\{2^{-j} \mid \theta{\downarrow_j} = \varphi{\downarrow_j}\}.$$

□

2.3 Static Components

We model components by sets of *stream processing functions.* A deterministic component, whose input and output channels are named by respectively I and O, is modeled by a stream processing function

$$f \in (I \to [D^*]) \to (O \to [D^*])$$

mapping named communication histories for its input channels to 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 informing the function that time has progressed. The lack of this information causes the famous fair merge anomaly [Kel78].

A stream processing function is said to be *weakly pulse-driven* if its input until time j completely determines its output until time j. It is said to be *strongly pulse-driven* if its input until time j completely determines its output until time $j + 1$. Formally:

Definition 3 (Pulse-driven functions) *A function* $f \in (I \to [D^*]) \to (O \to [D^*])$ *is* weakly pulse-driven *if*

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

and strongly pulse-driven *if*

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

□

We use the arrow $\xrightarrow{}$ to distinguish strongly pulse-driven functions from functions that are not strongly pulse-driven.

A weakly pulse-driven function is non-expansive and a strongly pulse-driven function is contractive with respect to the metric on streams. As a consequence, strong pulse-drivenness not only replaces the usual monotonicity and continuity constraints of domain theory but also guarantees unique fixed points of feedback loops.

Theorem 3 *A stream processing function is strongly pulse-driven iff it is contractive with respect to the metric of streams. A stream processing function is weakly pulse-driven iff it is non-expansive with respect to the metric of streams.*

Proof:

(⇒) : Suppose that $d(x, y) = 2^{-k}$ and that f is strongly pulse-driven. Then by definition:

$$d(f(x), f(y)) = inf\{2^{-m} \mid f(x){\downarrow_m} = f(y){\downarrow_m}\} \leq 2^{-(k+1)} = (1/2) \cdot d(x, y).$$

(⇐) : Suppose that $d(f(x), f(y)) = 2^{-l}$ and that f is contractive i.e.

$$\exists c < 1 : \forall x, y : \; d(f(x), f(y)) \leq c \cdot d(x, y).$$

Then $2^{k-l} \leq c < 1 = 2^0$. This implies that $k - l \leq -1$ i.e. that $l \geq k + 1$. As a consequence

$$x{\downarrow_k} = y{\downarrow_k} \quad \Rightarrow \quad f(x){\downarrow_{k+1}} = f(y){\downarrow_{k+1}}.$$

In other words, f is strongly pulse driven. The second equivalence is proven accordingly.

□

Definition 4 (Static components) *A static component whose input and output channels are named by I and O, respectively, is modeled by a nonempty set of stream processing functions*

$$F \subseteq (I \to [D^*]) \xrightarrow{} (O \to [D^*]),$$

that is closed *in the sense that for any strongly pulse-driven function f of the same sig-*

nature

$$(\forall \theta \in (I \rightarrow [D^*]) : \exists f' \in F : f(\theta) = f'(\theta)) \Rightarrow f \in F.$$

$\square$

The above definition is very powerful. It not only makes our model *fully abstract* (see Chapter 5), but it also allows us to handle *unbounded nondeterminism*.

2.4 Point-to-Point Composition

We now introduce an operator $\otimes$ which allows components to be composed into networks of components — networks, which when observed from the outside, themselves behave as components.

Definition 5 (Point-to-point composition) *Given two static components:*

$$F_1 \subseteq (I_1 \rightarrow [D^*]) \rightharpoonup (O_1 \rightarrow [D^*]), \qquad F_2 \subseteq (I_2 \rightarrow [D^*]) \rightharpoonup (O_2 \rightarrow [D^*]),$$

where $I_1 \cap O_1 = I_2 \cap O_2 = I_1 \cap I_2 = O_1 \cap O_2 = \emptyset$. *Let*

$$I = (I_1 \setminus O_2) \cup (I_2 \setminus O_1), \qquad O = (O_1 \setminus I_2) \cup (O_2 \setminus I_1),$$

we define

$$F_1 \otimes F_2 \subseteq (I \rightarrow [D^*]) \rightharpoonup (O \rightarrow [D^*]),$$

$$F_1 \otimes F_2 = \{f \in (I \rightarrow [D^*]) \rightharpoonup (O \rightarrow [D^*]) \mid \forall \theta : \exists f_1 \in F_1, f_2 \in F_2 : f(\theta) = (\varphi + \psi)|_O \quad \text{where} \quad \varphi = f_1((\theta + \psi)|_{I_1}), \quad \psi = f_2((\theta + \varphi)|_{I_2})\}$$

$\square$

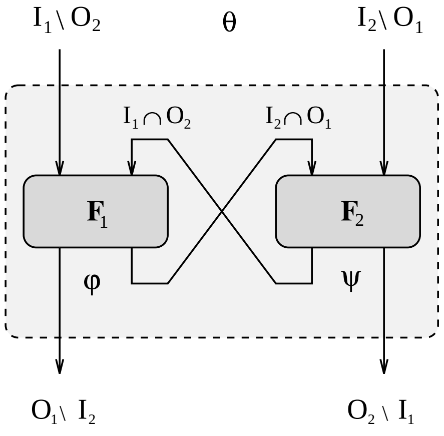

Figure 2.2: Point-to-Point communication

Note the close correlation between this definition and Figure 2.2. Any input channel of F_1 which is also an output channel of F_2, and any input channel of F_2 which is also an input channel of F_1 are connected and hidden. Any channel that cannot be connected in this way remains external. We refer to the resulting network as the *point-to-point* composition of F_1 and F_2.

Theorem 4 *$F_1 \otimes F_2 \neq \emptyset$.*

Proof: Since F_1 and F_2 are static components we may find functions f_1, f_2 such that $f_1 \in F_1$ and $f_2 \in F_2$. The function:

$$g((\varphi, \psi), \theta) = (f_1((\theta + \psi)|_{I_1}), f_2((\theta + \varphi)|_{I_2}))$$

where $\varphi \in (O_1 \to [D^*])$ and $\psi \in (O_2 \to [D^*])$ is strongly pulse-driven as the composition of strongly pulse-driven and weakly pulse-driven functions[1]. By Theorem 18 (in the appendix) so is μg. Moreover, by non-expansiveness of Cartesian projection, sum and stream tuple projection, f defined by:

$$f(\theta) = (\varphi + \psi)\,|_O \quad \text{where} \quad (\varphi, \psi) = (\mu g)(\theta)$$

is also strongly pulse-driven. Since for any predicate P:

$$\exists f_1 \in F_1, f_2 \in F_2 : \forall \theta : P \Rightarrow \forall \theta : \exists f_1 \in F_1, f_2 \in F_2 : P,$$

we obtain $F_1 \otimes F_2 \neq \emptyset$. $\square$

Theorem 5 *$F_1 \otimes F_2$ is a static component.*

Proof: By Theorem 4 it is enough to prove that $F_1 \otimes F_2$ is closed. Suppose $f \in (I \to [D^*]) \to (O \to [D^*])$ and

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

Then for a given θ there is an $f' \in F_1 \otimes F_2$ an $f_1 \in F_1$ and an $f_2 \in F_2$ such that:

$$f'(\theta) = (\varphi + \psi)|_O \quad \text{where} \quad \varphi = f_1((\theta + \psi)|_{I_1}), \quad \psi = f_2((\theta + \varphi)|_{I_2}).$$

Hence, for every θ there are functions $f_1 \in F_1$ and $f_2 \in F_2$ such that:

$$f(\theta) = (\varphi + \psi)|_O \quad \text{where} \quad \varphi = f_1((\theta + \psi)|_{I_1}), \quad \psi = f_2((\theta + \varphi)|_{I_2}).$$

In other words $f \in F_1 \otimes F_2$. $\square$

---

[1] Tupling and projection are non-expansive in the finitary product metric space. See for example [Eng77].

Chapter 3

Mobile Data

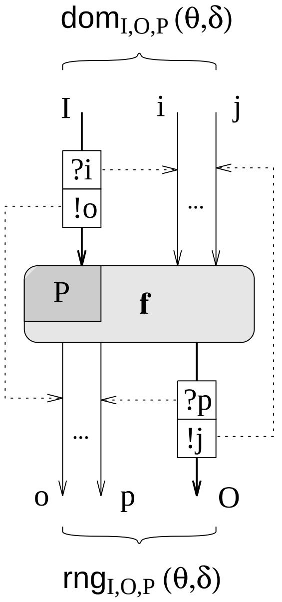

Figure 3.1: Mobile Stream Processing Function

Initially, each mobile function reads from a designated set of input channels I and writes on a designated set of output channels O. These two sets name the *static* channels or the initial wiring. To make sure that the channels *created* by the different components in a network are given 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 $(I \cup O) \cap P = \emptyset$.

At any point of time n the sets of accessible input and output channels are characterized by respectively do

3.2 Operations on Named Communication Histories

The new definition of named communication histories requires that the summation and projection operators are redefined. In the static case, the sum $\varphi+\psi$ was defined for φ and ψ with disjoint domains. In the mobile case we need a weaker condition, because all named communication histories have the same domain N. Let $\mathsf{act}(\varphi) = \{i \mid \varphi(i) \neq \epsilon^\infty\}$ be the set of active channels in φ, i.e. the set of channel names whose corresponding communication histories contain at least one message. The sum $\varphi+\psi$ is then defined if $\mathsf{act}(\varphi)$ is disjoint from $\mathsf{act}(\psi)$.

Definition 6 (Partial Sum) *Given two named communication histories* $\varphi \in (N \to [M^*])$ *and* $\psi \in (N \to [P^*])$ *such that* $\mathsf{act}(\varphi) \cap \mathsf{act}(\psi) = \emptyset$. *We define their* partial sum $\varphi+\psi$ *to denote the element of* $N \to [(M \cup P)^*]$ *such that for all* $i \in N$*:*

$$(\varphi+\psi)(i) = \begin{cases} \varphi(i) & \textit{if } i \in \mathsf{act}(\varphi), \\ \psi(i) & \textit{if } i \notin \mathsf{act}(\varphi). \end{cases}$$

□

Although analogous to the summation operator defined in Section 2.2, partial summation has no syntactic conditions assuring its well-definedness. We therefore define a total version $\varphi \nleftrightarrow \psi$. This simplifies the use of the Banach's fixed point theorem. Totalisation is achieved by defining $(\varphi \nleftrightarrow \psi)(i)$ to consist of only ϵ's from the first moment n in which both $\varphi{\downarrow_n}$ and $\psi{\downarrow_n}$ are active on i, i.e. different from ϵ^n.

Definition 7 (Total sum) *Given two named communication histories* $\varphi \in (N \to [M^*])$ *and* $\psi \in (N \to [P^*])$. *We define their* total sum $\varphi \nleftrightarrow \psi$ *to denote the element of* $N \to [(M \cup P)^*]$ *such that for all* $i \in N$*:*

$$(\varphi \nleftrightarrow \psi)(i)(n) = \begin{cases} \psi(i)(n) & \textit{if } \varphi(i){\downarrow_n} = \epsilon^n, \\ \varphi(i)(n) & \textit{if } \varphi(i){\downarrow_n} \neq \epsilon^n \wedge \psi(i){\downarrow_n} = \epsilon^n, \\ \epsilon & \textit{if } \varphi(i){\downarrow_n} \neq \epsilon^n \wedge \psi(i){\downarrow_n} \neq \epsilon^n. \end{cases}$$

□

Note that $\varphi \nleftrightarrow \psi$ has a hiding effect if $\mathsf{act}(\varphi) \cap \mathsf{act}(\psi) \neq \emptyset$, and that $\varphi \nleftrightarrow \psi$ is equal to $\varphi+\psi$, otherwise.

Theorem 6 *The total summation operator is weakly pulse-driven.*

Proof: Suppose that $\varphi_1, \varphi_2 \in N \to [M^*]$ and that $\psi_1, \psi_2 \in N \to [P^*]$ such that $\max\{d(\varphi_1,\varphi_2), d(\psi_1,\psi_2)\} = 2^{-n}$, i.e. such that $\varphi_1{\downarrow_n} = \varphi_2{\downarrow_n}$ and $\psi_1{\downarrow_n} = \psi_2{\downarrow_n}$. Since the definition of $(\varphi+\psi){\downarrow_n}$ depends only on $\varphi{\downarrow_n}$ and $\psi{\downarrow_n}$, it follows that $d(\varphi_1+\psi_1, \varphi_2+\psi_2) \leq 2^{-n}$. □

In the static case, the sets of input and output channels of a component were constant. This was also reflected by the projection operator, which did a constant projection over time. In the mobile case, however, the sets of known input and output channels do

Definition 8 (Projection) *For any named communication history* $\theta \in (N \rightarrow [M^*])$*, its* projection $\theta\,|_O$ *on* $O \in [\mathcal{P}(N)]$ *denotes the element of* $N \rightarrow [M^*]$ *such that for all* i, k*:*

$$\theta\,|_O\,(i)(k) = \begin{cases} \theta(i)(k) & \text{if } i \in O(k), \\ \epsilon & \text{otherwise.} \end{cases}$$

□

Theorem 7 *The projection operator is weakly pulse-driven.*

Proof: Suppose that $\theta_1, \theta_2 \in (N \rightarrow [M^*])$, that $O_1, O_2 \in [\mathcal{P}(N)]$ and that $\max\{d(\theta_1, \theta_2), d(O_1, O_2))\} = 2^{-n}$, i.e. that $\theta_1{\downarrow_n} = \theta_2{\downarrow_n}$ and $O_1{\downarrow_n} = O_2{\downarrow_n}$. Since $(\theta\,|_O){\downarrow_n}$ depends only on $\theta{\downarrow_n}$ and $O{\downarrow_n}$ it follows that $d(\theta_1\,|_{O_1}, \theta_2\,|_{O_2}) \leq 2^{-n}$. □

3.3 Mobile Components

Also in the case of mobile dataflow networks we model components by sets of stream processing functions. However, before we can give a formal characterization of these sets, we have to be more explicit with respect to what sort of networks we actually want to model.

Firstly, as in the static case, we want to model point-to-point communication. As a consequence, any network of components maintains the following invariant: each channel can be accessed by at most two components — the reader and the writer.

Secondly, the components can create channels and forward their ports. Each component has a set of private channel names P. Any channel created by a component is assigned a new name p from this set P. Any other component can only gain access to p by receiving either $?p$ or $!p$ along one of its input channels.

Thirdly, to ensure that the same channel is not accessed by more than two components, a port is sent by its creator along *only one channel.* Otherwise, it may happen that the same port is received and used by two different components. To simplify things, we assume the same port is never sent more than *once.* In addition, the creator is not allowed to send both ports of a channel. Thus for instance if it sends the write port of a channel then it is not allowed to send the read port for the same channel.

In a network where all components have disjoint sets of private names, and where all components behave in accordance with the communication constraints imposed above, we may restrict ourselves to named communication histories in which the same port occurs not more than once and where two different ports are assigned different channel names. We say that a named communication history is *port-unique* if it has these two properties. Moreover, we use the arrow $\overset{\mathsf{u}}{\rightarrow}$ to distinguish named communication histories that are port-unique from other named communication histories.

Port uniqueness is preserved by projection, and also by summation as long as the named communication histories have disjoint sets of ports.

Theorem 8 *If* $\theta \in N \overset{\mathsf{u}}{\rightarrow} [P^*]$ *we also have that* $\theta\,|_O \in N \overset{\mathsf{u}}{\rightarrow} [P^*]$*. Moreover, if* $\varphi \in N \overset{\mathsf{u}}{\rightarrow} [P_1^*], \psi \in N \overset{\mathsf{u}}{\rightarrow} [P_2^*]$ *and* $P_1 \cap P_2 = \emptyset$*, we also have that* $\varphi \nleftrightarrow \psi \in N \overset{\mathsf{u}}{\rightarrow} [(P_1 \cup P_2)^*]$*.*

Proof: Trivial. □

Note that a received port can be forwarded without violating privacy as long as it is not read or written by the component forwarding it. Nevertheless, to keep it simple, we do not allow received ports to be forwarded. This constraint does not reduce the number of networks that can be expressed, because whenever the forwarding component receives a port $?i$, it can always forward a copy $?p$ by generating a new channel p and then transmitting anything received on i along p. We also restrict components from communicating the ports used for their initial wiring.

It follows from these constraints that any port sent by a component has to belong to a channel created by the component.

For simplicity, we split the set of names N into two disjoint sets – a set of static channel names S, which can only be used for the initial wiring, and a set of dynamic channel names A, which can only be assigned channels created during the computation.

Based on the discussion above, when modeling a component with private names P, we only have to consider stream processing functions of the following signature:

$$(N \xrightarrow{\mathsf{u}} [\overline{P}^*]) \rightarrow (N \xrightarrow{\mathsf{u}} [P^*]),$$

where $P \subseteq A$ and $\overline{P} = A \setminus P$.

So far we have fixed the syntactic signature of our stream processing functions. The next step is to characterize their semantic properties. As in the static case, any stream processing function is required to be strongly pulse-driven. However, contrary to earlier, it is also required to have a second property — as already mentioned, it is required to be *generic* in the sense that it only accesses a channel via a port it has already received or created itself.

Before giving a formal definition of what it means for a stream processing function to be generic, we have to define the two sets do

Theorem 10 *The functions* do

nature

$$(\forall \theta \in (N \to [\overline{P}^*]) : \exists f' \in F : f(\theta) = f'(\theta)) \Rightarrow f \in F.$$

□

3.4 Point-to-Point Composition

We now redefine the composition operator $\otimes$ for mobile components.

Definition 12 (Point-to-point composition) *Given two mobile components:*

$$F_1 \subseteq (N \overset{\mathsf{u}}{\to} [\overline{P_1}^*]) \overset{I_1,O_1}{\to} (N \overset{\mathsf{u}}{\to} [P_1^*]), \qquad F_2 \subseteq (N \overset{\mathsf{u}}{\to} [\overline{P_2}^*]) \overset{I_2,O_2}{\to} (N \overset{\mathsf{u}}{\to} [P_2^*]).$$

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

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

we define

$$F_1 \otimes F_2 \subseteq (N \overset{\mathsf{u}}{\to} [\overline{P}^*]) \overset{I,O}{\to} (N \overset{\mathsf{u}}{\to} [P^*])$$

$$\begin{aligned} F_1 \otimes F_2 = \{ f \in (N \overset{\mathsf{u}}{\to} [\overline{P}^*]) \overset{I,O}{\to} (N \overset{\mathsf{u}}{\to} [P^*]) \ \mid \ \forall \theta : \exists f_1 \in F_1, f_2 \in F_2 : \\ f(\theta) = (\varphi \nleftrightarrow \psi)|_{\mathsf{rng}_{I,O}(\theta, \varphi \nleftrightarrow \psi)} \quad \text{where} \\ \varphi = f_1(\delta \nleftrightarrow \psi), \quad \psi = f_2(\delta \nleftrightarrow \varphi), \quad \delta = \theta|_{\mathsf{dom}_{I,O}(\theta, \varphi \nleftrightarrow \psi)} \} \end{aligned}$$

□

Theorem 11 $F_1 \otimes F_2 \neq \emptyset$.

Proof: Since F_1 and F_2 are mobile components we may find functions f_1, f_2 such that $f_1 \in F_1$ and $f_2 \in F_2$. Based on these functions we construct a function f which is strongly pulse-driven, generic and satisfies the recursive definition above.

Let

$$g \in ((N \overset{\mathsf{u}}{\to} [P_1^*]) \times (N \overset{\mathsf{u}}{\to} [P_2^*])) \times (N \overset{\mathsf{u}}{\to} [\overline{P}^*]) \to (N \overset{\mathsf{u}}{\to} [P_1^*]) \times (N \overset{\mathsf{u}}{\to} [P_2^*])$$

be defined as follows:

$$g((\varphi, \psi), \theta) = (f_1(\delta \nleftrightarrow \psi),\ f_2(\delta \nleftrightarrow \varphi)) \quad \text{where} \quad \delta = \theta|_{\mathsf{dom}_{I,O}(\theta, \varphi \nleftrightarrow \psi)}\ .$$

Lemma 1 *g is well defined.*

Proof: Since $\theta \in (N \overset{\mathsf{u}}{\to} [\overline{P}^*])$ it follows by Theorem 8 that $\delta \in (N \overset{\mathsf{u}}{\to} [\overline{P}^*])$. Since $\psi \in (N \overset{\mathsf{u}}{\to} [P_2{}^*])$ and $P_2 \cap \overline{P} = \emptyset$, and since $\varphi \in (N \overset{\mathsf{u}}{\to} [P_1{}^*])$ and $P_1 \cap \overline{P} = \emptyset$, it follows by Theorem 8 that $\delta \nleftrightarrow \psi \in (N \overset{\mathsf{u}}{\to} [(P_2 \cup \overline{P})^*])$ and that $\delta \nleftrightarrow \varphi \in (N \overset{\mathsf{u}}{\to} [(P_1 \cup \overline{P})^*])$. Since $P_2 \cup \overline{P} \subseteq \overline{P}_1$ and $P_1 \cup \overline{P} \subseteq \overline{P}_2$, we have that

$$\delta \nleftrightarrow \psi \in (N \overset{\mathsf{u}}{\to} [\overline{P_1}^*]), \qquad \delta \nleftrightarrow \varphi \in (N \overset{\mathsf{u}}{\to} [\overline{P_2}^*]).$$

As a consequence $f_1(\delta \nleftrightarrow \psi) \in (N \overset{\mathsf{u}}{\to} [P_1^*])$ and $f_2(\delta \nleftrightarrow \varphi) \in (N \overset{\mathsf{u}}{\to} [P_2^*])$. □

Theorem 16 (in the appendix) and the way g is defined in terms of strongly and weakly pulse-driven functions imply that g is strongly pulse-driven. Thus $\mu\, g$ is well-defined, in which case Theorem 18 (in the appendix) implies that $\mu\, g$ is strongly pulse-driven. Let

$$f \in (N \overset{\mathsf{u}}{\to} [\overline{P^*}]) \to (N \overset{\mathsf{u}}{\to} [P^*])$$

be defined as follows:

$$f(\theta) = (\varphi \nleftrightarrow \psi)|_{\mathsf{rng}_{I,O}(\theta,\varphi \nleftrightarrow \psi)} \quad \textsf{where} \quad (\varphi,\psi) = (\mu\, g)(\theta).$$

Lemma 2 *f is well defined.*

Proof: It is enough to show that $\varphi \nleftrightarrow \psi \in N \overset{\mathsf{u}}{\to} [P^*]$. This follows straightforwardly by the signature of g and Theorem 8 since $P_1 \cap P_2 = \emptyset$. □

Theorem 16 and the way f is defined in terms of strongly and weakly pulse-driven functions imply that f is strongly pulse-driven.

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

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

Proof: First, note that $f(\theta) = f(\theta|_{\mathsf{dom}_{I,O}(\theta,\varphi \nleftrightarrow \psi)})$ because

$$\begin{array}{ll}
\mathsf{rng}_{I,O}(\theta,\varphi \nleftrightarrow \psi) = \mathsf{rng}_{I,O}(\theta|_{\mathsf{dom}_{I,O}(\theta,\varphi \nleftrightarrow \psi)},\varphi \nleftrightarrow \psi) & \{\text{by Theorem 10}\} \\
\delta = \theta|_{\mathsf{dom}_{I,O}(\theta,\varphi \nleftrightarrow \psi)} = \theta|_{\mathsf{dom}_{I,O}(\theta,\varphi \nleftrightarrow \psi)}|_{\mathsf{dom}_{I,O}(\theta,\varphi \nleftrightarrow \psi)} & \{\text{by idempotence of projection.}\}
\end{array}$$

Now

$$\begin{array}{ll}
f(\theta|_{\mathsf{dom}_{I,O}(\theta,f(\theta))}) = & \\
f(\theta|_{\mathsf{dom}_{I,O}(\theta,(\varphi \nleftrightarrow \psi)|_{\mathsf{rng}_{I,O}(\theta,\varphi \nleftrightarrow \psi)})}) = & \{\text{by definition of } f\} \\
f(\theta|_{\mathsf{dom}_{I,O}(\theta,\varphi \nleftrightarrow \psi)}) = & \{\text{by Theorem 10}\} \\
f(\theta) & \{\text{by above remark}\}
\end{array}$$

□

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

Proof:

$$\begin{array}{ll}
f(\theta)|_{\mathsf{rng}_{I,O}(\theta,f(\theta))} = & \\
(\varphi \nleftrightarrow \psi)|_{\mathsf{rng}_{I,O}(\theta,\varphi \nleftrightarrow \psi)}|_{\mathsf{rng}_{I,O}(\theta,(\varphi \nleftrightarrow \psi)|_{\mathsf{rng}_{I,O}(\theta,\varphi \nleftrightarrow \psi)})} = & \{\text{by definition of } f\} \\
(\varphi \nleftrightarrow \psi)|_{\mathsf{rng}_{I,O}(\theta,\varphi \nleftrightarrow \psi)}|_{\mathsf{rng}_{I,O}(\theta,\varphi \nleftrightarrow \psi)} = & \{\text{by Theorem 10}\} \\
(\varphi \nleftrightarrow \psi)|_{\mathsf{rng}_{I,O}(\theta,\varphi \nleftrightarrow \psi)} = & \{\text{by idempotence of projection}\} \\
f(\theta) & \{\text{by definition of } f\}
\end{array}$$

□

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

Theorem 12 *$F_1 \otimes F_2$ is a mobile component.*

Proof: That $F_1 \otimes F_2 \neq \emptyset$ follows from Theorem 11. To see that $F_1 \otimes F_2$ is closed, let $f \in (N \overset{\mathsf{u}}{\rightarrow} [\overline{P}^*]) \overset{I,O}{\rightarrow} (N \rightarrow [P^*])$, and assume that

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

The definition of $\otimes$ implies that for any θ there are $f_1 \in F_1$, $f_2 \in F_2$ such that:

$$f(\theta) = (\varphi \nleftrightarrow \psi)|_{\mathsf{rng}_{I,O}(\theta, \varphi \nleftrightarrow \psi)} \quad \text{where}$$
$$\varphi = f_1(\delta \nleftrightarrow \psi), \quad \psi = f_2(\delta \nleftrightarrow \varphi), \quad \delta = \theta|_{\mathsf{dom}_{I,O}(\theta, \varphi \nleftrightarrow \psi)}$$

By the definition of $\otimes$, it follows that $f \in F_1 \otimes F_2$. $\square$

The definition of $\otimes$ depends on the total operator for summation. This operator is a bit strange since it results in hiding if its arguments are active on the same channels. Our composition operator $\otimes$, on the other hand, should only hide the feedback channels. This means that all messages sent or received by the components along the external channels should be visible also after the composition. As a consequence, in the definition of $\otimes$ it should be possible to replace the total operator for summation by the partial one.

Theorem 13 *The operator $\nleftrightarrow$ can be replaced by $+$ in the definition of $\otimes$.*

Proof: Given that

$$\varphi = f_1(\delta \nleftrightarrow \psi), \quad \psi = f_2(\delta \nleftrightarrow \varphi), \quad \delta = \theta|_{\mathsf{dom}_{I,O}(\theta, \varphi \nleftrightarrow \psi)} \,.$$

It is enough to show that

$$\mathsf{act}(\varphi) \cap \mathsf{act}(\psi) = \mathsf{act}(\varphi) \cap \mathsf{act}(\delta) = \mathsf{act}(\psi) \cap \mathsf{act}(\delta) = \emptyset.$$

Since the genericity of f_1 and f_2 imply

$$\varphi = \varphi|_{\mathsf{rng}_{I_1,O_1}(\delta \nleftrightarrow \psi, \varphi)}, \qquad \psi = \psi|_{\mathsf{rng}_{I_2,O_2}(\delta \nleftrightarrow \varphi, \psi)},$$

this is equivalent to showing that for all $n \in \mathsf{N}$, the sets

$$\mathsf{rng}_{I_1,O_1}(\delta \nleftrightarrow \psi, \varphi)(n), \; \mathsf{rng}_{I_2,O_2}(\delta \nleftrightarrow \varphi, \psi)(n), \; \text{do}$$

$$\begin{array}{lclcl}
R_1^{n+1} = R_1^n & \cup & \bigcup_{i\in D_1^n}\{p \mid !p \in (\delta\nleftrightarrow\psi)(i)(n)\} & \cup & \bigcup_{i\in R_1^n}\{p \mid ?p \in \varphi(i)(n)\} \\
D_1^{n+1} = D_1^n & \cup & \bigcup_{i\in D_1^n}\{p \mid ?p \in (\delta\nleftrightarrow\psi)(i)(n)\} & \cup & \bigcup_{i\in R_1^n}\{p \mid !p \in \varphi(i)(n)\} \\
R_2^{n+1} = R_2^n & \cup & \bigcup_{i\in D_2^n}\{p \mid !p \in (\delta\nleftrightarrow\varphi)(i)(n)\} & \cup & \bigcup_{i\in R_2^n}\{p \mid ?p \in \psi(i)(n)\} \\
D_2^{n+1} = D_2^n & \cup & \bigcup_{i\in D_2^n}\{p \mid ?p \in (\delta\nleftrightarrow\varphi)(i)(n)\} & \cup & \bigcup_{i\in R_2^n}\{p \mid !p \in \psi(i)(n)\} \\
R^{n+1} = R^n & \cup & \bigcup_{i\in D^n}\{p \mid !p \in \theta(i)(n)\} & \cup & \bigcup_{i\in R^n}\{p \mid ?p \in (\varphi\nleftrightarrow\psi)(i)(n)\} \\
D^{n+1} = D^n & \cup & \bigcup_{i\in D^n}\{p \mid ?p \in \theta(i)(n)\} & \cup & \bigcup_{i\in R^n}\{p \mid !p \in (\varphi\nleftrightarrow\psi)(i)(n)\}.
\end{array}$$

We show that $R_1^{n+1} \cap R_2^{n+1} = \emptyset$. The justification of a proof step is given in curly brackets: $pU(\varphi,\psi)$ means "by port uniqueness" of φ and ψ, $dJ(\varphi,\psi,\delta)$ means "by mutually disjointness" of φ,ψ and δ, and iH means "by induction hypothesis". If two cases are symmetric only one is presented.

$$R_1^n \cap R_2^n = \emptyset \qquad \{\text{iH}\}$$
$$R_1^n \cap \bigcup_{i\in D_2^n}\{p \mid !p \in (\delta\nleftrightarrow\varphi)(i)(n)\} = \emptyset \qquad \{pU(\varphi,\delta), dJ(P_1,P_2,\overline{P})\}$$
$$R_1^n \cap \bigcup_{i\in R_2^n}\{p \mid ?p \in \psi(i)(n)\} = \emptyset \qquad \{pU(\psi), dJ(P_1,P_2,\overline{P})\}$$

$$\bigcup_{i\in D_1^n}\{p \mid !p \in (\delta\nleftrightarrow\psi)(i)(n)\} \cap \bigcup_{i\in D_2^n}\{p \mid !p \in (\delta\nleftrightarrow\varphi)(i)(n)\} = \emptyset$$
$$\{dJ(P_1,P_2,\overline{P}),\ pU(\delta),\ \text{iH}\}$$
$$\bigcup_{i\in D_1^n}\{p \mid !p \in (\delta\nleftrightarrow\psi)(i)(n)\} \cap \bigcup_{i\in R_2^n}\{p \mid ?p \in \psi(i)(n)\} = \emptyset \qquad \{pU(\psi), dJ(P_2,\overline{P})\}$$

$$\bigcup_{i\in R_1^n}\{p \mid ?p \in (\varphi)(i)(n)\} \cap \bigcup_{i\in R_2^n}\{p \mid ?p \in \psi(i)(n)\} = \emptyset \qquad \{dJ(P_1,P_2)\}$$

We show that $R_1^{n+1} \cap D^{n+1} = \emptyset$. The proof that $R_2^{n+1} \cap D^{n+1} = \emptyset$ is symmetric and therefore left out.

$$R_1^n \cap D^n = \emptyset \qquad \{\text{iH}\}$$
$$R_1^n \cap \bigcup_{i\in D^n}\{p \mid ?p \in \theta(i)(n)\} = \emptyset \qquad \{pU(\theta), dJ(P_1,P_2,\overline{P})\}$$
$$R_1^n \cap \bigcup_{i\in R^n}\{p \mid !p \in (\varphi\nleftrightarrow\psi)(i)(n)\} = \emptyset \qquad \{pU(\varphi,\psi), dJ(P_1,P_2,\overline{P})\}$$

$$\bigcup_{i\in D_1^n}\{p \mid !p \in (\delta\nleftrightarrow\psi)(i)(n)\} \cap D^n = \emptyset \qquad \{pU(\psi,\theta), dJ(P_1,P_2,\overline{P})\}$$
$$\bigcup_{i\in D_1^n}\{p \mid !p \in (\delta\nleftrightarrow\psi)(i)(n)\} \cap \bigcup_{i\in D^n}\{p \mid ?p \in \theta(i)(n)\} = \emptyset \qquad \{pU(\theta), dJ(P_2,\overline{P})\}$$
$$\bigcup_{i\in D_1^n}\{p \mid !p \in (\delta\nleftrightarrow\psi)(i)(n)\} \cap \bigcup_{i\in R^n}\{p \mid !p \in (\varphi\nleftrightarrow\psi)(i)(n)\} = \emptyset$$
$$\{dJ(P_1,P_2,\overline{P}),\ pU(\psi),\ \text{iH}\}$$

$$\bigcup_{i\in R_1^n}\{p \mid ?p \in (\varphi)(i)(n)\} \cap D^n = \emptyset \qquad \{pU(\varphi), dJ(P_1,P_2,\overline{P})\}$$
$$\bigcup_{i\in R_1^n}\{p \mid ?p \in (\varphi)(i)(n)\} \cap \bigcup_{i\in D^n}\{p \mid ?p \in \theta(i)(n)\} = \emptyset \qquad \{dJ(P_1,\overline{P}\}$$
$$\bigcup_{i\in R_1^n}\{p \mid ?p \in (\varphi)(i)(n)\} \cap \bigcup_{i\in R^n}\{p \mid !p \in (\varphi\nleftrightarrow\psi)(i)(n)\} = \emptyset \qquad \{pU(\varphi), dJ(P_1,P_2)\}$$

We show that $D_1^{n+1} \cap D_2^{n+1} = \emptyset$. If two cases are symmetric only one is presented.

$$D_1^n \cap D_2^n = \emptyset \qquad \{\text{iH}\}$$
$$D_1^n \cap \bigcup_{i\in D_2^n}\{p \mid ?p \in (\delta\nleftrightarrow\varphi)(i)(n)\} = \emptyset \qquad \{pU(\varphi,\delta), dJ(P_1,P_2,\overline{P})\}$$
$$D_1^n \cap \bigcup_{i\in R_2^n}\{p \mid !p \in \psi(i)(n)\} = \emptyset \qquad \{pU(\psi), dJ(P_1,P_2,\overline{P})\}$$

$$\bigcup_{i\in D_1^n}\{p \mid ?p \in (\delta\nleftrightarrow\psi)(i)(n)\} \cap \bigcup_{i\in D_2^n}\{p \mid ?p \in (\delta\nleftrightarrow\varphi)(i)(n)\} = \emptyset$$
$$\{dJ(P_1,P_2,\overline{P}),\ pU(\delta),\ \text{iH}\}$$
$$\bigcup_{i\in D_1^n}\{p \mid ?p \in (\delta\nleftrightarrow\psi)(i)(n)\} \cap \bigcup_{i\in R_2^n}\{p \mid !p \in \psi(i)(n)\} = \emptyset \qquad \{pU(\psi), dJ(P_2,\overline{P})\}$$

$$\bigcup_{i\in R_1^n}\{p \mid !p \in (\varphi)(i)(n)\} \cap \bigcup_{i\in R_2^n}\{p \mid !p \in \psi(i)(n)\} = \emptyset \qquad \{dJ(P_1,P_2)\}$$

We show that $D_1^{n+1} \cap R^{n+1} = \emptyset$. The proof that $D_2^{n+1} \cap R^{n+1} = \emptyset$ is symmetric and therefore left out.

$$D_1^n \cap R^n = \emptyset \qquad \{\text{iH}\}$$
$$D_1^n \cap \bigcup_{i \in D^n} \{p \mid !p \in \theta(i)(n)\} = \emptyset \qquad \{pU(\theta), dJ(P_1, P_2, \overline{P})\}$$
$$D_1^n \cap \bigcup_{i \in R^n} \{p \mid ?p \in (\varphi \nleftrightarrow \psi)(i)(n)\} = \emptyset \qquad \{pU(\varphi, \psi), dJ(P_1, P_2, \overline{P})\}$$

$$\bigcup_{i \in D_1^n} \{p \mid ?p \in (\delta \nleftrightarrow \psi)(i)(n)\} \cap R^n = \emptyset \qquad \{pU(\psi, \theta), dJ(P_1, P_2, \overline{P})\}$$
$$\bigcup_{i \in D_1^n} \{p \mid ?p \in (\delta \nleftrightarrow \psi)(i)(n)\} \cap \bigcup_{i \in D^n} \{p \mid !p \in \theta(i)(n)\} = \emptyset \qquad \{pU(\theta), dJ(P_2, \overline{P})\}$$
$$\bigcup_{i \in D_1^n} \{p \mid ?p \in (\delta \nleftrightarrow \psi)(i)(n)\} \cap \bigcup_{i \in R^n} \{p \mid ?p \in (\varphi \nleftrightarrow \psi)(i)(n)\} = \emptyset$$
$$\{dJ(P_1, P_2, \overline{P}),\ pU(\psi),\ \text{iH}\}$$

$$\bigcup_{i \in R_1^n} \{p \mid !p \in (\varphi)(i)(n)\} \cap R^n = \emptyset \qquad \{pU(\varphi), dJ(P_1, P_2, \overline{P})\}$$
$$\bigcup_{i \in R_1^n} \{p \mid !p \in (\varphi)(i)(n)\} \cap \bigcup_{i \in D^n} \{p \mid !p \in \theta(i)(n)\} = \emptyset \qquad \{dJ(P_1, \overline{P}\}$$
$$\bigcup_{i \in R_1^n} \{p \mid !p \in (\varphi)(i)(n)\} \cap \bigcup_{i \in R^n} \{p \mid ?p \in (\varphi \nleftrightarrow \psi)(i)(n)\} = \emptyset \qquad \{pU(\varphi), dJ(P_1, P_2)\}\}$$

$\square$

Chapter 4

Communication Central

As an example we specify a communication central (see Figure 4.1). Its task is to build up connections between station$_1$ and station$_2$. The initial "wires" are a_1 and a_2. Station$_1$ can send ports to be connected (both read and write) along a_1, and station$_2$ can send ports to be connected (both read and write) along a_2.

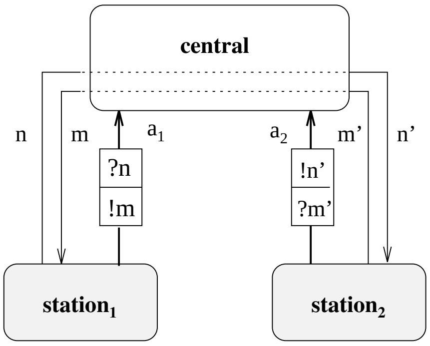

Figure 4.1: Communication central

Let $?n$ be the j'th read port sent along a_1 by station$_1$. The central is allowed to read from n as soon as this port is received. Moreover, let $!n'$ be the j'th write port sent along a_2 by station$_2$. The central is allowed to write on n' as soon as this port is received. The central "connects" these two channels by forwarding each message it receives on the channel n along the channel n'. Symmetrically, if $?m'$ is the k'th read port sent along a_2 by station$_2$ and $!m$ is the k'th write port sent along a_1 by station$_1$, the messages received on m' are forwarded along m.

In order to specify this component, let us introduce three basic operators. The first one is the *filter* operator: for any set of messages M and stream of messages s, $M \copyright s$ denotes the stream we get by removing any message in s not contained in M. The second one is the *length* operator: for any stream s, $\#s$ yields its length. This means that $\#s = \infty$ if s is infinite. Finally, we need a *time abstraction operator*: for any named communication history β, $\overline{\beta}$ denotes the result of removing all time information in β. For any i, this is achieved by concatenating all the finite streams in $\beta(i)$ into one stream. Thus each communication history consisting of infinitely many finite streams of messages is replaced by the result of concatenating its finite streams into one stream of messages. As a result the timing information is abstracted away.

$$\text{Central} \in ((N \overset{\mathsf{u}}{\rightarrow} [\overline{P}]) \overset{\{a_1,a_2\},\emptyset}{\rightarrow} (N \overset{\mathsf{u}}{\rightarrow} [P])) \rightarrow \mathsf{Bool}$$

$$\begin{array}{ll}
\multicolumn{2}{l}{\text{Central}(f) \overset{\mathsf{def}}{=} \forall \alpha : f(\alpha) = \beta \quad \mathsf{where}} \\
\quad \forall (n, n') \in acon : \overline{\beta}(n') = \overline{\alpha}(n) & -- \text{ the forwarding mechanism} \\
\quad acon = con(a_1, a_2) \cup con(a_2, a_1) & -- \text{ the set of all connections} \\
\quad con(a, b) = \{(n, n') \mid & -- \text{ the set of connections from ``}a\text{ to }b\text{''} \\
\qquad \exists k \in [1.. &
\end{array}$$

Chapter 5

Full Abstractness

Syntactically, a dataflow network is either a basic component or the point-to-point composition of two networks. The context free syntax is given by the following productions:

$$net ::= x \mid net \otimes net$$

Network terms defined in this way are also called *rough terms*, because they do not contain any information about their syntactic interface i.e. information about their input and output channels. By adding interface (or type) information we define the *context sensitive syntax* for networks as a typing system. As axioms we have typing declarations of the form:

$$x :: (N \xrightarrow{\mathsf{u}} [\overline{P}^*]) \xrightarrow{I,O} (N \xrightarrow{\mathsf{u}} [P^*]).$$

We have only one typing rule:

$$\frac{\begin{array}{c} x_1 :: (N \xrightarrow{\mathsf{u}} [\overline{P_1}^*]) \xrightarrow{I_1,O_1} (N \xrightarrow{\mathsf{u}} [P_1^*]) \\ x_2 :: (N \xrightarrow{\mathsf{u}} [\overline{P_2}^*]) \xrightarrow{I_2,O_2} (N \xrightarrow{\mathsf{u}} [P_2^*]) \end{array}}{x_1 \otimes x_2 :: (N \xrightarrow{\mathsf{u}} [\overline{P}^*]) \xrightarrow{I,O} (N \xrightarrow{\mathsf{u}} [P^*])} \; \{ \; I_1 \cap I_2 = O_1 \cap O_2 = P_1 \cap P_2 = \emptyset$$

where $I = (I_1 \setminus O_2) \cup (I_2 \setminus O_1)$, $O = (O_1 \setminus I_2) \cup (O_2 \setminus I_1)$ and $P = P_1 \cup P_2$. It states that we can compose two networks if their input and output channels are disjoint. The composition operator hides the channels connecting the output of one network with the input of the other one.

A context $C(.) :: (N \xrightarrow{\mathsf{u}} [\overline{P}^*]) \xrightarrow{I,O} (N \xrightarrow{\mathsf{u}} [P^*])$ is a network with exactly one hole of type $(N \xrightarrow{\mathsf{u}} [\overline{P}^*]) \xrightarrow{I,O} (N \xrightarrow{\mathsf{u}} [P^*])$. A network of this type is allowed to be substituted for the hole.

The denotational semantics of a network expression is defined as follows

$$\begin{array}{l} \mathcal{D}[n] \subseteq (N \xrightarrow{\mathsf{u}} [\overline{P}^*]) \xrightarrow{I,O} (N \xrightarrow{\mathsf{u}} [P^*]) \quad \text{if} \quad n :: (N \xrightarrow{\mathsf{u}} [\overline{P}^*]) \xrightarrow{I,O} (N \xrightarrow{\mathsf{u}} [P^*]) \\ \mathcal{D}[n \otimes m] = \mathcal{D}[n] \otimes \mathcal{D}[m] \end{array}$$

We define the observation semantics of a network as the input output behavior of its semantic denotation as follows:

$$\mathcal{O}[n] \quad = \quad \{(x, y) \mid \exists f \in \mathcal{D}[n] : \; y = f(x)\}$$

Now, we define full abstraction as in [Jon89].

Definition 13 Full abstraction *The model* $\mathcal{D}$ *is said to be fully abstract with respect to* $\mathcal{O}$ *if for all networks* $n, m :: (N \xrightarrow{\mathtt{u}} [\overline{P}^*]) \xrightarrow{I,O} (N \xrightarrow{\mathtt{u}} [P^*])$

$$\begin{array}{llcll}
(1) & \mathcal{D}[n] = \mathcal{D}[m] & \Rightarrow & \mathcal{O}[n] = \mathcal{O}[n] & \{\mathcal{D} \textit{ is more distinguishing}\} \\
(2) & \forall C(.) : \mathcal{D}[n] = \mathcal{D}[m] & \Rightarrow & \mathcal{D}[C(n)] = \mathcal{D}[C(m)] & \{\mathcal{D} \textit{ is compositional}\} \\
(3) & \mathcal{D}[n] \neq \mathcal{D}[m] & \Rightarrow & \exists C(.) : \mathcal{O}[C(n)] \neq \mathcal{O}[C(m)] &
\end{array}$$

□

Theorem 14 *Our denotational semantics* $\mathcal{D}$ *is fully abstract with respect to* $\mathcal{O}$.

Proof: Property (1) follows from the definition of $\mathcal{O}$. Property (2) follows straightforwardly from the definition of D. To prove (3) suppose that $\mathcal{D}[n] \neq \mathcal{D}[m]$. This means, there is an f which is contained in the first set but not in the second. Since $\mathcal{D}[m]$ is closed there is a θ such that $\forall g \in \mathcal{D}[m] :\ f(\theta) \neq g(\theta)$. Otherwise, closeness would require that f is also in $\mathcal{D}[m]$. As a consequence $(\theta, f(\theta)) \notin \mathcal{O}[m]$. Hence $\mathcal{O}[n] \neq \mathcal{O}[m]$ □

Note that the full abstractness property is lost if the components are not required to be closed.

Chapter 6

Discussion

We have had several sources of inspiration. First of all, the semantic model for static networks is inspired by [Par83], [Kok87], [Bro87]. Park models components by sets of functions in the same way as we do. However, he models time with time ticks $\surd$ and his functions are defined also for finite streams. Moreover, infinite streams are not required to have infinitely many ticks. Kok models nondeterministic components by functions mapping communication histories to sets of communication histories. We use instead a closed set of strongly pulse-driven functions. This allows us to model unbounded nondeterminism without having to introduce a more complex metric. [Bro87] employs sets of functions as we do, but these functions work on untimed finite and infinite streams. This makes the model more abstract but at the same time more complex with respect to its theoretical basis. The formulation of pulse-drivenness has been taken from [Bro95a], and the use of named communication histories is based on [BD92]. The use of closed sets of functions to model dataflow components is not new — see for example [Rus90].

Our ideas on mobility have also had several sources. [Bro95b] and [Gro94] give an equational characterization of dynamic reconfiguration. [Gro94] also presents a semantic model for mobile, deterministic networks. However, that model is higher-order and mobility is achieved by communicating channels and functions instead of ports.

The idea of communicating names (ports) was inspired by [MPW92a, MPW92b]. The action structures [Mil92] are also related to our model. The idea of associating an access right with each channel in order to control interference was taken from [SKL90]. Our semantic conditions which assure point-to-point communication were also inspired by the syntactic conditions given in that paper. The semantic model of [SRP91] has also many similarities to our model. However, the intentions are complementary. [SRP91] aims at generality. We, on the other hand, have developed a particular model based on traditional stream processing functions. In [SRP91] inconsistencies can only be avoided by syntactic constraints. In our model, inconsistencies can easily be avoided at the semantic level without additional syntactic constraints.

There are many directions for future research. We believe that this work represents a first step towards a compositional development method for mobile systems in the tradition of Focus [BDD$^+$93]. We also intend to extend our model to handle many-to-many composition, recursive (infinite) networks and parameterized components. Since these issues are well understood in the case of the static networks, we do not expect these extensions to be very difficult.

Chapter 7

Acknowledgments

We thank Manfred Broy for stimulating discussions and valuable feedback. Thanks go also to Thomas Streicher who pointed out the close connection between pulse-driven and contractive functions. Christian Prehofer read an early version of this paper and made helpful comments. The first author has been financially supported by the DFG project Spectrum. The second author has been financially supported by the Sonderforschungsbereich 342 “Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen”.

Bibliography

[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]

[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.

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

[Mil92] R. Milner. Action structures. Technical Report ECS-LFCS-92-249, University of Edinburgh, 1992.

[MPS86] D. MacQueen,

Appendix A

Metric Space Basics

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

Definition 14 (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 15 (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 16 (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 15 *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 17 (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 16 *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 17 (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] or [Sut75]. □

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

Definition 18 (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 18 *If* f *is contractive (non-expansive) so is* μf.
Proof: See for example [MPS86] pages 114–115. □

Appendix B

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 19 (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 19 *The metric space of streams* $([E], d)$ *is complete.*
Proof: See for example [Eng77]. □

A *named stream tuple* is a mapping $\theta \in (I \rightarrow [E])$ 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 20 (The metric of named stream tuples) *The metric of* named stream tuples $(I \rightarrow [E], d)$ *with names in* I *and elements in* (E, ρ) *is defined as follows:*

$I \rightarrow [E]$ *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 20 *The metric space of named stream tuples* $(I \rightarrow [E], d)$ *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

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