A Model for Mobile Point-to-Point Data-flow Networks without Channel Sharing

Radu Grosu, Ketil Stølen*

Institut für Informatik, TU München, Postfach 20 24 20, D-80290 München
email:grosu,stoelen@informatik.tu-muenchen.de

Abstract. We present a fully abstract, denotational model for mobile, timed, nondeterministic data-flow networks whose components communicate in a point-to-point 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, generic and point-to-point. A stream processing function is strongly guarded if it is contractive with respect to the metric on streams. This property guarantees the existence of unique fix-points. Genericity is a privacy requirement specific to mobile systems. It guarantees that a function never accesses, depends on or sends a port whose name it does not already know. The point-to-point property guarantees that no port is known to more than two components: the sender and the receiver. Our model allows the description of a wide variety of networks — in particular, the description of mobile, unbounded nondeterministic networks. We demonstrate some features of our model by specifying a communication central.

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 communicating 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 two directions.

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.

* Address of Ketil Stølen from September 1, 1996: Institute for Energy Technology, P.O.Box 173, 1751 Halden, Norway. Email:Ketil.Stoelen@hrp.no

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 which includes all the fair merge components described 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] and [Rus90], we describe *dynamically reconfigurable* or *mobile* networks. 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], the $\pi-$Calculus [EN86, MPW92], 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 network components. Secondly, the components preserve privacy: their behavior cannot 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 never receives on, sends along or sends a port whose name "it does not already know". By "the ports it does not already know" we basically mean any port which is not in its initial interface, it has not already received, and it has not 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.

Our semantic framework is powerful enough to allow the modeling of both *point-to-point* and *many-to-many* communication. In [GS96a] we model many-to-many communication. In this paper we concentrate on point-to-point communication. By point-to-point communication we mean that no port is known to more than two components: the sender and the receiver. Some readers may wonder why we at all find point-to-point communication interesting. After all, point-to-point communication is only a special case of many-to-many communication. The main reason is that point-to-point communication allows a tight control of channel interference. In a point-to-point model the default situation is no interference at all or a very restricted form of interference. Unrestricted interference is only simulated by introducing explicit fair merge components for those channels where this is desirable. In a many-to-many model there is unrestricted interference by default. The tight control of interference in a point-to-point setting simplifies both specification (programming) and formal reasoning. Thus, our interest in point-to-point communication is methodological: we want to combine the power of nondeterminism and mobility with the simplicity of point-to-point communication.

There are basically two different variants of point-to-point communication. In the first case, the sender and the receiver of a channel remain the same during the whole lifetime of the channel. In the second case, the sender and the receiver of a channel may change. However, at any point in time a channel has not more than one sender and one receiver. In the first case there is no interference at all — two different components cannot send along the same channel. In the second case only a restricted type of interference may occur — two different components may send on the same channel, but never simultaneously. The advantage of the first alternative is its simplicity with respect to formal reasoning and understanding. The advantage of the second alternative is that many things can be expressed more directly. However, the price to pay is a more complicated model. In this paper we concentrate on the first alternative. The second alternative is investigated in [GS96b].

To keep the model simple components are not allowed to forward ports they receive on their input channels. Thus, we cannot change the communication partners of a component. However, we can build up new connections between components that are already connected. These new connections can be in the opposite directions of the already existing ones. In our opinion, this facility of building up new connections is the basic building block of mobility. It allows us to dynamically change the interfaces of components.

Although we could have formulated our semantics in a cpo context, we decided to base it on the topological tradition of *metric spaces* [dBZ82]. 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.

Because of the space limitations, we assume basic knowledge of metric spaces. For more details on metric spaces we refer to the full version of the paper [GS95]. The full version also provides detailed proofs.

The rest of the paper is organized as follows. Section 2 introduces basic notions like communication histories and stream processing functions. Section 3 formalizes the privacy invariants of mobile point-to-point systems. Section 4 introduces mobile components. Section 5 is devoted to composition. Section 6 gives an example. Section 7 contains a discussion. Finally, there is a short appendix containing a metric formalization of streams and named stream tuples.

2 Basic Notions

We model interactive systems by networks 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 transmitted within the first time unit, the second the messages transmitted within the second time unit, and so on. Since time never halts, any complete communication history is infinite.

A message is either a *port* or a *data element*. 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 !. Let N be the set of all channel names and let $C \subseteq N$. Then $?C = \{?c \mid c \in C\}$ is the corresponding set of receive ports and $!C = \{!c \mid c \in C\}$ is the corresponding set of send ports. We also write $?!C$ for $?C \cup !C$. A data element is any message not contained in $?!N$. Let D be the set of all data elements. The set of all complete[2], and partial communication histories for a channel are then characterized by $[(D \cup ?!C)^*]$ and $((D \cup ?!C)^*)^*$, respectively. When no ambiguity occurs we use $[C^*]$ and $(C^*)^*$ as short-hands. This is justified by the convention that D is fixed.

Since ports are exchanged dynamically between network components, each component can in principle access any channel in N. For that reason we model the complete and partial input and output histories of a component by named stream tuples contained in $N \to [C^*]$ and $N \to (C^*)^*$, respectively. 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. The use of named communication histories is inspired by [BD92].

2.2 Guarded Functions

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

$$f \in (N \to [{C_1}^*]) \to (N \to [{C_2}^*])$$

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, represented by ϵ, occurs in the communication history for that channel. The lack of this information causes the fair merge anomaly [Kel78].

The functions process their input *incrementally* — at any point in the time, their output is not allowed to depend on future input. Functions satisfying this constraint are called *weakly guarded*. If the output they produce in time unit t, is not only independent of future input, i.e., the input received during time unit $t+1$ or later, but also of the input received during time unit t, then they are called *strongly guarded*. Intuitively, the strongly guarded functions introduce a

[2] For an arbitrary set S, S^* denotes the set of all finite streams over S, and $[S]$ denotes the set of all infinite streams over S. See also the appendix.

delay of at least one time unit between input and output. The weakly guarded functions allow in addition zero-delay behavior.

For any named communication history θ, let $\theta{\downarrow_j}$ represent the prefix of θ of length j, i.e., the result of cutting θ after the jth time unit. Then weak and strong guardedness can be formalized as below.

Definition 1. (Guarded functions) A function $f \in (N \to [{C_1}^*]) \to (N \to [{C_2}^*])$ is weakly guarded if

$$\forall \theta, \varphi \in (N \to [{C_1}^*]), j \in \mathbb{N} : \quad \theta{\downarrow_j} = \varphi{\downarrow_j} \quad \Rightarrow \quad f(\theta){\downarrow_j} = f(\varphi){\downarrow_j}$$

and strongly guarded if

$$\forall \theta, \varphi \in (N \to [{C_1}^*]), j \in \mathbb{N} : \quad \theta{\downarrow_j} = \varphi{\downarrow_j} \quad \Rightarrow \quad f(\theta){\downarrow_{j+1}} = f(\varphi){\downarrow_{j+1}}$$

We use the arrow $\multimap$ to characterize sets of strongly guarded functions. The actual formulation of guardedness has been taken from [Bro95a].

A weakly guarded function is *non-expansive* and a strongly guarded function is *contractive* with respect to the metric on stream-tuples. This metric is defined in the 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.

In the following sections we introduce two important operators on named communication histories.

2.3 Sum

A *sum* operator takes two named communication histories as input and delivers their "sum" as output. We define both a *partial* "disjoint" sum and a *total* sum. For any $\varphi \in (N \to [C^*])$, let

$$\mathsf{act}(\varphi) = \{i \in N \mid \varphi(i) \neq \epsilon^\infty\}$$

be the set of *active* channels of φ. The partial sum $\varphi + \psi$ is defined if $\mathsf{act}(\varphi)$ is disjoint from $\mathsf{act}(\psi)$.

Definition 2. (Partial sum) Given two named stream tuples $\varphi \in (N \to [{C_1}^*])$ and $\psi \in (N \to [{C_2}^*])$ such that $\mathsf{act}(\varphi) \cap \mathsf{act}(\psi) = \emptyset$. We define their *partial sum* $\varphi + \psi$ to be the element of $N \to [(C_1 \cup C_2)^*]$ such that for all $i \in N$

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

Note that the partial sum has no syntactic conditions assuring its well-definedness. We therefore also define a total version $\varphi \dot{+} \psi$. This simplifies the use of Banach's fix-point theorem. Totalisation is achieved by defining $(\varphi \dot{+} \psi)(i)$ to consist of only ϵ's from the first moment in which both $\varphi{\downarrow_n}$ and $\psi{\downarrow_n}$ are active, i.e., different from ϵ^n, the stream consisting of n ϵ's. For any stream s, by $s(n)$ we denote its nth element.

Definition 3. (Total sum) Given two named stream tuples $\varphi \in (N \to [{C_1}^*])$ and $\psi \in (N \to [{C_2}^*])$. We define their *total sum* $\varphi \twoheadleftarrow \psi$ to be the element of $N \to [(C_1 \cup C_2)^*]$ such that for all $i \in N$, $n \in \mathbb{N}$

$$(\varphi \twoheadleftarrow \psi)(i)(n) = \begin{cases} \psi(i)(n) & \text{if } \varphi(i)\downarrow_n = \epsilon^n \\ \varphi(i)(n) & \text{if } \varphi(i)\downarrow_n \neq \epsilon^n \wedge \psi(i)\downarrow_n = \epsilon^n \\ \epsilon & \text{if } \varphi(i)\downarrow_n \neq \epsilon^n \wedge \psi(i)\downarrow_n \neq \epsilon^n \end{cases}$$

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

Theorem 4. *The total sum operator is weakly guarded.*

Proof. The sum $(\varphi \twoheadleftarrow \psi)(i)(n)$ depends only on $\varphi\downarrow_n$ and $\psi\downarrow_n$.

2.4 Projection

The domain of any named communication history $\theta \in N \to [C^*]$ is N, the set of all channel names. However, in connection with generic functions and network composition, we often need to restrict the visible messages in θ with respect to a history of known channel names $O \in [\mathcal{P}(N)]$. To achieve this we introduce a projection operation $\theta|_O$ which, for each time unit k, replaces the finite stream of messages received during time unit k on each channel contained in $N \setminus O(k)$ by ϵ.

Definition 5. (Projection) For any named communication history $\theta \in (N \to [C^*])$, we define its *projection* $\theta|_O$ on $O \in [\mathcal{P}(N)]$ to be the element of $N \to [C^*]$ such that for all $i \in N, k \in \mathbb{N}$

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

Theorem 6. *The projection operator is weakly guarded.*

Proof. $\theta|_O(i)(k)$ depends only on $\theta\downarrow_k$ and $O\downarrow_k$.

3 Privacy Invariants

A stream processing function $f \in (N \to [{C_1}^*]) \to (N \to [{C_2}^*])$ used to model a component is not only required to be strongly guarded, but also to be *generic* and *point-to-point*. In this section we formalize these additional properties. As already explained, they can be thought of as privacy invariants satisfied by any mobile point-to-point system.

3.1 Genericity

The *genericity* constraint requires a function to access only ports contained in the function's initial, static interface; ports already created by the function itself or ports already received by the function. Genericity can be described with respect to Figure 1, as follows.

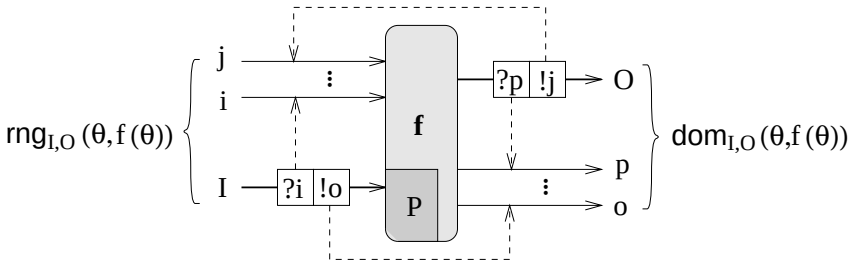

Fig. 1. Generic Stream Processing Function

Initially, each generic function receives on a designated set of input channels I and sends along a designated set of output channels O, disjoint from I. These two sets name the *static* channels or the initial wiring. To make sure that the *dynamic* channels *created* by the different components in a network have different names, each generic 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 computation the sets of accessible ports gradually grow. For example, if the function receives a receive port $?i$ then it may receive on the channel i, and if it receives a send port $!o$ then it may send along 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, and 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 component which receives the receive port.

For a given point in time n and a named input history θ, the sets of accessible input and output channels are represented by respectively $\mathsf{dom}_{I,O}(\theta, f(\theta))(n)$ and $\mathsf{rng}_{I,O}(\theta, f(\theta))(n)$. The functions $\mathsf{dom}_{I,O}$ and $\mathsf{rng}_{I,O}$ are formally defined at the end of the next section.

3.2 Point-to-Point Communication

To ensure the form of point-to-point communication investigated in this paper, the networks have to maintain the following invariant: each channel is used by *at most* two components, the *sender* and the *receiver*. As a consequence, the sender and the receiver of a channel cannot change during the lifetime of channel. This type of point-to-point communication can be captured by a few

simple constraints given that we do not allow forwarding of ports. Firstly, the creator of a channel is allowed to send only one of the channel's ports. If it sends a receive port then it keeps the send port, and the other way around. Secondly, we also insist that the same port is not sent more than once. Since we also restrict received ports from being forwarded, and different components to have disjoint sets of private channels, there is no way in which more than two components can gain access to the same channel.

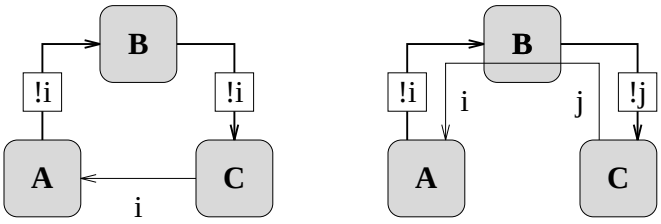

Fig. 2. Forwarding and Point-to-Point Privacy

To explain why we do not allow forwarding, let us have a careful look at a small example. Given a mobile system consisting of three components A, B and C. Assume there is a channel connecting A to B and a channel connecting B to C, but no channel connecting C to A. Now, suppose the component A creates a channel i; it keeps the receive port and sends the send port $!i$ to the component B, which again forwards $!i$ to C. We then obtain the network on the left-hand side of Figure 2. In [GS96b], where we model the more general form of point-to-point communication, we allow forwarding by constraining the functions to "forget" ports as soon as they are sent. A similar technique could have been used here. However, this would make the model more complicated. In fact, the whole advantage gained through the very restrictive communication paradigm would be lost. Since the emphasis in this paper is on a *simple* model we have chosen not to include forwarding. Nevertheless, we are able to express a nontrivial class of mobile networks.

Forwarding can be simulated *straightforwardly*, as indicated by the network on the right-hand side of Figure 2. The component B does not forward $!i$, but a send port $!j$ for a new channel j created by B. Thereafter, any data element B receives on j is forwarded along i. Hence, B "does not receive or send on i itself" — it only forwards the data elements sent by C along j. The component B in the network to the right simulates the "forget-constraint" required for the component B in the network to the left.

One may ask, how do we impose the point-to-point requirement in our model? We do that by imposing an invariant on the named communication histories. The important point to realize is that 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 only once and where two different ports are assigned different channel names. We use the arrow $\xrightarrow{\mathsf{u}}$ to distinguish named communication histories satisfying these two *port uniqueness* constraints from other named communication histories.

Port uniqueness is preserved by projection and summation on stream tuples whose sets of channel names are disjoint. More precisely, for $\theta \in N \xrightarrow{\mathsf{u}} [C^*], \varphi \in N \xrightarrow{\mathsf{u}} [C_1^*]$ and $\psi \in N \xrightarrow{\mathsf{u}} [C_2^*]$ such that $C_1 \cap C_2 = \emptyset$, we have that $\theta|_O \in N \xrightarrow{\mathsf{u}} [C^*]$ and that $\varphi \twoheadleftarrow \psi \in N \xrightarrow{\mathsf{u}} [(C_1 \cup C_2)^*]$.

As a consequence of the forwarding restriction, any port sent by a function has to belong to a channel created by the function. Moreover, since static channels are used for the initial wiring, their corresponding ports cannot be transmitted. For simplicity we split the set of names N into two disjoint sets — a set of *static* channel names S and a set of *dynamic* channel names A. Because of the above restrictions, it is enough to consider functions of the following signature

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

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

We are now ready to give the formal definitions of $\mathsf{dom}_{I,O}$ and $\mathsf{rng}_{I,O}$. In this definition, the operator $\in$ is overloaded to test for containment in a list

Definition 7. (Domain and range) Given $(I, O) \subseteq S \times S$, $P \subseteq A, I \cap O = \emptyset$, $\theta \in (N \xrightarrow{\mathsf{u}} [\overline{P}^*])$ and $\delta \in (N \xrightarrow{\mathsf{u}} [P^*])$. We define

$$\begin{array}{l} D_1 = I \\ R_1 = O \\ D_{n+1} = D_n \cup \bigcup_{i \in D_n} \{p \in A \mid ?p \in \theta(i)(n)\} \cup \bigcup_{i \in R_n} \{p \in A \mid !p \in \delta(i)(n)\} \\ R_{n+1} = R_n \cup \bigcup_{i \in D_n} \{p \in A \mid !p \in \theta(i)(n)\} \cup \bigcup_{i \in R_n} \{p \in A \mid ?p \in \delta(i)(n)\} \end{array}$$

The definitions of $\mathsf{dom}_{I,O}(\theta, \delta)$ and $\mathsf{rng}_{I,O}(\theta, \delta)$ follow immediately

$$\mathsf{dom}_{I,O}(\theta, \delta)(n) = D_n, \quad \mathsf{rng}_{I,O}(\theta, \delta)(n) = R_n$$

Theorem 8. *The functions* $\mathsf{dom}_{I,O}$ *and* $\mathsf{rng}_{I,O}$ *are strongly guarded.*

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

Theorem 9. *The functions* $\mathsf{dom}_{I,O}$ *and* $\mathsf{rng}_{I,O}$ *have the following properties*

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

Proof. By induction on the recursive definitions of $\mathsf{dom}_{I,O}$ and $\mathsf{rng}_{I,O}$.

Genericity can then be formalized as below.

Definition 10. (Generic functions) A function $f \in (N \xrightarrow{\mathsf{u}} [\overline{P}^*]) \to (N \xrightarrow{\mathsf{u}} [P^*])$ is *generic* with respect to the initial wiring (I, O) iff

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

We use the decorated arrow $\stackrel{I,O}{\rightarrow}$ to denote sets of strongly guarded functions that are generic with respect to the initial wiring (I, O). In the following we refer to such functions as *mobile*.

4 Mobile Components

We model a *mobile, 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 11. (Mobile components) A *mobile component*, with initial wiring $(I, O) \subseteq S \times S$ and private names $P \subseteq A$, where $I \cap O = \emptyset$, is a *nonempty* set of mobile functions

$$F \subseteq (N \stackrel{\mathsf{u}}{\rightarrow} [\overline{P}^*]) \stackrel{I,O}{\rightarrow} (N \stackrel{\mathsf{u}}{\rightarrow} [P^*])$$

that is *closed* in the sense that for any mobile function $f \in (N \stackrel{\mathsf{u}}{\rightarrow} [\overline{P}^*]) \stackrel{I,O}{\rightarrow} (N \stackrel{\mathsf{u}}{\rightarrow} [P^*])$

$$(\forall \theta \in N \stackrel{\mathsf{u}}{\rightarrow} [\overline{P}^*] : \exists f' \in F : f(\theta) = f'(\theta)) \Rightarrow f \in F$$

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 we have to a semantics dealing with time.

5 Point-to-Point Composition

We now introduce a *composition operator* $\otimes$ which allows us to compose mobile components into networks of mobile components. When observed from the outside these networks can themselves be understood as mobile components.

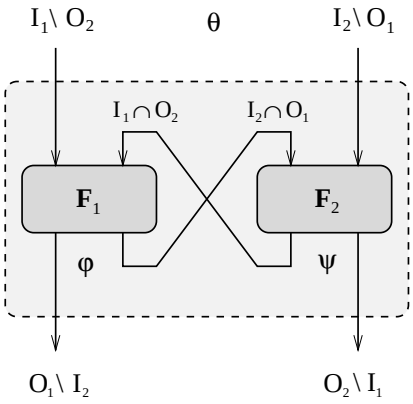

Fig. 3. Point-to-Point Composition

In the formal definition given below we use the operator for total sum. This operator allows us to exploit Banach's fix-point theorem. We later show that this operator can be replaced by the operator for partial sum.

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

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

such that $I_1 \cap I_2 = O_1 \cap O_2 = P_1 \cap P_2 = \emptyset$. Let

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

The *point-to-point composition* of F_1 and F_2 is defined as follows

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

Note the close correspondence between this definition and Figure 3. 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 output channel of F_1, are connected and hidden.

Note also the role of $\mathsf{dom}_{I,O}$ and $\mathsf{rng}_{I,O}$ in maintaining privacy. If F_1 sends a private port $!p$ on a feedback channel, then only F_2 should send along p and only F_1 should receive on p. F_1 can receive on p because dom_{I_1,O_1} is automatically enlarged with p. Only F_1 can receive on p because $\mathsf{rng}_{I,O}$ automatically hides from the environment what F_2 sends along p. F_2 can send along p because rng_{I_2,O_2} is automatically enlarged with p. Only F_2 can influence F_1 via p because $\mathsf{dom}_{I,O}$ automatically hides what the environment sends along p.

Similarly, if F_1 sends a private port $?p$ on a feedback channel, then only F_2 should receive on p and only F_1 should send along p. F_2 can receive on p because dom_{I_2,O_2} is automatically enlarged with p. Only F_2 can receive on p because $\mathsf{rng}_{I,O}$ automatically hides from the environment what F_1 sends along p. F_1 can send along p because rng_{I_1,O_1} is automatically enlarged with p. Only F_1 can influence F_2 via p because $\mathsf{dom}_{I,O}$ automatically hides what the environment sends along p.

Theorem 13. *$F_1 \otimes F_2$ is a mobile component.*

Proof. That $F_1 \otimes F_2 \neq \emptyset$ follows from Banach's fix-point theorem and Theorem 9. Closedness follows straightforwardly.

The definition of $\otimes$ depends on the operator for total sum. This operator is a bit strange since it results in hiding when the 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 operator for total sum by the partial one.

Theorem 14. *The operator $\nplus$ can be replaced by $+$ in the definition of $\otimes$.*

Proof. With respect to Definition 12, we have 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 implies that

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

it is enough to show that the sets $\mathsf{rng}_{I_1,O_1}(\delta \nplus \psi, \varphi)(n)$, $\mathsf{rng}_{I_2,O_2}(\delta \nplus \varphi, \psi)(n)$ and $\mathsf{dom}_{I,O}(\theta, \varphi \nplus \psi)(n)$ are mutually disjoint, for all $n \in \mathbb{N}$. The proof is by induction on the recursive definition of $\mathsf{dom}_{I,O}$ and $\mathsf{rng}_{I,O}$. The induction hypothesis requires both the above disjointness condition and the mutual disjointness of $\mathsf{dom}_{I_1,O_1}(\delta \nplus \psi, \varphi)(n)$, $\mathsf{dom}_{I_2,O_2}(\delta \nplus \varphi, \psi)(n)$ and $\mathsf{rng}_{I,O}(\theta, \varphi \nplus \psi)(n)$.

6 Communication Central

As an example we specify a communication $\mathsf{central}$ (see Figure 4). Its task is to build up connections between $\mathsf{station}_1$ and $\mathsf{station}_2$. The initial "wires" are a_1 and a_2. $\mathsf{Station}_1$ can send ports to be connected (both receive and send) along a_1; $\mathsf{station}_2$ can send ports to be connected (both receive and send) along a_2.

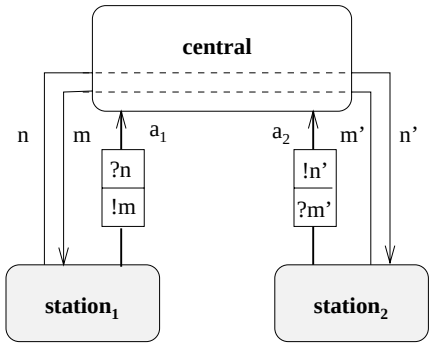

Fig. 4. Communication Central

Let $?n$ be the jth receive port sent along a_1 by $\mathsf{station}_1$. The $\mathsf{central}$ is allowed to receive on n as soon as this port is received. Moreover, let $!n'$ be the jth send port sent along a_2 by $\mathsf{station}_2$. The $\mathsf{central}$ is allowed to send along n' as soon as this port is received. The $\mathsf{central}$ "connects" these two channels by forwarding each data element in D it receives on the channel n along the channel n'. Symmetrically, if $?m'$ is the kth receive port sent along a_2 by $\mathsf{station}_2$, and $!m$ is the kth send port sent along a_1 by $\mathsf{station}_1$, then the data elements in D received on m' are forwarded along m.

In order to model this component, we introduce three basic operators. The first one is a *filter* operator: for any set of messages M and stream of messages s, $M \copyright s$ denotes the stream we obtain by removing any message in s that is not contained in M. The second one is a *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, $\widehat{\beta}$ 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. The timing information is thereby abstracted away.

$$
\begin{array}{l}
\mathrm{Central} = \{ f \in (N \xrightarrow{\mathsf{u}} [A^*])^{\overset{\{a_1,a_2\},\emptyset}{\rightarrow}} (N \xrightarrow{\mathsf{u}} [\emptyset^*]) \mid \\
\quad \forall \alpha : f(\alpha) = \beta \quad \mathsf{where} \\
\qquad \forall (n,n') \in acon : \widehat{\beta}(n') = D \copyright \widehat{\alpha}(n) \qquad -- \text{ the forwarding mechanism} \\
\qquad acon = con(a_1,a_2) \cup con(a_2,a_1) \qquad -- \text{ the set of all connections} \\
\qquad con(a,b) = \{(n,n') \mid \qquad -- \text{ the connections from "}a\text{ to }b\text{"} \\
\qquad\quad \exists k \in [1..\mathsf{min}\{\#\mathrm{rr}(a), \#\mathrm{wr}(b)\}] : \\
\qquad\qquad \mathrm{rr}(a)(k) = ?n \ \wedge \ \mathrm{wr}(b)(k) = !n'\} \\
\qquad \mathrm{rr}(a) = ?N \copyright \widehat{\alpha}(a) \qquad -- \text{ the set of receive ports from } a \\
\qquad \mathrm{wr}(b) = !N \copyright \widehat{\alpha}(b) \qquad -- \text{ the set of send ports from } b \\
\}
\end{array}
$$

Note that this expression does not say anything about the timing of the output. It may be argued that the same behavior could have been obtained in a static network, where an infinite number of channels connect the stations with the central. However, in that case both the central and the stations would be allowed to observe anything that is sent along the channels. This should be contrasted with our model, where the components are allowed to access only the channels whose ports they have received or created themselves. In our opinion, it is exactly this privacy that, not only captures the essence of mobility, but also simplifies the conceptual reasoning about mobile reconfiguration.

7 Discussion

The main contribution of this paper is that we have extended a denotational model for timed, point-to-point, nondeterministic data-flow networks to handle a notion of *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 [MPW92] and actor-based approaches [AMST92] is a interesting area for future research. For example, we believe that the model for many-to-many communication [GS96a] 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 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 behaviors of basic agents. This guarantees the existence of fix-points. We use sets of strongly guarded functions for the same purpose.

[Gro94, Bro95b] give equational characterizations of dynamic reconfiguration with respect to stream processing functions.

8 Acknowledgments

The authors would like to thank Manfred Broy for many inspiring discussions on this and related topics. The first author has been financially supported by the Syslab project. The second author has been financially supported by the Sonderforschungsbereich 342 “Werkzeuge und Methoden für die Nutzung paralleler Rechnerarchitekturen”.

References

[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.

[BD92] M. Broy and C. Dendorfer. Modelling operating system structures by timed stream processing functions. *Journal of Functional Programming*, 2:1–21, 1992.

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

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

[Bro95a] M. Broy. Advanced component interface specification. In *Proc. TPPP'94, Lecture Notes in Computer Science 907*, pages 369–392, 1995.

[Bro95b] M. Broy. Equations for describing dynamic nets of communicating systems. In *Proc. 5th COMPASS Workshop, Lecture Notes in Computer Science 906*, pages 170–187, 1995.

[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.

[Gro94] R. Grosu. *A Formal Foundation for Concurrent Object Oriented Programming.* PhD thesis, Technische Universität München, 1994. Also available as report TUM-I9444, Technische Universität München.

[GS95] R. Grosu and K. Stølen. A denotational model for mobile point-to-point dataflow networks. Technical Report SFB 342/14/95 A, Technische Universität München, 1995.

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

[GS96b] 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.

[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.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, part I. *Information and Computation*, 100:1–40, 1992.

[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 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 15. (The metric space of streams) The metric space 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 16. *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 space of named stream tuples) The metric space of *named stream tuples* $(I \to [E], d)$ with names in I and elements in (E, ρ) is defined as follows

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

where $I \to [E]$ is the set of functions from the countable set I to the metric $[E]$.

Theorem 18. *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].

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