Formal Aspects of Computing (2001) 13: 1–31
© 2001 BCS

Formal Aspects of Computing

Stream-Based Specification of Mobile Systems

Radu Grosu[1] and Ketil Stølen[2]

[1]Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY, USA
[2]SINTEF Telecom and Informatics, Oslo, Norway

Abstract. This paper presents a formal specification technique for mobile systems based on input/output relations on streams. We consider networks of components communicating asynchronously via unbounded directed channels. Mobility is achieved by allowing the components to communicate channel ports. We distinguish between many-to-many and two variants of point-to-point communication. The communication paradigms are semantically underpinned by denotational models. The models are formulated in the context of timed non-deterministic dataflow networks and presented in a stepwise fashion. The emphasis is on capturing the special kind of dynamic hiding characterising mobile systems. We demonstrate the proposed approach in a number of small examples.

Keywords: Denotational model; Input/output relation; Many-to-many communication; Mobile system; Point-to-point communication; Specification

1. Introduction

Motivated by the need to model object-oriented programming languages and openness in distributed applications, the study of mobile systems has become a very popular research area. Most of the early theoretical research on mobility is of a rather operational nature; see for instance [HBS73, EnN86, Tho89, BeB90, Mes91, MPW92]. A denotational understanding of mobility is, however, an essential prerequisite for modular development of mobile, and consequently object-oriented reactive systems. Recently several researchers have studied mobility in a denotational setting; see for example [JaJ95, FMS96, Sta96]. These denotational approaches are all directed towards the π-calculus and use a quite involved type theory. In this paper we look at mobility from a different angle; our objective is to build a specification formalism for mobile systems based on streams.

As usual in the case of natural language concepts, there is some disagreement with respect to what it actually means for a system to be mobile. In this paper we stick to the definition of Robin Milner: a mobile system is a system in which every component may change its communication partners on the basis of

Correspondence and offprint requests to: Radu Grosu, Department of Computer Science, State University of New York at Stony Brook, Stony Brook, NY 11794-4400, USA. Email: grosu@cs.sunysb.edu, Ketil.Stoelen@informatics.sintef.no

computation and interaction [Mil91]. This means, for example, that this paper is not concerned with the kind of mobility achieved by allowing components to communicate (migrate) components (although we believe this can be simulated by the communication of ports).

The use of *input/output relations* (I/O relations) to specify computerised components is well known. For example, VDM [Jon90] and Z [Spi88] are both based on this approach: a specification of a sequential component C characterises the relationship between its initial and final states. The initial state can be understood as the input of C produced by C's environment before the execution of C is initiated. The final state can be understood as the output produced by C itself.

Interactive and purely reactive components can be specified in a similar manner. For example, the Focus method [BrS01] for formal specification and design of interactive systems is based on I/O relations: a specification of a reactive component C characterises the relationship between its tuples of input and output streams. A tuple of input streams represents histories of input messages sent by C's environment along C's input channels. A tuple of output streams represents histories of output messages sent by C itself along C's output channels.

The main difference between ordinary interactive systems and mobile systems is the latter's much more sophisticated concept of hiding. In mobile systems the scope of identifiers changes dynamically at run-time. Hence, we need notions of hiding that, on the one hand, are sufficiently flexible to allow this kind of dynamic scoping, and, on the other hand, are sufficiently expressive to disallow undesirable visibility. The notion of hiding required is highly dependent upon the underlying communication paradigm. We demonstrate the importance of this by studying mobility with respect to three different communication paradigms: asynchronous *many-to-many* (m2m) communication and two variants of asynchronous *point-to-point* (p2p) communication.

In the m2m case several components may simultaneously output messages along the same channel, and several components may simultaneously input messages from the same channel. In the p2p case we distinguish between p2p communication *with* and *without* channel sharing.

In the case of p2p communication with channel sharing, a channel may have several receivers and also several senders, but never at the same time: at any point in time, a channel has exactly one sender and exactly one receiver. However, since channel ports can be forwarded from one component to another, the identities of the sender and the receiver may change during computation; a channel port is immediately forgotten by the forwarding component.

Ports can also be forwarded in the case of p2p communication without channel sharing. However, this is allowed only until the communication on the channel is started up. Thus, in this case, the sender and the receiver of a channel remain the same during the whole computation. P2p communication with channel sharing can be understood as a special case of m2m communication. Moreover, p2p communication without channel sharing can be understood as a special case of p2p communication with channel sharing.

As already mentioned, Focus is based on I/O relations on streams. Focus is semantically underpinned by a denotational model expressed in the form of a timed non-deterministic dataflow network. In this respect our approach is similar to Focus. Our approach generalises the I/O relations of Focus to handle mobility defined as dynamic network reconfiguration resulting from the communication of ports. We treat m2m as well as both variants of p2p communication.

We consider networks of autonomous components communicating and interacting via directed channels in a time-synchronous and message-asynchronous manner. 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. Mobility is achieved by allowing the components to communicate ports.

We distinguish between three specification formats — one for each communication paradigm. They are syntactically distinguished by labelling keywords. Each specification format allows a wide variety of mobile systems to be described. The particular choice of format for a given application depends on the nature of the application and the invariants to be maintained. To allow the reader to appreciate these differences, we specify several variants of the mobile telephone network discussed in [Mil91]: an m2m variant in Example 3 and p2p variants in Examples 4 and 5.

The paper is organised as follows. In Section 2 we introduce some basic notions and corresponding notation; in Section 3 we introduce the model for m2m communication and build a specification language on top of it; in Section 4 we do the same for the two variants of p2p communication; in Section 5 we sum up our results and relate our approach to the literature. There are also four appendices: in Appendix A we define the underlying metrics; in Appendix B we prove some results for the m2m model; in Appendix C we do the same for the p2p models; in Appendix D we relate the p2p and the m2m models.

2. Basic Notions

As mentioned in the introduction, our approach is based on streams. In this section we introduce notation for the description, manipulation and composition of streams.

2.1. Communication Histories

A *stream* is a sequence of elements of some type E. E^*, E^∞ and E^ω are the sets of finite, infinite and finite as well as infinite streams over E, respectively. 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 fixed least unit of time. M is the set of all messages; hence, $(M^*)^\infty$ and $(M^*)^*$ are, respectively, the sets of all *complete* and *partial* communication histories. In the sequel, by communication histories we mean complete communication histories unless otherwise stated.

A port is a *channel name* together with an *access right*, which is either an *input* right, represented by ?, or an *output* right, represented by !. Hence, if N is the set of all channel names, then $?N \equiv \{?i \mid i \in N\}$ is the corresponding set of input ports, $!N \equiv \{!i \mid i \in N\}$ is the corresponding set of output ports and $?!N \equiv ?N \cup !N$ is the set of all ports. We assume that $?!N \subseteq M$. $D \equiv M \setminus ?!N$ is the set of all messages not contained in the set of ports. For any $n \in N$ and $S \subseteq ?!N$, we define:

$$\widetilde{!n} \equiv ?n, \qquad \widetilde{?n} \equiv !n, \qquad \overline{S} \equiv ?!N \setminus S, \qquad \widetilde{S} \equiv \{\widetilde{p} \mid p \in S\}$$

Since components exchange ports, each component can potentially access any channel in N. For that reason we model the *input* and the *output histories* of a component by functions of the following signature: $N \to (M^*)^\infty$. We refer to these functions as *named communication histories*, or just *histories*. In the sequel we use H to denote this set.

2.2. Guarded Functions

We model *deterministic components* by functions $f \in H \to H$ mapping input histories to output histories, often referred to as *stream processing functions*. We model *non-deterministic components* by sets of such functions. The functions process their inputs *incrementally*: at any point in time, their outputs are independent of their future inputs. Such functions are called *weakly guarded*. If the outputs the functions produce in time unit t are not only independent of future inputs — the inputs received during time unit $t+1$ or later — but also of the inputs received during time unit t, the functions are called *strongly guarded*. Intuitively, the strongly guarded functions introduce a delay of at least one time unit between input and output; the weakly guarded functions also allow zero-delay behaviour.

In the following, *Nat* denotes the set of natural numbers and Nat_+ the set $Nat \setminus \{0\}$. We also identify $(M^*)^\infty$ with the set of total functions $Nat_+ \to M^*$. For any $t \in Nat_+$ and $r \in E^\infty$, by $r{\downarrow_t}$ we denote the prefix of r consisting of exactly t elements. By $r{\downarrow_0}$ we denote $\langle\rangle$, the empty stream. This operator is overloaded to H in the obvious manner: for any $\theta \in H$, $\theta{\downarrow_t}$ is obtained from θ by substituting $\theta(n){\downarrow_t}$ for $\theta(n)$ for each $n \in N$.

Definition 1 (Guarded function). A function $f \in H \to H$ is weakly guarded if

$$\forall\, \theta, \varphi \in H\,;\ t \in Nat : \theta{\downarrow_t} = \varphi{\downarrow_t} \Rightarrow f(\theta){\downarrow_t} = f(\varphi){\downarrow_t}$$

and strongly guarded if

$$\forall\, \theta, \varphi \in H\,;\ t \in Nat : \theta{\downarrow_t} = \varphi{\downarrow_t} \Rightarrow f(\theta){\downarrow_{t+1}} = f(\varphi){\downarrow_{t+1}}$$

Strongly and weakly guarded functions are also known as respectively *contractive* and *non-expansive* functions with respect to the Baire metric (see [Eng89] and Appendix A). It is well known that the functional composition of a contractive and a non-expansive function is a contractive function. Since by the Banach's fix-point theorem, each contractive function has a unique fix-point, the functional composition of a strongly and a weakly guarded function also has a unique fixed point. As we see later, this assures that our composition operators are well defined.

2.3. Notational Conventions

In this section we introduce some helpful notation. For any n-tuple of elements w, stream of elements s, set of elements A and $j \in Nat_+$:

- $\langle\rangle$ is the empty stream;
- $\pi_j(w)$ is the jth element of w if $1 \leqslant j \leqslant n$;
- $\#s$ is the length of s;
- $s(j)$ is the jth element of s if $1 \leqslant j \leqslant \#s$;
- $\langle a_1, \dots, a_j\rangle$ is the stream of length j starting with element a_1 followed by a_2, a_3 and so on;
- $A \circledS s$ is the stream obtained from s by removing any element in s not contained in A; for instance, $\{a,b\} \circledS \langle a,b,c,d,a\rangle = \langle a,b,a\rangle$.

The $\circledS$ operator is overloaded to sets of pairs of messages $X \subseteq A \times B$ and pairs of streams (r,s) of the same length in a straightforward manner. For each t, $(r(t), s(t))$ is filtered away iff it is not in X. For instance,

$$\{(a,b),(a,a)\} \circledS (\langle a,a,b,b\rangle, \langle a,b,b,a\rangle) = (\langle a,a\rangle, \langle a,b\rangle)$$

For any $s \in M^*$, we define $\mathsf{pt}(s)$ to denote the set of all ports contained in s.

3. Many-to-Many Communication

In this section we consider m2m communication. We start by defining static networks; then we show that mobility can be understood as a *privacy preserving* property of stream processing functions; finally, we define components in terms of such functions, introduce operators for parallel composition and hiding and build a small specification language on the top of this formalism.

3.1. Static Many-to-Many Networks

In static m2m networks, each component interacts over a fixed and possibly shared, set of input and output channels. As a consequence, the channel topology of a static network does not vary over time. Considering I and O to be sets of input and output channel names, respectively, the strongly guarded functions used to represent static m2m networks are of the form $f \in In \to Out$ where $In = I \to (D^*)^\infty$ and $Out = O \to (D^*)^\infty$.

3.1.1. Privacy Preservation

To give a uniform treatment of static and mobile components we consider, however, strongly guarded functions $f \in H \to H$ defined over the whole set of channel names N and require that they communicate only over channels with names in I and O.

Definition 2 (Domain and range). For any $t \in Nat_+$; $I, O \subseteq N$; $\theta, \delta \in H$; the domain and range at time t are characterised by

$$\mathsf{dmSM}_{I,O}(\theta)(i)(t) \equiv \begin{cases} \theta(i)(t) & \text{if } i \in I \\ \langle\rangle & \text{otherwise} \end{cases}$$

$$\mathsf{rnSM}_{I,O}(\delta)(i)(t) \equiv \begin{cases} \delta(i)(t) & \text{if } i \in O \\ \langle\rangle & \text{otherwise} \end{cases}$$

The restriction to the appropriate sets of channels is characterised by the following definition.

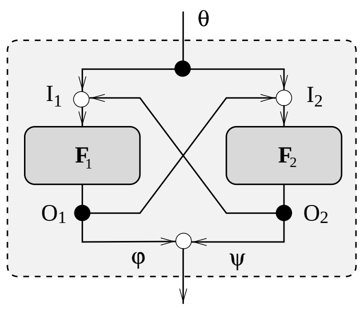

Fig. 1. Static many-to-many composition.

Definition 3 (Privacy preserving function). A function $f \in H \to H$ preserves static privacy with respect to $I, O \subseteq N$ iff

$$\forall\, \theta \in H\ : f(\theta) = f(\mathsf{dmSM}_{I,O}(\theta)) = \mathsf{rnSM}_{I,O}(f(\theta))$$

Informally speaking, dmSM makes sure that f inputs only on its input ports; rnSM makes sure that f outputs only along its output ports. We use $Stat_{m2m}(I, O)$ to denote the set of all strongly guarded functions that preserve static privacy with respect to (I, O). In the sequel we refer to such functions as *static m2m functions.*

Any strongly guarded function $f \in H \to H$ can be transformed into a static m2m function $sm2m_{I,O}(f) \in Stat_{m2m}(I, O)$ as follows.

$$sm2m_{I,O}(f)(\theta) = \mathsf{rnSM}_{I,O}(\delta) \text{ where } \delta = f(\mathsf{dmSM}_{I,O}(\theta))$$

3.1.2. Static Many-to-Many Components

We model *static m2m components* by sets of static m2m functions.

Definition 4 (Static m2m component). A static m2m component with interface (I, O) is represented by a non-empty set of static m2m functions $F \subseteq Stat_{m2m}(I, O)$.

Any pair $(\theta, f(\theta))$ such that $f \in F$ is a possible *input/output history* of the component F; $\theta(c)$ is the history of all messages sent by its environment along the channel c; similarly, $f(\theta)(c)$ is the history of all messages sent along c by the component itself. Thus, although we model m2m communication, each component is represented by a pure input/output relation, where each input history contains only messages sent by the environment and each output history contains only messages sent by the component. We use $SComp_{m2m}(I, O)$ to denote the set of all static m2m components with respect to (I, O).

3.1.3. Static Many-to-Many Composition

The *parallel composition* of two static m2m components F_1 and F_2 is illustrated by the network in Fig. 1. The hollow circles denote *interference points*, i.e., points where output from the environment, F_1, or F_2 sent during the same time unit is interleaved. In our approach, interference is modelled by building copies of a *merge node* into the interference points and, therefore, implicitly into the network operators. This allows composition to be described in a very abstract and, in our opinion, intuitive manner. The merge node $\mathbb{M}$ takes two named communication histories as input and yields a merge as output. Any occurrence of $\mathbb{M}$ is hidden in the semantic definition of the network operators. Since we want the network operators to preserve causality (and, in principle, also support the specification of timing constraints, although this plays no role in this paper), $\mathbb{M}$ should neither add nor reduce delay. This means that the output history of $\mathbb{M}$ for some channel n during time unit k must be a merge of the two finite streams characterising the input histories on n in time unit k. Moreover, $\mathbb{M}$ should not fix the interleaving. Thus, any interleaving of the messages received within a time unit should be allowed. Hence, $\mathbb{M}$ is non-deterministic in the sense that a pair of input histories may result in several (often infinitely many) different output histories.

The definition below formalises what it means for a finite stream to be a merge of two finite streams. The *oracle* p 'marks' the messages in the output stream with 1 if they occurred in the first stream and with 2 if they occurred in the second stream.

Definition 5 (Merge function on finite streams). FM is the set-valued function such that

$$FM \in M^* \times M^* \to \mathscr{P}(M^*)$$

$$\begin{aligned} FM(s_1,s_2) = \{\, s \in M^* \mid \exists\, p \in \{1,2\}^* : \; & \#p = \#s && \wedge \\ & s_1 = \pi_1[(M \times \{1\}) ⓢ (s,p)] && \wedge \\ & s_2 = \pi_1[(M \times \{2\}) ⓢ (s,p)] && \} \end{aligned}$$

where $\mathscr{P}(S) \equiv \{T \mid T \subseteq S \wedge T \neq \{\}\}$ is the set of non-empty subsets of S.

It is now straightforward to define the merge node.

Definition 6 (Merge node). $\mathbb{M}$ denotes the set of all functions $f \in H \times H \to H$ such that

$$\forall\, \varphi, \psi \in H\,;\; n \in N\,;\; t \in Nat_+ : f(\varphi,\psi)(n)(t) \in FM(\varphi(n)(t), \psi(n)(t))$$

Note that each $f \in \mathbb{M}$ is weakly guarded since the output produced during any time unit t depends only on the input received during the same time unit t. Note also that $\mathbb{M}$ is deterministic (it yields a singleton set) if the two input histories are chosen such that

$$\forall\, n \in N\,;\; t \in Nat_+ : \varphi(n)(t) = \langle\rangle \vee \psi(n)(t) = \langle\rangle$$

Now, we are ready to give the formal definition of the static m2m composition. Note the close relationship to Fig. 1.

Definition 7 (Static m2m composition). Given two static m2m components

$$F_1 \subseteq SComp_{m2m}(I_1, O_1), \quad F_2 \subseteq SComp_{m2m}(I_2, O_2)$$

Let

$$I \equiv I_1 \cup I_2, \quad O \equiv O_1 \cup O_2$$

We define the m2m composition of F_1 and F_2 as follows.

$$\begin{aligned} F_1 \odot F_2 \equiv \{\; f \in H \to H \;\mid\; & \exists f_1 \in F_1; f_2 \in F_2;\; m_1, m_2, m_3 \in \mathbb{M} : \forall\, \theta \in H : \\ & f(\theta) = m_3(\varphi,\psi) \text{ where } \varphi = f_1(m_1(\theta,\psi)),\; \psi = f_2(m_2(\theta,\varphi)) \;\} \end{aligned}$$

It follows straightforwardly from Theorem 6 that $F_1 \odot F_2$ is a static m2m component in $SComp_{m2m}(I, O)$.

The definition of the parallel operator may seem strange in the sense that the messages that a component sends will not be received as input by the same component. Assume for example that a component S_1 has both an input and an output port for the channel c in its initial interface. If we compose S_1 with a component S_2 then the messages sent by S_1 along $!c$ can be received by S_2 and the overall environment, but not by S_1 itself. The reason why $\odot$ has been defined without local feedback is that in the m2m case there is no implicit hiding when components are composed. Hence, if $\odot$ had been redefined to support local feedback, the result of composing $S_1 \odot S_2$ with a third component S_3 would be that S_1 receives each message sent by S_1 along $!c$ not once, but twice — once through the composition with S_2 and once through the composition of $S_1 \odot S_2$ with S_3. This is clearly not desirable. One way to avoid this problem is to define composition as we do and in addition introduce a special operator for local feedback, as suggested in [GrS96a]. In this paper we do not consider local feedback since, as carefully explained in [GrS96a], the operator for local feedback is just a simplified version of our composition operator. A detailed treatment of this operator would therefore not add much to the paper.

3.1.4. Explicit Hiding

To hide ports we use an explicit *hiding operator*. If Q is a set of channel names, then $\nu Q.F$ is the component obtained from F by deleting Q from I and O. The domain and range of the static m2m functions modelling $\nu Q.F$ are modified accordingly.

Definition 8 (Hiding). Given a static m2m component $F \subseteq Stat_{m2m}(I, O)$ and a set of channel names Q. Then $\nu Q.F$ is defined as below:

$$I' \equiv I \setminus Q, \qquad O' \equiv O \setminus Q$$

$$\nu Q.F \equiv \{sm2m_{I',O'}(f) \mid f \in F\}$$

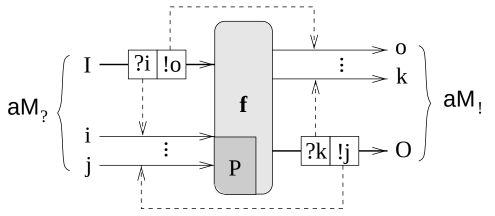

Fig. 2. Privacy preserving function.

It is easy to show that that $vQ.F$ belongs to $SComp_{m2m}(I', O')$.

3.2. Mobile Many-to-Many Networks

In the mobile case the domain and range of the strongly guarded functions $f \in H \rightarrow H$ may vary over time.

3.2.1. Privacy Preservation

In the mobile m2m case the privacy preservation property formalises the rules for how components may gain access to ports. We may think of this property as an invariant for mobile m2m communication. The behaviour of a privacy preserving function f can be described with respect to Fig. 2 as follows. Initially, f inputs on a designated set of input ports $?I$ and outputs along a designated set of output ports $!O$. These two sets identify the *initial interface* of the component modelled by f; we often refer to it as (I, O). To make sure that channels *created* by different components in a network have different names, the function f is also assigned an initial set of private channel names P known only by the component modelled by f. The ports in $?!P$ are *passive*; the ports in the initial interface are *active*. By aM_t we denote the set of active ports at time t; by pM_t the set of passive ports at time t. Initially, we have that $\mathsf{aM}_1 = ?I \cup !O$ and $\mathsf{pM}_1 = ?!P$. Obviously, the initial set of passive ports should be disjoint from the initial set of active ports; thus, we require that $(I \cup O) \cap P = \{\}$.

During the computation, the number of active ports gradually increases and the number of passive ports gradually decreases. For example, if the function f inputs a port $?i \notin \mathsf{pM}_t$ on an input port it already knows, then it may later also input messages on $?i$; if it inputs a port $!o \notin \mathsf{pM}_t$ on an input port it already knows then it may also later output messages along $!o$. Accordingly, whenever the function f outputs a passive port $!j \in \mathsf{pM}_t$, it may later input on $?j$ what the components that received $!j$ output along j; whenever the function f outputs a passive port $?k \in \mathsf{pM}_t$, it may itself output messages along $!k$ that eventually are input by the components that received $?k$. Hence, a port p remains passive as long as its complement port $\widetilde{p}$ is not known by the environment. If $\widetilde{p}$ is not known by the environment, then the environment has no means to interact with f along p.

Let θ and δ denote the input and output of f, respectively. The active and passive ports of f are formally characterised below.

Definition 9 (Active and passive ports). For any $I, O, P \subseteq N$; $\theta, \delta \in H$; $t \in Nat_+$; let aM and pM be defined recursively as follows.

$$\mathsf{aM}_1 \equiv ?I \cup !O, \quad \mathsf{pM}_1 \equiv ?!P, \quad \mathsf{aM}_{t+1} \equiv \mathsf{aM}_t \cup \mathsf{rM}_t \cup \mathsf{gM}_t, \quad \mathsf{pM}_{t+1} \equiv \mathsf{pM}_t \setminus \mathsf{gM}_t$$

where

$$\begin{aligned} \mathsf{rM}_t &\equiv \textstyle\bigcup_{?i \in \mathsf{aM}_t} \{p \mid p \in \overline{\mathsf{aM}_t \cup \mathsf{pM}_t} \wedge p \in \mathsf{pt}(\theta(i)(t))\} \\ \mathsf{gM}_t &\equiv \textstyle\bigcup_{!i \in \mathsf{aM}_t} \{p \mid p \in \mathsf{pM}_t \wedge \widetilde{p} \in \mathsf{pt}(\delta(i)(t))\} \end{aligned}$$

Then the sets of active and passive ports at time t are characterised by:

$$\mathsf{aM}_{I,O,P}(\theta, \delta)(t) \equiv \mathsf{aM}_t, \qquad \mathsf{pM}_{I,O,P}(\theta, \delta)(t) \equiv \mathsf{pM}_t$$

The sets rM_t and gM_t are the sets of *received* and *generated ports*, respectively. If the sets of active and passive ports are disjoint initially, then they are also disjoint at any later point in time. Note that

$$\mathsf{aM}_{t+1} \cup \mathsf{pM}_{t+1} = \mathsf{aM}_t \cup \mathsf{pM}_t \cup \mathsf{rM}_t$$

In the definition of privacy preservation (Definition 11) we use the functions dmM and rnM (Definition 10) to constrain f to maintain the privacy invariant with respect to active and passive ports described above. The functions dmM and rnM characterise the input and output histories that are actually considered by f.

Since the function f runs in an open environment this privacy invariant is not sufficient unless also the environment sticks to the rules of the game. There are basically two ways the environment of f can break the rules of the game. First, the environment can *output a port* $p \in \widetilde{\mathsf{pM}_t}$ that it has *not yet received* from f (its dual port $\widetilde{p} \in \mathsf{pM}_t$ is passive). Remember that sending a private port p automatically activates its dual $\widetilde{p}$. In that case the environment does not yet know p because it has not yet been output by f. Second, the environment can *output along a port* $!i \in \widetilde{\mathsf{pM}_t}$ it has *not yet received* (its dual port $?i \in \mathsf{pM}_t$ is passive and, therefore, not in aM_t).

There are several ways to deal with this problem. One alternative is to use a typing discipline that assures that the function is well typed only if the environment never breaks the rules of the game; a second alternative is to impose an environment assumption in all definitions characterising exactly those input histories in which the environment sticks to the rules of the game; a third alternative, which is used in this paper, is to constrain dmM and rnM to ignore the input messages that do not respect the privacy restrictions.

This solution is simpler than the other two and it is satisfactory because we are only interested in environments that can be understood as components in the sense of this paper; such components will never break the rules of the game. For that reason, the functions dmM and rnM are defined in such a way that they, in addition to their main task of characterising the actual domain and range of a function, also correct environment mistakes.

Definition 10 (Domain and range). For any $I, O, P \subseteq N$; $\theta, \delta \in H$; $t \in Nat_+$; the domain and range at time t are characterised by:

$$\mathsf{dmM}_{I,O,P}(\theta,\delta)(i)(t) \equiv \begin{cases} (\overline{\widetilde{\mathsf{pM}_t}} \cup D) \circledS \theta(i)(t) & \text{if } ?i \in \mathsf{aM}_t \\ \langle\rangle & \text{otherwise} \end{cases}$$

$$\mathsf{rnM}_{I,O,P}(\theta,\delta)(i)(t) \equiv \begin{cases} (\mathsf{pM}_t \cup \mathsf{aM}_t \cup D) \circledS \delta(i)(t) & \text{if } !i \in \mathsf{aM}_t \\ \langle\rangle & \text{otherwise} \end{cases}$$

where $\mathsf{aM}_t \equiv \mathsf{aM}_{I,O,P}(\theta,\delta)(t)$ and $\mathsf{pM}_t \equiv \mathsf{pM}_{I,O,P}(\theta,\delta)(t)$.

We can now define what it means for a function to be privacy preserving in the mobile m2m case.

Definition 11 (Privacy preserving function). A function $f \in H \to H$ is privacy preserving with respect to $I, O, P \subseteq N$ iff

$$\forall\, \theta \in H\ : f(\theta) = f(\mathsf{dmM}_{I,O,P}(\theta, f(\theta))) = \mathsf{rnM}_{I,O,P}(\theta, f(\theta))$$

Informally speaking, dmM makes sure that f inputs on its active input ports only and ignores the ports that are not known by its environment (since pM_t contains passive ports, its dual $\widetilde{\mathsf{pM}_t}$ is not known by the environment); rnM makes sure that f outputs along its active ports only and never sends a port not contained in its sets of active and passive ports.

Privacy preservation is intimately related to the notion of time. For each port p received (passive port p sent) for the first time in time unit t, the function f may communicate via p (via $\widetilde{p}$) from time unit $t+1$ onwards. Note that such a causality relation cannot be expressed in an untimed input/output model.

We use $Mob_{m2m}(I, O, P)$ to denote the set of all strongly guarded functions that are privacy preserving with respect to (I, O, P). In the sequel we refer to such functions as mobile *m2m functions.*

In Appendix B (Theorem 5) we prove that any strongly guarded function $f \in H \to H$ can be transformed into a mobile m2m function $m2m_{I,O,P}(f) \in Mob_{m2m}(I, O, P)$ as follows.

$$m2m_{I,O,P}(f)(\theta) = \mathsf{rnM}_{I,O,P}(\theta,\delta) \text{ where } \delta = f(\mathsf{dmM}_{I,O,P}(\theta,\delta))$$

3.2.2. Mobile Many-to-Many Components

We model *mobile m2m components* by sets of mobile m2m functions.

Definition 12 (Mobile m2m component). A mobile m2m component with initial interface (I,O) and initial set of passive channel names P is represented by a non-empty set of m2m functions $F \subseteq Mob_{m2m}(I, O, P)$.

We use $Comp_{m2m}(I, O, P)$ to denote the set of all mobile m2m components with respect to (I, O, P).

3.2.3. Typed Channels and Tuple Messages

The mobile m2m model defines mobility in a simple and elegant way. To this end, we deliberately ignored some practical aspects like

- typed channels and ports;
- tuple messages consisting of both ordinary messages and typed ports.

The usefulness of the first extension should be obvious; the second allows us to bind a port to a message — for example, the message may be some request whose reply should be sent to a particular component identified by the port. In this section we outline how the model can be extended to handle these aspects.

Let T be the set of all types. Each channel is assigned a type by the function

$$type \in N \to T$$

This function is overloaded to ports in the obvious manner:

$$type(?n) \equiv type(n), \qquad type(!n) \equiv type(n)$$

To accommodate tuple messages, we assume that any finite tuple of messages from M is itself a member of M; accordingly, any finite Cartesian product of elements from T is itself an element of T. H_T is the set of communication histories that are type-correct according to *type*. Formally,

$$H_T \equiv \{\theta \in H \mid \forall n \in N : \theta(n) \in (type(n)^*)^\infty\}$$

Definitions 9, 10, 11 and 12 carry over straightforwardly: dmM and rnM are redefined to look for ports inside tuple messages. The two extensions outlined in this section are straightforward but result in more complicated definitions, thereby reducing the readability of the paper; for this reason, we work with the basic model (without the two extensions) when we define operators for parallel composition and hiding.

3.2.4. Elementary Many-to-Many Specifications

The next step is to build a specification language on top of the model introduced above. This language is presented in an example-driven manner. Since the m2m model is timed, we can easily handle time constraints. Nevertheless, since this paper is concerned with the specification of mobility and not with the specification of timing, we abstract away the timing and work with untimed streams when we write specifications. H_A is the set of all (abstract) *untimed* type-correct communication histories. For any $\theta \in H_T$, by $\mathsf{ta}(\theta)$ we denote its *time-abstraction*: the element in H_A obtained from θ by concatenating the finite substreams in each infinite stream into a stream of messages. For instance, given that $\frown$ is the concatenation operator for streams, we have

$$\forall n \in N : \mathsf{ta}(\theta)(n) = \theta(n)(1) \frown \theta(n)(2) \frown \ldots \frown \theta(n)(j) \frown \ldots$$

We start by specifying the behaviour of a consultant that communicates with customers via some communication system.

Example 1. Specification of a consultant:
We consider the following scenario. A number of consultants reply to questions posed by customers; the consultants are connected to an administrator that inputs questions and distributes them to the consultants depending on workload, specialisation and experience; each question forwarded by the administrator to a consultant is accompanied by the output port along which the reply is to be sent. A consultant is specified, as follows.

Con m2m

in $c : (Q \times !N)$

$con(\mathsf{in}) = \mathsf{out}$

where $\forall o \in N ;\ q \in Q ;\ v \in H_A :$

$$con(\{c \mapsto (q, !o)\} \,\&\, v) = \{o \mapsto r(q)\} \,\&\, con(v)$$

Con is the name of the specification. The upper-most frame declares the initial interface. Thus, initially the consultant has access to only one port, namely the input port $?c$ on which it inputs questions and their associated output ports from the administrator. Its set of output ports is initially empty. The lower-most frame, called the *body*, describes the dynamic behaviour by a function *con* defined by the where-clause. In any elementary specification, $\mathsf{in} \in H_A$ represents the input history and $\mathsf{out} \in H_A$ represents the output history. For example, $\mathsf{in}(c)$ is the input history for the channel c. The function r describes the replies made by the consultant; since this paper is concerned with communication and not with computation, the latter is left unspecified; a consultant differs from another consultant in the choice of r. By $\{n \mapsto m\} \,\&\, \theta$ we denote the result of appending m to the head of the stream $\theta(n)$ and leaving the rest of θ unchanged. By $\{n \mapsto m_1, \dots, m_k\} \,\&\, \theta$ we mean $\{n \mapsto m_1\} \,\&\, \dots \,\&\, \{n \mapsto m_k\} \,\&\, \theta$. □

We assume that each specification S has associated a unique, infinite set of (initially) private channel names P_s. As shown later, in Example 5, this set is referenced by the keyword priv. The *semantics* of an elementary specification S with external interface (I, O) and body B is then defined as follows.

$$[\![\ S\ ]\!] \equiv \{\ \ g \in Mob_{m2m}(I, O, P_s) \mid \forall i' \in H_T : \exists o' \in H_T : o' = g(i') \wedge B(\mathsf{in}, \mathsf{out}) \quad \text{where } \mathsf{in} = \mathsf{ta}(\mathsf{dmM}_{I,O,P}(i', o')),\ \mathsf{out} = \mathsf{ta}(o')\ \ \}$$

In the above definition, the fact that g is a mobile m2m function enforces that $o' = \mathsf{rnM}_{I,O,P}(i', o')$. Hence, it is enough to define out as the time-abstraction of o'.

Note the importance of implicitly assuring strong guardedness and privacy preservation at the semantic level. Strong guardedness allows us to assume that input and output are properly sequenced in time without having to treat time explicitly in B. Privacy preservation allows us to assume that input and output respect the privacy requirements without having to handle them explicitly in B. This allows the specifier to concentrate on the characteristics of the application itself.

3.2.5. *Many-to-Many Composition*

The *parallel composition* of two mobile m2m components F_1 and F_2 is defined in terms of the operator for composition of static components.

Definition 13 (M2m composition). Given two mobile m2m components

$$F_1 \subseteq Comp_{m2m}(I_1, O_1, P_1), \quad F_2 \subseteq Comp_{m2m}(I_2, O_2, P_2)$$

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

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

We define the m2m composition of F_1 and F_2 as follows.

$$F_1 \oplus F_2 \equiv \{m2m_{I,O,P}(f) \mid f \in F_1 \odot F_2\}$$

In Appendix B (Theorem 7) we prove that $F_1 \oplus F_2$ belongs to $Comp_{m2m}(I, O, P)$. The functions $F_1 \odot F_2$ are additionally constrained by dmM and rnM in order to capture interconnection information, i.e., information local to $F_1 \oplus F_2$ but global to F_1 and F_2. For example, if F_1 outputs one of its passive ports $!c$ on a feedback channel and keeps $?c$ to itself, then both the environment and F_2 can output along $!c$, but only F_1 is allowed to input from $?c$. In that case, the output of F_2 along the port $!c$ should not be observable by the environment; this is ensured by rnM. Similarly, if F_1 outputs one of its passive input ports $?c$ on a feedback channel and keeps $!c$ to itself, then both the environment and F_2 can input on $?c$, but only F_1 is allowed to output along $!c$. In that case, the input of F_2 on $?c$ should contain messages sent only by F_1; this is ensured by dmM.

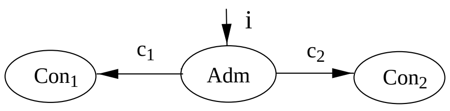

Fig. 3. Consultancy network.

3.2.6. Explicit Hiding

The privacy of a port not contained in the initial interface is guaranteed by privacy preservation. To hide ports in the initial interface, we use, similarly to the static case, an explicit *hiding operator*. If Q is a set of channel names of the mobile m2m component F, then $\nu Q.F$ is the m2m component obtained from F by adding Q to the initial set of passive channel names and deleting Q from the initial interface. The domain and range of the mobile m2m functions modelling $\nu Q.F$ are modified accordingly. As a consequence, only components receiving $\widetilde{p} \in ?!Q$ as an input message can communicate with F via the port p later on.

Definition 14 (Hiding). Given an m2m component $F \sqsubseteq Mob_{m2m}(I, O, P)$ and a set of channel names Q. Then $\nu Q.F$ is defined as below:

$$I' \equiv I \setminus Q, \quad O' \equiv O \setminus Q, \quad P' \equiv P \cup Q$$
$$\nu Q.F \equiv \{m2m_{I',O',P'}(f) \mid f \in F\}$$

In Appendix B (Theorem 8) we prove that $\nu Q.F$ belongs to $Comp_{m2m}(I', O', P')$. Note the role of dmM and rnM in maintaining privacy. If $p \in ?!Q$ is an input port then dmM makes sure that the behaviour of $\nu Q.F$ is independent of what the environment outputs along $\widetilde{p}$ before the environment has received $\widetilde{p}$. If $p \in ?!Q$ is an output port then rnM makes sure that $\nu Q.F$ does not output messages along p before it has sent $\widetilde{p}$ to its environment.

Example 2. Consultancy network:
In Example 1 we specified a consultant communicating with an administrator and a number of customers; we now specify the administrator and a consultancy network consisting of the administrator and two consultants. The consultancy network, whose initial configuration is illustrated graphically by Fig. 3, is described by a composite specification expressed in terms of elementary specifications, composition and hiding.

The consultancy network consists of the m2m composition of the administrator specified by Adm and two consultants described by two instances of Con. The initial input and output ports of each specification (instance) are renamed according to the input and output ports within the brackets to the left and right of $\rhd$, respectively. Renaming is positional and defines a new specification.

$$\nu c_1, c_2 : (Q \times !N).\ \text{Adm}(i \rhd c_1, c_2) \oplus (\text{Con}(c_1 \rhd) \oplus \text{Con}(c_2 \rhd))$$

Initially, the consultancy network has one external input port $?i$ on which it inputs questions from customers. Moreover, it has two local channels c_1 and c_2 on which the administrator distributes questions to the consultants; the set of external output ports is empty; the output ports are input during run-time via the input port $?i$.

The administrator is described by an elementary m2m specification as follows.

```
Adm                                                        m2m
  in    $i : (Q \times !N)$
  out   $c_1, c_2 : (Q \times !N)$
  ------------------------------------------------------------
  $\exists\, p \in \mathscr{P}(\{c_1, c_2\})^\infty : adm(p)(\text{in}) = \text{out}$
  where $\forall\, m \in Q \times !N;\ v \in H_A;\ p \in \mathscr{P}(\{c_1, c_2\})^\infty$ :
      $adm(p)(\{i \mapsto m\} \,\&\, v) = (\bigcup_{c \in \text{ft}.p} \{c \mapsto m\}) \,\&\, adm(\text{rt}.p)(v)$
```

For any non-empty stream s, the operators ft and rt are defined by $s = \langle \text{ft}.s \rangle \frown \text{rt}.s$. The existentially quantified variable p assigns a non-empty set of output ports to each question; this set identifies the set of consultants that will receive a copy of this particular question. Hence, p is used as an *oracle*. □

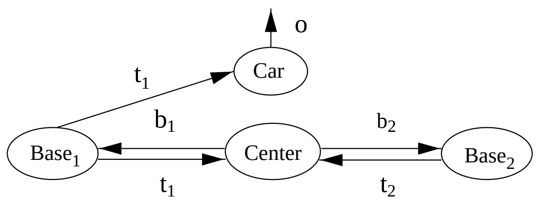

Fig. 4. Mobile telephone network — m2m version.

We do not need oracles to specify non-determinism, but the use of oracles is often convenient. As should be clear from the semantics of elementary specifications, the body of an elementary specification may be an arbitrary predicate. Hence, we can express non-determinism in the same way as non-determinism is expressed in traditional formal methods like VDM (pre/post-condition style) and Z. [StF98] demonstrates this kind of declarative specification style in a number of examples. In this paper, however, we have tried to write our specifications in an applicative style based on pattern matching since we believe an algorithmic specification style is more understandable for most specifiers. This requires the non-determinism to be filtered out with the help of oracles. In our opinion, by the use of oracles we get a very structured style of specification. Specifications can be understood as 'functional programs' based on pattern matching where the non-determinism is captured by oracles.

Example 3. Mobile telephones network — m2m version:
A centre is in permanent contact with two base stations, each in a different part of the country. A car with a mobile telephone moves about the country; it should always be in contact with a base. If it gets rather far from its current base contact, then a hand-over procedure is initiated, and as a result the car relinquishes contact with its current base and assumes contact with the other.

The mobile m2m format allows arbitrary sharing of both input and output channels. If we do not worry about interference, this is surely the most appropriate format; it often leads to very compact specifications. The system, whose initial configuration is illustrated by Fig. 4, is described by a composite specification as follows.

$$\nu t_1, t_2 : Talk \cup ?N \; ; \; b_1, b_2 : ?N \cup \{act\}.$$
$$(\mathrm{Centre}(t_1, t_2 \rhd b_1, b_2) \oplus \mathrm{Car}(t_1 \rhd o)) \oplus (\mathrm{Base}(b_1 \rhd t_1) \oplus \mathrm{Base}(b_2 \rhd t_2))$$

Initially, the car is in contact with the first base; between the car and the second base there is no direct link. For simplicity, we assume that the communication between the base stations and the car is unidirectional. The car forwards the information it inputs from the base stations to its environment via the channel o. The car can input either talk messages $m \in Talk \subseteq D$ or switch messages $?c \in ?N$. Any talk message is forwarded along o; the arrival of a switch message $?c$ forces the component to switch its input reading to $?c$.

Car — m2m

in $t_1 : Talk \cup ?N$
out $o : Talk$

$car(t_1)(\mathsf{in}) = \mathsf{out}$

where $\forall v \in H_A ; \; m \in Talk ; \; c, n \in N$:

$$\begin{aligned} car(n)(\{n \mapsto m\} \,\&\, v) &= \{o \mapsto m\} \;\&\; car(n)(v) \\ car(n)(\{n \mapsto ?c\} \,\&\, v) &= car(c)(v) \end{aligned}$$

An activated base may talk repeatedly with the car; it is activated by the receipt of the message act. If it receives an input port on its input channel, it may transmit this port to the car and itself become idle. Whether it ignores this input port or not is determined by the oracle p.

Base m2m

in $b : ?N \cup \{act\}$

out $t : Talk \cup ?N$

$\exists\, p \in \{1,2\}^{\infty};\ m \in Talk^{\infty} : idle(p,m)(\mathsf{in}) = \mathsf{out}$

where $\forall\, v \in H_A;\ p \in \{1,2\}^{\infty};\ m \in Talk^{\infty};\ c \in N\ :$

$$\begin{array}{lcll}
idle(p,m)(\{b \mapsto act\}\,\&\,v) & = & & act(p,m)(v) \\
act(1\,\&\,p,m)(v) & = & \{t \mapsto \mathsf{ft}.m\} & \&\ \ act(p,\mathsf{rt}.m)(v) \\
act(2\,\&\,p,m)(\{b \mapsto ?c\}\,\&\,v) & = & \{t \mapsto ?c\} & \&\ \ idle(p,m)(v)
\end{array}$$

The centre knows that the car is connected to the first base station, initially. During run-time it decides (according to information which we do not model) to transmit the input port $?t_2$ of the second base to the car via the first base. Subsequently, it inspects the communication on the channel t_1. When $?t_2$ has been forwarded to the car along t_1, it may activate the second base. Hence, t_1 also plays the role of an acknowledgment channel; it permits the centre to synchronise the activity of the two base stations.

Centre m2m

in $t_1, t_2 : Talk \cup ?N$

out $b_1, b_2 : ?N \cup \{act\}$

$left(\mathsf{in}) = \mathsf{out}$

where $\forall\, v \in H_A;\ m \in Talk\ :$

$$\begin{array}{lcll}
left(v) & = & \{b_1 \mapsto act,\ ?t_2\} & \&\ \ wait\_l(v) \\
wait\_l(\{t_1 \mapsto m\}\,\&\,v) & = & & wait\_l(v) \\
wait\_l(\{t_1 \mapsto ?t_2\}\,\&\,v) & = & & right(v) \\
right(v) & = & \{b_2 \mapsto act,\ ?t_1\} & \&\ \ wait\_r(v) \\
wait\_r(\{t_2 \mapsto m\}\,\&\,v) & = & & wait\_r(v) \\
wait\_r(\{t_2 \mapsto ?t_1\}\,\&\,v) & = & & left(v)
\end{array}$$

Note that despite the massive use of channel sharing, the above specification guarantees that no interference can occur on any of the channels involved. This is in accordance with the problem statement. However, the specification format itself does not impose this invariant. This is in contrast to the formats for p2p communication studied in the next section. □

4. Point-to-Point Communication

P2p communication differs from m2m communication in that different components are disallowed from outputting along the same channel within the same time unit.

4.1. Static Point-to-Point Networks

The set $SComp_{p2p}$ of static p2p components is identical to the set $SComp_{m2m}$ of static m2m components. Static p2p components interact only over a fixed set of communication channels with names in I and O.

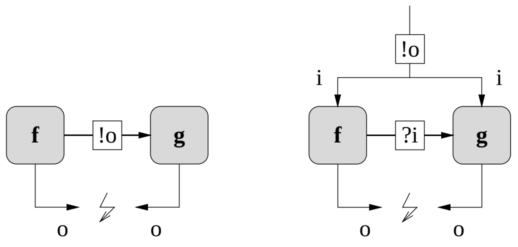

Fig. 5. Point-to-point invariant violation.

The difference compared to the m2m case is composition. In the p2p case channel interference is avoided by making sure that the sets of names for the input and output channels are pairwise disjoint.

Definition 15 (Static p2p composition). Given two static p2p components

$$F_1 \subseteq SComp_{p2p}(I_1, O_1), \qquad F_2 \subseteq SComp_{p2p}(I_2, O_2)$$

where $I_1 \cap I_2 = O_1 \cap O_2 = \{\}$. Let

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

We define the p2p composition of F_1 and F_2 as follows.

$$F_1 \ominus F_2 \equiv \{sm2m_{I,O}(f) \mid f \in F_1 \odot F_2\}$$

It follows straightforwardly from Theorem 6 that $F_1 \ominus F_2$ is a static p2p component in $SComp_{p2p}(I, O)$.

4.2. Mobile Point-to-Point Networks

The mobile p2p case is more subtle because it has to guarantee that different components do not output along the same channel within the same time unit even if the sets of input, output and passive ports known to a component vary in time. As mentioned in the introduction, we distinguish between p2p communication with and without channel sharing. We concentrate on the first variant in Sections 4.2.1 through 4.2.5; the second variant is treated in Section 4.2.6. To keep the presentation simple, we mainly work in an untyped setting without tuple messages; the exception is the semantics of specifications where we assume the model is extended in accordance with Section 3.2.3.

4.2.1. Point-to-Point Invariant

In the p2p case a network of components maintains the following invariant.

- At any given point in time, each port is known to at most one component.

This means that for any channel c, at any point in time, only two components may access c, namely the component that knows the input port and the component that knows the output port.

We ensure this p2p invariant by local requirements on the behaviour of the strongly guarded functions. To see the need for these requirements, consider once more the m2m case and assume that f outputs one of its *active* ports (say p) to another function g; then there are two ways in which the p2p invariant can be broken:

1. if p is an output port $!o$ as indicated by the network on the left-hand side of Fig. 5 then f and g may output simultaneously along $!o$;
2. if p is an input port $?i$ as indicated by the network on the right-hand side of Fig. 5 then both f and g may at some point in the future receive the same output port $!o$ on i and thereafter output simultaneously along $!o$.

Sending a *passive* port p is equally dangerous: f may at any point decide to activate p by outputting its complement $\widetilde{p}$. To eliminate the risk of channel interference we restrict a function to immediately forget any port it outputs along its output channels. Thus, with respect to our example, as soon as f forwards p, it may no longer take advantage of this port; this means that p is deleted from its set of active ports.

Note that a function may output the same port several times if it gains access to the same port several times. It may, however, not output the same port more than once for each time it gains access to it. For example, if a function f initially has access to a port p and f forwards this port, then f must postpone retransmitting it until it has regained access to p by receiving p via one of its input ports.

In the case of p2p communication, an active port p of a function f becomes passive as soon as f inputs its complement port $\widetilde{p}$. After all, if f has both ports to a channel, then only f knows about this channel. Consequently, both p and $\widetilde{p}$ should be added to the set of passive ports for f and p should be deleted from its set of active ports. Accordingly, if f receives both ports for a channel they are immediately included in its set of passive ports.

As in the m2m case, we are only interested in environments that stick to the rules of the game. We therefore constrain our functions to ignore the input messages that do not respect the privacy restrictions captured by the p2p invariant.

4.2.2. Formalising the Point-to-Point Invariant

We now explain how the p2p invariant described above is captured formally. We start by reformulating Definitions 9 and 10 for the p2p case.

Definition 16 (Active and passive ports). For $I, O, P \subseteq N$; $\theta, \delta \in H$; $t \in Nat_+$; let aP and pP be defined recursively as follows.

$$\begin{aligned} \mathsf{aP}_1 &\equiv ?I \cup !O, & \mathsf{pP}_1 &\equiv ?!P \\ \mathsf{aP}_{t+1} &\equiv (\mathsf{aP}_t \cup \mathsf{rP}_t \cup \mathsf{gP}_t) \setminus (\mathsf{sP}_t \cup \mathsf{hP}_t), & \mathsf{pP}_{t+1} &\equiv (\mathsf{pP}_t \cup \mathsf{hP}_t) \setminus (\mathsf{sP}_t \cup \widetilde{\mathsf{sP}_t}) \end{aligned}$$

where

$$\begin{aligned} \mathsf{rP}_t &\equiv \textstyle\bigcup_{?i \in \mathsf{aP}_t} \{p \mid p \in \overline{\mathsf{pP}_t \cup \mathsf{aP}_t} \cap \mathsf{pt}(\theta(i)(t))\}, & \mathsf{hP}_t &\equiv \{p, \widetilde{p} \mid p \in \mathsf{rP}_t \wedge \widetilde{p} \in (\mathsf{aP}_t \setminus \mathsf{sP}_t) \cup \mathsf{rP}_t\} \\ \mathsf{sP}_t &\equiv \textstyle\bigcup_{!i \in \mathsf{aP}_t} \{p \mid p \in (\mathsf{pP}_t \cup \mathsf{aP}_t) \cap \mathsf{pt}(\delta(i)(t))\}, & \mathsf{gP}_t &\equiv \{\widetilde{p} \mid p \in \mathsf{sP}_t \wedge p \in \mathsf{pP}_t\} \end{aligned}$$

Then the sets of active and passive ports at time t are characterised by

$$\mathsf{aP}_{I,O,P}(\theta, \delta)(t) \equiv \mathsf{aP}_t, \qquad \mathsf{pP}_{I,O,P}(\theta, \delta)(t) \equiv \mathsf{pP}_t$$

rP_t, sP_t, gP_t and hP_t are the sets of received, sent, generated and to-be-hidden ports, respectively. If the sets of active and passive ports are disjoint initially then they are also disjoint at any later point in time. Note that

$$\widetilde{\mathsf{pP}_t} = \mathsf{pP}_t, \qquad \mathsf{aP}_{t+1} \cup \mathsf{pP}_{t+1} = (\mathsf{aP}_t \cup \mathsf{pP}_t \cup \mathsf{rP}_t) \setminus \mathsf{sP}_t$$

Definition 17 (Domain and range). For any $I, O, P \subseteq N$; $\theta, \delta \in H$; $t \in Nat_+$; the domain and range at time t are characterised by

$$\mathsf{dmP}_{I,O,P}(\theta, \delta)(i)(t) \equiv \begin{cases} (\overline{\mathsf{pP}_t \cup \mathsf{aP}_t} \cup D) \circledS \theta(i)(t) & \text{if } ?i \in \mathsf{aP}_t \\ \langle\rangle & \text{otherwise} \end{cases}$$

$$\mathsf{rnP}_{I,O,P}(\theta, \delta)(i)(t) \equiv \begin{cases} (\mathsf{pP}_t \cup \mathsf{aP}_t \cup D) \circledS \delta(i)(t) & \text{if } !i \in \mathsf{aP}_t \\ \langle\rangle & \text{otherwise} \end{cases}$$

where $\mathsf{aP}_t \equiv \mathsf{aP}_{I,O,P}(\theta, \delta)(t)$ and $\mathsf{pP}_t \equiv \mathsf{pP}_{I,O,P}(\theta, \delta)(t)$.

Since a function can output the same port only once for each time it gains access to it, we consider only named communication histories $\theta \in H$ in which the same port does not occur twice in the same time unit for different channels. Such communication histories are *port unique*.

Definition 18 (Port uniqueness). A named communication history $\theta \in H$ is port unique iff

$$\forall\, t \in Nat_+;\ p \in ?!N;\ n, m \in N : p \in \mathsf{pt}(\theta(n)(t)) \wedge p \in \mathsf{pt}(\theta(m)(t)) \Rightarrow n = m$$

H_U is the set of all port unique communication histories in H. The merge component $\mathbb{M}$ preserves port uniqueness if its two arguments are without occurrences of the same port within the same time unit. More precisely, if

$$\mathsf{pts}(\theta, t) \equiv \{p \in ?!N \mid \exists\, n \in N : p \in \mathsf{pt}(\theta(n)(t))\}$$

we have

$$\forall\, \varphi, \psi \in H_U : (\forall\, t \in Nat_+ : \mathsf{pts}(\varphi, t) \cap \mathsf{pts}(\psi, t) = \{\}) \Rightarrow \forall\, m \in \mathbb{M} : m(\varphi, \psi) \in H_U$$

We can now characterise what it means for a function to be privacy preserving in the p2p case.

Definition 19 (Privacy preservation). A function $f \in H \to H$ is privacy preserving with respect to $I, O, P \subseteq N$ iff

$$\forall\, \theta \in H : f(\theta) = f(\mathsf{dmP}_{I,O,P}(\theta, f(\theta))) = \mathsf{rnP}_{I,O,P}(\theta, f(\theta))$$
$$\forall\, \theta \in H_U : f(\theta) \in H_U$$

Note that we defined the function f on H and not on H_U because we want any p2p function to be an m2m function. We use $Mob_{p2p}(I, O, P)$ to denote the set of all strongly guarded functions that are privacy preserving with respect to (I, O, P) in the p2p case. In the sequel we refer to such functions as *mobile p2p functions.*

As we said in the introduction, p2p communication can be understood as a particular case of m2m communication. Informally, a mobile p2p function is a mobile m2m function that preserves port uniqueness and forgets a port as soon as it is sent. In Appendix D (Theorem 18) we prove that this is indeed the case, i.e., that any mobile p2p function is also m2m.

As in the m2m case, in Appendix C (Theorem 11) we prove that any strongly guarded function $f \in H \to H$ that preserves port uniqueness can be transformed into a mobile p2p function $p2p_{I,O,P}(f) \in Mob_{p2p}(I, O, P)$ as follows.

$$p2p_{I,O,P}(f)(\theta) = \mathsf{rnP}_{I,O,P}(\theta, \delta) \text{ where } \delta = f(\mathsf{dmP}_{I,O,P}(\theta, \delta))$$

4.2.3. Mobile Point-to-Point Components

We model *mobile p2p components* by sets of mobile p2p functions.

Definition 20 (Mobile p2p component). A mobile p2p component with initial interface (I, O) and initial set of passive channel names P is represented by a non-empty set of mobile p2p functions $F \subseteq Mob_{p2p}(I, O, P)$.

We use $Comp_{p2p}(I, O, P)$ to denote the set of all mobile p2p components with respect to (I, O, P).

4.2.4. Mobile Point-to-Point Composition

Mobile p2p composition is defined similarly to the static case. However, feedback channels are in this case hidden both statically and dynamically.

Definition 21 (Mobile p2p composition). Given two mobile p2p components

$$F_1 \subseteq Comp_{p2p}(I_1, O_1, P_1), \qquad F_2 \subseteq Comp_{p2p}(I_2, O_2, P_2)$$

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

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

We define the p2p composition of F_1 and F_2 as follows.

$$F_1 \otimes F_2 \equiv \{p2p_{I,O,P}(f) \mid f \in F_1 \odot F_2\}$$

In Appendix C (Theorem 12) we prove that $F_1 \otimes F_2$ belongs to $Comp_{p2p}(I, O, P)$. As in the m2m case, the restriction of f with dmP and rnP is necessary to capture interconnection information, i.e., information local to $F_1 \otimes F_2$ but global to F_1 and F_2. For example, if p is an active port of F_1 and $\widetilde{p}$ is an active port of F_2 then the pair $\{p, \widetilde{p}\}$ is private to $F_1 \otimes F_2$. However, neither F_1 nor F_2 can be aware of this fact.

Note that contrary to the m2m operator $\oplus$, the p2p operator $\otimes$ hides ports in the initial interface: those channels that belong to the initial interface of both components are automatically hidden (see the definition of I, O and P); an additional hiding operator is, therefore, not needed. Hence, in the p2p case we do not need a hiding operator of the kind introduced for m2m communication since at any given point in time each port is known to at most one component.

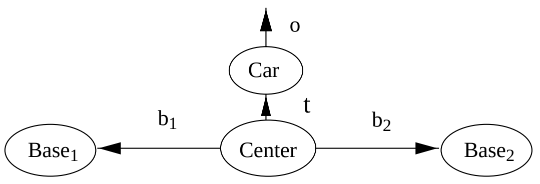

Fig. 6. Mobile telephone network — p2p version.

4.2.5. *Point-to-Point Specifications*

The p2p model is just a special case of the mobile m2m model. Hence, we may still use the specification formats for m2m communication. This requires, however, the specifiers to explicitly capture the hiding invariants for p2p communication; this results in unnecessarily complex specifications. Specification formats specially tuned towards p2p communication are therefore desirable.

Syntactically, elementary p2p specifications differ from elementary m2m specifications in only one respect: the label m2m is replaced by p2p. The semantics of an elementary p2p specification S with initial interface (I, O), initial set of passive channel names P_s and body B is defined as follows.

$$[\![S]\!] \equiv \{ \quad g \in Mob_{p2p}(I, O, P_s) \ \mid \ \forall i' \in H_{TU} : \exists o' \in H_{TU} : o' = g(i') \wedge B(\mathsf{in}, \mathsf{out}) \\ \text{where } \mathsf{in} = \mathsf{ta}(\mathsf{dmP}_{I,O,P}(i', o')),\ \mathsf{out} = \mathsf{ta}(o') \quad \}$$

By H_{TU} we denote the type-correct subset of H_U. H_{AU} is the corresponding set of untimed typed communication histories. Note the role of the time-abstraction operator in the definition of in and out.

Example 4. Mobile telephone — p2p version:
The mobile p2p model constrains a component to forget a port p as soon as it is sent; the component regains access to p if p is later input via one of its input ports. In the specification of the mobile telephone network considered in this example, we make strong use of this feature. The specification demonstrates switching as a process of gaining and losing access to an output port. This network, whose initial configuration is illustrated by Fig. 6, is specified by the following composite specification.

$$(\text{Centre}(\rhd b_1, b_2, t) \otimes \text{Car}(t \rhd o)) \otimes (\text{Base}(b_1 \rhd) \otimes \text{Base}(b_2 \rhd))$$

Initially there is no direct or indirect communication link from the base stations to the car. The centre is connected to the car via the channel t. The centre itself does not communicate via t: during run-time it transmits the port $!t$ to and from the two base stations via the channels b_1 and b_2.

The specification of the car is very simple: the external interface does not change and the input from t is just forwarded along o with an arbitrary delay. Formally,

```
══ Car ═══════════════════════════════════════ p2p ══
│ in    t : Talk
│ out   o : Talk
├─────────────────────────────────────────────────────
│ out(o) = in(t)
└─────────────────────────────────────────────────────
```

A base station is initially idle; it remains idle until it inputs an output port $!k$ on its input port $?b$; then it communicates via $!k$ until it inputs a second output port $!l$ on $?b$. The base station responds to the second output port by halting the communication on $!k$ and sending both output ports back along $!l$. Thereafter it remains idle until the whole procedure is restarted by the receipt of another output port on $?b$. Note that the amount of talking is underspecified by the oracle p (& is redefined for streams in the obvious manner).

Base p2p

in $b : !N$

$\exists\, p \in \{1,2\}^{\infty};\ m \in \mathit{Talk}^{\infty} : \mathit{idle}(p,m)(\mathsf{in}) = \mathsf{out}$

where $\forall\, k,l \in N;\ v \in H_{AU};\ p \in \{1,2\}^{\infty};\ m \in \mathit{Talk}^{\infty}$:

$$\begin{array}{lcl}
\mathit{idle}(p,m)(\{b \mapsto !k\} \,\&\, v) & = & \mathit{act}(k)(p,m)(v) \\
\mathit{act}(k)(1 \,\&\, p,m)(v) & = & \{k \mapsto \mathsf{ft}.m\} \ \&\ \mathit{act}(k)(p,\mathsf{rt}.m)(v) \\
\mathit{act}(k)(2 \,\&\, p,m)(\{b \mapsto !l\} \,\&\, v) & = & \{l \mapsto !k, !l\} \ \&\ \mathit{idle}(p,m)(v)
\end{array}$$

Finally, we specify the centre: as already mentioned, it manages the transmission of t to and from the two base stations.

Centre p2p

out $b_1, b_2 : !N;\ t : \mathit{Talk}$

$\exists\, q \in \mathsf{priv} : \mathit{left}(q)(\mathsf{in}) = \mathsf{out}$

where $\forall\, v \in H_{AU}$:

$$\begin{array}{lcl}
\mathit{left}(v) & = & \{b_1 \mapsto !t, !q\} \ \&\ \mathit{wait\_l}(v) \\
\mathit{wait\_l}(\{q \mapsto !t, !q\} \,\&\, v) & = & \mathit{right}(v) \\
\mathit{right}(v) & = & \{b_2 \mapsto !t, !q\} \ \&\ \mathit{wait\_r}(v) \\
\mathit{wait\_r}(\{q \mapsto !t, !q\} \,\&\, v) & = & \mathit{left}(v)
\end{array}$$

Both $!t$ and $!q$ are used repeatedly by the base stations, i.e., they are shared. They are, however, never used simultaneously; each time a base station returns $!t$ and $!q$ back to the centre it loses access to these ports. By sending the private port $!q \in \mathsf{priv}$ (remember that priv denotes the initial set of passive ports), the centre automatically gets access to the port $?q$. By receiving the port $!q$ back, the centre has access to both $?q$ and $!q$, i.e., q becomes passive once more. □

4.2.6. Restrictive Point-to-Point Communication

So far we have introduced two specification formats: one for m2m communication and one for p2p communication. Of course, we may also define formats specially tuned towards other communication paradigms. In this section we strengthen the privacy invariant for p2p communication to disallow channel sharing. Let $\theta\dagger_n$ denote the history obtained from θ by hiding the information sent along the channel n, i.e.,

$$\theta \dagger_n (m) \;\equiv\; \begin{cases} \langle\rangle^{\infty} & \text{if } n = m \\ \theta(m) & \text{otherwise} \end{cases}$$

Restrictive p2p communication guarantees that forwarded ports are not used for communication purposes by the forwarding component. The privacy invariant of mobile p2p functions guarantees that ports are not used after they have been forwarded. Hence, it is enough to restrict the behaviour until a port is forwarded.

Definition 22 (Restrictive p2p component). A p2p component F is a restrictive p2p component if for all $f \in F;\ n,o \in N;\ t \in \mathit{Nat}_{+};\ \theta \in H$:

$$?n \in \mathsf{pt}(f(\theta)(o)(t)) \Rightarrow f(\theta){\downarrow_t} = f(\theta\dagger_n){\downarrow_t}, \qquad !n \in \mathsf{pt}(f(\theta)(o)(t)) \Rightarrow f(\theta){\downarrow_t} = (f(\theta)\dagger_n){\downarrow_t}$$

Consequently, channel sharing is no longer possible; for example, this excludes the shared use of the channels t and q in Example 4. The set of restrictive p2p components with respect to (I,O,P) is denoted by

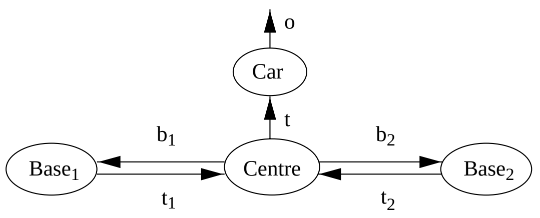

Fig. 7. Mobile telephone network — restrictive p2p version.

$Comp_{r\_p2p}(I, O, P)$. In Appendix C (Theorem 13) we prove that the p2p composition of two restrictive p2p components yields a restrictive p2p component.

The specification formats for p2p communication are redefined for restrictive p2p communication in the obvious manner. To demonstrate the potential of having additional specification formats, we once more specify a variant of the mobile telephone network.

Example 5. Mobile telephones — restrictive p2p version:
Contrary to earlier, the centre employs only new channels to connect the base stations to the car: at each communication switch, both the car and the activated base station receives a port to a completely new channel. The network, whose initial configuration is illustrated by Fig. 7, is specified as follows.

$$(\mathrm{Centre}(t_1, t_2 \rhd b_1, b_2, t) \otimes \mathrm{Car}(t \rhd o)) \otimes (\mathrm{Base}(b_1 \rhd t_1) \otimes \mathrm{Base}(b_2 \rhd t_2))$$

The specification of the car is identical to that of Example 3 with the exception that its label is replaced by r_p2p. The specification of the centre is similar to the m2m version. However, in this case, for each new channel the centre must take care to send the input port to the car and the output port to the corresponding base.

Centre r_p2p

in $t_1, t_2 : \{\mathsf{ok}\}$

out $b_1, b_2 : ?!N$; $t : Talk \cup ?N$

$\exists\, p \in \mathsf{priv}^{\infty} : left(t \,\&\, p)(\mathsf{in}) = \mathsf{out}$

where $\forall\, v \in H_{AU}$; $n \in \mathsf{priv}$; $p \in \mathsf{priv}^{\infty}$:

$$\begin{array}{lcll}
left(n \,\&\, p)(v) & = & \{b_1 \mapsto !n,\ ?\mathsf{ft}.p\} & \&\ \ wait\_l(p)(v) \\
wait\_l(p)(\{t_1 \mapsto \mathsf{ok}\} \,\&\, v) & = & & right(p)(v) \\
right(n \,\&\, p)(v) & = & \{b_2 \mapsto !n,\ ?\mathsf{ft}.p\} & \&\ \ wait\_r(p)(v) \\
wait\_r(p)(\{t_2 \mapsto \mathsf{ok}\} \,\&\, v) & = & & left(p)(v)
\end{array}$$

Note that the r_p2p constraint enforces (as desired) that all ports generated from p are distinct. Hence, we do not have to write down this requirement.

The specification of the base differs from the m2m version in that the base receives the new output channel instead of *act* and that the forwarding of the output port is signalled by an ok.

Base r_p2p

in $b : ?!N$

out $t : \{\mathsf{ok}\}$

$\exists\, p \in \{1,2\}^{\infty};\ m \in \mathit{Talk}^{\infty} : \mathit{idle}(p,m)(\mathsf{in}) = \mathsf{out}$

where $\forall\, v \in H_{AU}\,;\ c,e \in N\,;\ p \in \{1,2\}^{\infty};\ m \in \mathit{Talk}^{\infty}$:

$$
\begin{array}{lcll}
\mathit{idle}(p,m)(\{b \mapsto !e\} \,\&\, v) & = & & \mathit{act}(e)(p,m)(v) \\
\mathit{act}(e)(1 \,\&\, p,m)(v) & = & \{e \mapsto \mathsf{ft}.m\} & \&\ \mathit{act}(e)(p,\mathsf{rt}.m)(v) \\
\mathit{act}(e)(2 \,\&\, p,m)(\{b \mapsto ?c\} \,\&\, v) & = & \{e \mapsto ?c,\ t \mapsto \mathsf{ok}\} & \&\ \mathit{idle}(p,m)(v)
\end{array}
$$

□

5. Discussion

In this paper we have defined a denotational model for mobile systems, i.e., for systems in which every component may change its communication partners on the basis of computation and interaction. This model allows a more profound understanding of mobility as a particular privacy invariant that is maintained by the mobile system. We analysed privacy with respect to three communication paradigms: many-to-many communication (m2m), point-to-point communication with channel sharing (p2p) and point-to-point communication without channel sharing (r_p2p).

For each of these paradigms we defined a specification format that supports the maintenance of the associated invariant by imposing it implicitly via the semantic mapping. By relieving the specifier from the burden of stating the invariants explicitly in each specification these formats allow 'high level' specifications of mobile systems. The models and their associated specification formats were defined in a stepwise manner from the most liberal m2m model to the most restrictive r_p2p model. We showed that each of them is obtained from the previous one by strengthening its privacy invariant.

Based on this, when should which format be used? The general rule is to use the most restrictive format that fits the communication paradigm of the component you consider. Hence, if you have a component based on restrictive p2p communication then you may in principle use all three formats, but the format for restrictive p2p communication is recommended since it imposes all the required communication invariants implicitly via the semantics — invariants that otherwise have to be specified explicitly if any of the two other formats are used.

We also believe there are situations where p2p communication could be useful to give an abstract view of m2m systems. The reason is quite simply the very controlled form of interference guaranteed by the p2p formats, which has a simplifying effect on formal reasoning. This requires, however, flexible refinement paradigms allowing specifications based on p2p communication to be refined into specifications based on m2m communication. We believe such refinement paradigms can be formulated, but this is an issue of further research.

Since our specification formats suppress the guardedness and privacy constraints by imposing them implicitly via the semantics, an interesting question is to what extent reasoning is possible without unwinding the definitions. The answer is 'to a large extent'. For example, with respect to guardedness and time abstraction this issue is investigated in [BrS94], which proposes a rule for feedback that is complete in a certain sense with respect to the kind of components that can be fully characterised by relations on untimed streams, which is a large and in practice important class of untimed dataflow components. In the timed case, we may prove many properties — but, of course, not all — without referring to guardedness. When the body of the specification itself is not sufficiently strong to deduce a property depending on guardedness we may use an adaptation rule to strengthen the specification with guardedness. The need for adaptation rules is well known from other kinds of proof calculi for computer programs (see for instance [Hoa71, LGH78, GrL80]).

In the case of privacy the situation is as for guardedness: many proofs can be carried out without additional privacy information. In those cases where this is not possible, we use an adaptation rule to make the specification sufficiently strong. By using adaptation rules in this sense we are able to keep the specifications simple and at the same time obtain full proof power when this is required.

This paper considers neither refinement nor program verification. Reasoning in the context of streams is, however, well documented in the literature [Ste97]. [Stø96], which gives a complete solution to the RPC-Memory Specification Problem, demonstrates reasoning in the context of an oracle-based specification style. The reasoning techniques of [Stø96] can be adapted to the various specification formats introduced above in the form of specialised deduction rules. The actual formulation of such rules that are both simple to use and sufficiently powerful is a matter of further research.

The exact relationship between our model and more operational models for mobile systems like, for instance, the π-calculus [Mil91] and the actor-based approaches [AMS92] is an interesting area for future research. For example, we believe that our model can be used to give a denotational semantics for the asynchronous π-calculus. We also believe that the actor languages can be smoothly integrated within our formalism.

Our approach is related to the work of Kok [Kok87, Kok89]. The major difference is that Kok does not deal with mobility. Moreover, the handling of non-determinism differs from ours. In [Kok89], where a metric on relations is used, basically only bounded non-determinism can be handled. In [Kok87], which is not based on metric spaces, an automaton is used to generate the behaviour of basic agents. This guarantees the existence of fix-points. We use sets of strongly guarded functions for the same purpose. Another important difference with respect to [Kok87] is that we do not consider time abstraction (at the semantic level). The reasons are quite simple. First, we want to model reactive systems and for such systems time plays an important role. Second, in an untimed input/output model one cannot define and understand the privacy invariant.

Our main contribution is that we have formalised mobility in the context of streams and functions on streams. In this respect our work is in the tradition of Kahn [Kah74, KaM77]. The close relationship between stream-based models and models based on traces[1] as, for example, advocated by Jonsson [Jon89], is well known. Hence, it makes sense to ask why we use a stream-based model and not one based on traces. Well, we think that each model has its own merits and it is hard to classify one as better than the other. Each model was pursued in different schools and each school prefers its own model. This might also be a matter of tradition. We are not aware of work on mobility within the framework of [Jon89]. How our approach to mobility should be reformulated in a trace-based setting and whether this reformulated trace-based version would be simpler or more elegant than the one presented in this paper, is a matter of further research.

We think that the trace-based automata-like models are closer to *logic*, whereas the stream-based functional models are closer to *engineering*. The second class of models promotes thinking in terms of blocks, sequential composition, parallel composition and feedback — concepts long used and validated in other engineering disciplines (e.g., control theory). Moreover, it promotes the use of paradigms already developed in the (sequential) programming languages community, like subtyping, parametric polymorphism, higher-order types, etc. In fact, these days there is intensive work on the unification of these two approaches. Among these efforts, probably the best known is the work of Samson Abramski on interaction categories [Abr96].

Mathematical modelling involves abstraction and abstraction means leaving something out. As highlighted by the Greek philosopher Zenon more than 2400 years ago, when something is left out we may get some strange effects — often referred to as anomalies. Two very famous anomalies known from stream-based models for concurrent systems is the merge anomaly [Kel78] and the Brock/Ackermann anomaly [BrA81]. They can be summarised as follows.

- A fair merge component merging two untimed streams into a third untimed stream containing all the messages of the argument streams cannot be represented by a prefix-monotonic function.
- Relations on untimed streams are not sufficiently expressive to distinguish all observationally different dataflow components.

As explained carefully in [BrS94], the fair merge anomaly occurs because in the case of untimed streams there is no way to distinguish a finite incomplete input history with exactly n messages from a complete input history consisting of exactly the same sequence of messages. In our model, we consider only complete communication histories, which means that this problem cannot occur even in the case of time-abstraction.

On the other hand, our specification formats seem to suffer from the Brock/Ackermann anomaly because

[1] A stream is a sequence representing the communication history of a channel; a trace is a sequence representing the communication history of a component run; in a trace the streams of the input/output channels are interleaved into a single history.

they are all based on time-abstraction, which means that a specification is basically a relation on untimed streams. However, this is only seemingly so, because the underlying model is timed and we can of course define similar formats without time-abstraction that are well known to be sufficiently expressive (see [BrS94]). Hence, we use a format with time-abstraction when we describe a component for which the untimed format is sufficiently expressive; otherwise, we use a timed format. The timed format is of course in any case needed to specify components with time constraints. We have not considered timed specification formats in this paper, because it is not required for the examples we consider. [Stø96, StF98, BrS01] are all based on the idea originally suggested in [BrS94] of distinguishing between timed and untimed formats for the specification of dataflow networks. Note that although the Brock/Ackermann anomaly is an interesting theoretical challenge, it is seldom a problem in practice. The fact that the merge nodes used in the definition of composition are undelayed has no impact in the context of Brock/Ackermann since they are weakly guarded and used in such a way that the resulting network is guaranteed to be strongly guarded.[2]

[Gro94] defines a stream-based semantic model for mobile deterministic dataflow networks. This model is, however, higher-order and mobility is achieved by communicating channels and functions instead of ports. [Bro95, Gro94] give also an equational characterisation of dynamic reconfiguration. Mobility in the more general framework of non-deterministic systems where reconfiguration is achieved by sending ports is studied in [GrS95, GrS96a, GrS96b, GSB97] and this paper unifies and summarises this work. Our formalism has been applied successfully to give a formal high-level specification of the kernel functionality of an operating system [HiS96, Spi98]. In this specification, mobility represents resource allocation and recursion represents process creation. The m2m model was also successfully used in [Hin98] to give a formal semantics for the object-oriented extension of the ITU standardised specification and description language SDL [ITU93]. [GrS97, Stø99] study m2m mobility in a purely relational setting. [Stø99] defines the semantic mapping without recursive characterisations of domain and range.

Acknowledgements

We thank Manfred Broy for stimulating discussions and valuable feedback. Thanks go also to Ursula Hinkel, Ingolf Krüeger, Jan Philipps and Katharina Spies for reading an early draft and providing useful comments. The suggestions and remarks made by the anonymous referees were also very helpful.

Ketil Stølen's contribution to this paper has partly been sponsored by his part-time employer, the Department of Informatics at the Univerisity of Oslo.

References

[Abr96] Abramski, S.: Retracing Some Paths in Process Algebra. In *Proceedings of the International Conference on Concurrency Theory*. LNCS 1055, Springer, Berlin, 1996, pp. 21–33.

[AMS92] Agha, G., Mason, I. A. Smith, S. F. and Talcott, C. L.: Towards a theory of actor computation. In *Proceedings of the International Conference on Concurrency Theory*, LNCS 630, Springer, Berlin, 1992, pp. 565–579.

[BrA81] Brock, J. D. and Ackermann, W. B.: Scenarios: A model of non-determinate computation. In *Proceedings of the Formalisation of Programming Concepts*, LNCS 107, Springer, Berlin, 1981, pp. 252–259.

[BeB90] Berry, G. and Boudol, G.: The chemical abstract machine. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, San Francisco, 7th Annual ACM Symp., ACM Press, 1990, pp. 81–94.

[BDD93] Broy, M., Dederichs, F., Dendorfer, C., Fuchs, M., Gritzner, T. F. and Weber, R.: The design of distributed systems: an introduction to Focus (revised version). *Technical Report SFB 342/2/92 A*, Technische Universität München, 1993.

[Bro95] Broy, M.: Equations for describing dynamic nets of communicating systems. In *Proceedings of the COMPASS Workshop*, LNCS 906, Springer, Berlin, 1995, pp. 170–187.

[BrS94] Broy, M. and Stølen, K.: Specification and refinement of finite dataflow networks: a relational approach. In *Proceedings of the International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. LNCS 863, Springer, Berlin, 1994, pp. 247–267.

[BrS01] Broy, M. and Stølen, K.: *Specification and Development of Interactive Systems: FOCUS on Streams, Interfaces and Refinement*. Springer, Berlin, 2001.

[EnN86] Engberg, U. and Nielsen, M.: A calculus of communicating systems with label-passing. *Technical Report DAIMI PB-208*, University of Aarhus, 1986.

[Eng89] Engelking, R.: *General Topology*. volume 6 of Sigma Series in Pure Mathematics, Heldermann Verlag, Berlin, revised and completed edition, 1989.

[2] Remember that the functional composition of a strongly and weakly guarded function yields a strongly guarded function [Eng89].

[FMS96] Fiore, M. P., Moggi, E. and Sangiorgi, D.: A fully-abstract model for the pi-calculus (extended abstract). In *Proc. of the 11th Annual IEEE Symposium on Logic in Computer Science*, New Brunswick, New Jersey, 27–30 July 1996, IEEE Computer Society Press.

[GrL80] Gries, D. and Levin, G. M.: Assignment and procedure call proof rules. *ACM Transactions on Programming Languages and Systems*, 2:564–579, 1980.

[Gro94] Grosu, R.: *A Formal Foundation for Concurrent Object Oriented Programming*. PhD thesis, *Technical Report TUM-I9444*, Technische Universität München, 1994.

[GrS95] Grosu, R. and Stølen, K.: Selected Papers from 8th Nordic Workshop on Programming Theory, NWPT'96, Oslo, Norway, 4–6 Dec. 1996, Research Report 248, Dept. of Informatics, Univ. of Oslo, 1997, pp. 67–76.

[GrS96a] Grosu, R. and Stølen, K.: A denotational model for mobile many-to-many dataflow networks. *Technical Report TUM-I9622*, Technische Universität München, 1996.

[GrS96b] Grosu, R. and Stølen, K.: A model for mobile point-to-point dataflow networks without channel sharing. In *Proceedings of the Conference on Algebraic Methodology and Software Technology*, LNCS 1101, Springer, Berlin, 1996, pp. 504–519.

[GrS97] Grosu, R. and Stølen, K.: Specification of Dynamic Networks. In *Proceedings of the Nordic Workshop on Programming Theory*, 1997, pp. 67–76.

[GSB97] Grosu, R., Stølen, K. and Broy, M.: A denotational model for mobile point-to-point dataflow networks with channel sharing. *Technical Report TUM-I9717*, Technische Universität München, 1997.

[HBS73] Hewitt, C., Bishop, P. and Steiger, R.: A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, Stanford, CA, August 1973, William Kaufmann, Nilsson, N. J. editor, pp. 235–245.

[Hin98] Hinkel, U.: Verification of SDL specifications on base of a stream semantics. In *Proceedings of the Workshop of the SDL Forum Society on SDL and MSC*, 1998, pp. 241–250.

[Hoa71] Hoare, C. A. R.: Procedures and parameters: an axiomatic approach. In *Symposium on Semantics of Algorithmic Languages*, E. Engeler (editor), Lecture Notes in Mathematics, vol. 188, Springer Verlag, 1971, pp. 102–116.

[HiS96] Hinkel, U. and Spies, K.: Anleitung zur Spezifikation von mobilen, dynamischen FOCUS-Netzen. *Technical Report TUM-I9639*, Technische Universität München, 1996.

[ITU93] International Telecommunication Union, Geneva. *Recommendation Z.100: Functional Specification and Description Language (SDL)*, 1993.

[JaJ95] Jagadeesan, L. J. and Jagadeesan, R.: Causality and true concurrency: A dataflow analysis of the pi-calculus. In *Proceedings of the Conference on Algebraic Methodology and Software Technology*, LNCS 936, 1995, pp. 277–291.

[Jon90] Jones, C. B.: *Systematic Software Development Using VDM*, (2nd edn). Prentice-Hall, Englewood Cliffs, NJ, 1990.

[Jon89] Jonsson, B.: A fully abstract trace model for dataflow networks. In *Proceedings of the ACM Symposium on Principles of Programming Languages*, 1989, pp. 155–165.

[Kah74] Kahn, G.: The semantics of a simple language for parallel programming. Jack L. Rosenfeld (Ed.), Information Processing 74, In *Proceedings of IFIP Congress 74*, Stockholm, Sweden, August 5–10, 1974, North Holland, 1974.

[Kel78] Keller, R. M.: Denotational models for parallel programs with indeterminate operators. In *Proceedings of the IFIP Working Conference on Formal Description of Programming Concepts*, 1978, pp. 337–363.

[KaM77] Kahn, G. and MacQueen, B. D.: Coroutines and networks of parallel processes, In *Proceedings of the IFIP Congress*, 1977, pp. 993–998.

[Kok87] Kok, J. N.: A fully abstract semantics for data flow nets. In *Proceedings of the Conference on Parallel Architectures and Languages Europe*, LNCS 259, Springer, Berlin, 1987, pp. 351–368.

[Kok89] Kok, J. N.: An iterative metric fully abstract semantics for non-deterministic dataflow. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science*, LNCS 379, Springer, Berlin, 1989, pp. 321–331.

[LGH78] London, R. L., Guttag, J. V., Horning, J. J., Lampson, B. W., Mitchell, J. G. and Popek, G. J.: Proof rules for the programming language Euclid. *Acta Informatica*, 1978, 10:1–26.

[Mes91] Meseguer, J.: Conditional rewriting logic as a model of concurrency. *Theoretical Computer Science*, 96(1), 73–155 6 April 1992.

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

[MPW92] Milner, R., Parrow,J. and Walker, D.: A calculus of mobile processes, parts I and II. *Information and Computation*, 1992, 100:1–77.

[StF98] Stølen, K. and Fuchs, M.: An exercise in conditional refinement. In *Prospects for Hardware Foundations*, LNCS 1546, 1998, pp. 390–420.

[Spi88] Spivey, J. M.: *Understanding Z, a Specification Language and its Formal Semantics*, Vol. 3 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1988.

[Spi98] Spies, K.: *Eine Methode zur Formalen Modellierung von Betriebsystemkonzepten*. PhD thesis, Technische Universität München, 1998.

[Sta96] Stark, I.: A fully abstract domain model for the π-calculus. In *Proceedings of the IEEE Symposium on Logic in Computer Science*, 1996, pp. 36–42.

[Ste97] Stephens, R.: A survey of stream processing. *Acta Informatica*, 1997, 34:491–541.

[Stø96] Stølen, K.: Using relations on streams to solve the RPC-memory specification problem. In *Formal Systems Specification: The RPC-Memory Specification Case Study*, LNCS 1169, Springer, Berlin, 1996, pp. 477–520.

[Stø99] Stølen, K.: Specification of Dynamic Reconfiguration in the Context of Input/Output Relations. In *Proceedings of the Formal Methods for Open Object-Based Distributed Systems*, 1999, pp. 259–272.

[Tho89] Thomsen, B.: A calculus of higher order communicating systems. In *16th Annual Symposium on Principles of Programming Languages*, Austin, Texas, January 1989, ACM Press.

A. Metrics on Streams and Named Stream Tuples

Definition 23 (Metric of streams). The *metric of streams* (E^∞, d) is defined as follows.

$$E^\infty \equiv \times_{t \in Nat} E, \qquad d(r,s) \equiv inf\{2^{-t} \mid r{\downarrow_t} = s{\downarrow_t}\}$$

This metric is also known as the Baire metric [Eng89].

Theorem 1. The metric space of streams (E^∞, d) is complete.

Proof. See [Eng89]. □

Definition 24 (Metric of named stream tuples). The *metric of named stream tuples* $(I \to E^\infty, d)$ over a countable set of names I is defined as follows.

$$d(\theta, \varphi) \equiv inf\{2^{-t} \mid \theta{\downarrow_t} = \varphi{\downarrow_t}\}$$

Theorem 2. The metric space of named stream tuples $(I \to E^\infty, d)$ is complete.

Proof. The metric of named stream tuples is equivalent to the Cartesian product metric $\times_{i \in I} E^\infty$, which is complete because E^∞ is complete (see [Eng89]). □

B. Proofs: Mobile M2m Case

Theorem 3. The functions pM and aM are strongly guarded and the functions dmM and rnM are weakly guarded.

Proof. $\mathsf{pM}_{I,O,P}(\theta,\delta)(t)$ and $\mathsf{aM}_{I,O,P}(\theta,\delta)(t)$ depend only on $\theta \downarrow_{t-1}$ and $\delta \downarrow_{t-1}$. $\mathsf{dmM}_{I,O,P}(\theta,\delta)(i)(t)$ and $\mathsf{rnM}_{I,O,P}(\theta,\delta)(i)(t)$ depend only on $\theta{\downarrow_t}$ and $\delta{\downarrow_t}$. □

Theorem 4. The functions dmM and rnM satisfy the following properties.

$$\begin{aligned}
\mathsf{dmM}_{I,O,P}(\theta,\delta) &= \mathsf{dmM}_{I,O,P}(\mathsf{dmM}_{I,O,P}(\theta,\delta),\delta) \\
&= \mathsf{dmM}_{I,O,P}(\theta,\mathsf{rnM}_{I,O,P}(\theta,\delta)) \\
\mathsf{rnM}_{I,O,P}(\theta,\delta) &= \mathsf{rnM}_{I,O,P}(\mathsf{dmM}_{I,O,P}(\theta,\delta),\delta) \\
&= \mathsf{rnM}_{I,O,P}(\theta,\mathsf{rnM}_{I,O,P}(\theta,\delta))
\end{aligned}$$

Proof. The proof is based on the inductive definitions of aM and pM. To save space we drop the I, O, P subscripting. We do similar simplifications in later proofs.

Induction hypothesis:

$$\begin{aligned}
\mathsf{aM}(\theta,\delta)(n) &= \mathsf{aM}(\mathsf{dmM}(\theta,\delta),\delta)(n) = \mathsf{aM}(\theta,\mathsf{rnM}(\theta,\delta))(n) \\
\mathsf{pM}(\theta,\delta)(n) &= \mathsf{pM}(\mathsf{dmM}(\theta,\delta),\delta)(n) = \mathsf{pM}(\theta,\mathsf{rnM}(\theta,\delta))(n)
\end{aligned}$$

To simplify the notation we define:

$$\begin{aligned}
\mathsf{aM}_n &\equiv \mathsf{aM}(\theta,\delta)(n), \quad \mathsf{aM}'_n \equiv \mathsf{aM}(\mathsf{dmM}(\theta,\delta),\delta)(n), \quad \mathsf{aM}''_n \equiv \mathsf{aM}(\theta,\mathsf{rnM}(\theta,\delta))(n) \\
\mathsf{pM}_n &\equiv \mathsf{pM}(\theta,\delta)(n), \quad \mathsf{pM}'_n \equiv \mathsf{pM}(\mathsf{dmM}(\theta,\delta),\delta)(n), \quad \mathsf{pM}''_n \equiv \mathsf{pM}(\theta,\mathsf{rnM}(\theta,\delta))(n)
\end{aligned}$$

Base Case: $\mathsf{aM}_1 = \mathsf{aM}'_1 = \mathsf{aM}''_1 = ?I \cup !O$ and $\mathsf{pM}_1 = \mathsf{pM}'_1 = \mathsf{pM}''_1 = ?!P$.

Induction Step: By the induction hypothesis $\mathsf{aM}_n = \mathsf{aM}'_n = \mathsf{aM}''_n$ and $\mathsf{pM}_n = \mathsf{pM}'_n = \mathsf{pM}''_n$. By the definition of aM and pM:

$$\begin{array}{lcll}
\mathsf{aM}_{n+1} & = & \mathsf{aM}_n & \cup \; \bigcup_{?i \in \mathsf{aM}_n} \{c \mid c \in \overline{\mathsf{aM}_n \cup \mathsf{pM}_n} \wedge c \in \mathsf{pt}(\theta(i)(n))\} \quad \cup \\
 & & & \; \bigcup_{!i \in \mathsf{aM}_n} \{c \mid c \in \mathsf{pM}_n \wedge \widetilde{c} \in \mathsf{pt}(\delta(i)(n))\} \\
\mathsf{aM}'_{n+1} & = & \mathsf{aM}'_n & \cup \; \bigcup_{?i \in \mathsf{aM}'_n} \{c \mid c \in \overline{\mathsf{aM}'_n \cup \mathsf{pM}'_n} \wedge c \in \mathsf{pt}(\mathsf{dmM}(\theta,\delta)(i)(n))\} \quad \cup \\
 & & & \; \bigcup_{!i \in \mathsf{aM}'_n} \{c \mid c \in \mathsf{pM}'_n \wedge \widetilde{c} \in \mathsf{pt}(\delta(i)(n))\} \\
\mathsf{aM}''_{n+1} & = & \mathsf{aM}''_n & \cup \; \bigcup_{?i \in \mathsf{aM}''_n} \{c \mid c \in \overline{\mathsf{aM}''_n \cup \mathsf{pM}''_n} \wedge c \in \mathsf{pt}(\theta(i)(n))\} \quad \cup \\
 & & & \; \bigcup_{!i \in \mathsf{aM}''_n} \{c \mid c \in \mathsf{pM}''_n \wedge \widetilde{c} \in \mathsf{pt}(\mathsf{rnM}(\theta,\delta)(i)(n))\}
\end{array}$$

$$\begin{array}{lcll}
\mathsf{pM}_{n+1} & = & \mathsf{pM}_n & \setminus \; \bigcup_{!i \in \mathsf{aM}_n} \{c \mid c \in \mathsf{pM}_n \wedge \widetilde{c} \in \mathsf{pt}(\delta(i)(n))\} \\
\mathsf{pM}'_{n+1} & = & \mathsf{pM}'_n & \setminus \; \bigcup_{!i \in \mathsf{aM}'_n} \{c \mid c \in \mathsf{pM}'_n \wedge \widetilde{c} \in \mathsf{pt}(\delta(i)(n))\} \\
\mathsf{pM}''_{n+1} & = & \mathsf{pM}''_n & \setminus \; \bigcup_{!i \in \mathsf{aM}''_n} \{c \mid c \in \mathsf{pM}''_n \wedge \widetilde{c} \in \mathsf{pt}(\mathsf{rnM}(\theta,\delta)(i)(n))\}
\end{array}$$

By the definition of dmM and rnM:

$$\begin{array}{ll}
\mathsf{dmM}(\theta,\delta)(i)(n) = (\overline{\widetilde{\mathsf{pM}_n}} \cup D) \circledS \theta(i)(n) & \text{if } \; ?i \in \mathsf{aM}_n = \mathsf{aM}'_n = \mathsf{aM}''_n \\
\mathsf{rnM}(\theta,\delta)(i)(n) \; = (\mathsf{pM}_n \cup \mathsf{aM}_n \cup D) \circledS \delta(i)(n) & \text{if } \; !i \in \mathsf{aM}_n = \mathsf{aM}'_n = \mathsf{aM}''_n
\end{array}$$

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

$$\mathsf{dmM}(\theta,\delta)(i)(n) = (\overline{\widetilde{\mathsf{pM}_n}} \cup D) \circledS \theta(i)(n)$$

holds inside this union. It is enough to show that

$$c \in \mathsf{pt}(\theta(i)(n)) \Leftrightarrow c \in \mathsf{pt}((\overline{\widetilde{\mathsf{pM}_n}} \cup D) \circledS \theta(i)(n))$$

under the assumptions that $c \notin \mathsf{aM}_n$ and $c \in \overline{\mathsf{pM}_n}$. This follows since $\widetilde{c} \in \mathsf{pM}_1 \Leftrightarrow c \in \mathsf{pM}_1$ and the two assumptions imply that $c \notin \widetilde{\mathsf{pM}_n}$.

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

$$\mathsf{rnM}(\theta,\delta)(i)(n) = (\mathsf{pM}_n \cup \mathsf{aM}_n \cup D) \circledS \delta(i)(n)$$

holds inside this union. It is enough to show that

$$\widetilde{c} \in \mathsf{pt}(\delta(i)(n)) \Leftrightarrow \widetilde{c} \in \mathsf{pt}((\mathsf{pM}_n \cup \mathsf{aM}_n \cup D) \circledS \delta(i)(n))$$

under the assumption that $c \in \mathsf{pM}_n$. This follows since $\widetilde{c} \in \mathsf{pM}_1 \Leftrightarrow c \in \mathsf{pM}_1$ and the assumption imply that $\widetilde{c} \in \mathsf{pM}_n \cup \mathsf{aM}_n$. This proves that $\mathsf{aM}_{n+1} = \mathsf{aM}'_{n+1} = \mathsf{aM}''_{n+1}$. That $\mathsf{pM}_{n+1} = \mathsf{pM}'_{n+1} = \mathsf{pM}''_{n+1}$ follows accordingly. Finally, because of these equalities, $\mathsf{dmM}(\theta,\delta)(i)(n)$ simplifies to $\theta(i)(n)$ and $\mathsf{rnM}(\theta,\delta)(i)(n)$ simplifies to $\delta(i)(n)$ inside the definitions of dmM and rnM. This immediately proves the theorem. $\square$

Theorem 5. If $g \in H \to H$ is a strongly guarded function then $m2m_{I,O,P}(g) \in Mob_{m2m}(I,O,P)$.

Proof. Let us abbreviate $m2m_{I,O,P}(g)$ by f. Then by the definition of $m2m_{I,O,P}$ we have

$$f(\theta) = \mathsf{rnM}(\theta,\delta) \text{ where } \delta = g(\mathsf{dmM}(\theta,\delta))$$

The function f is well defined and strongly guarded because g is strongly guarded and dmM and rnM are weakly guarded. That f is privacy preserving follows from the following two lemmas.

Lemma 1. $f(\theta) = f(\mathsf{dmM}(\theta, f(\theta)))$.

Proof. The idea of the proof is to transform $f(\mathsf{dmM}(\theta, f(\theta)))$ to $f(\theta)$ by using the equalities from Theorem 4. By definition, $f(\mathsf{dmM}(\theta, f(\theta)))$ is equal to

$$\mathsf{rnM}(\mathsf{dmM}(\theta, f(\theta)), \gamma) \text{ where } \gamma = g(\mathsf{dmM}(\mathsf{dmM}(\theta, f(\theta)), \gamma))$$

By Theorem 4 and definition of f we have

$$\mathsf{dmM}(\theta, f(\theta)) = \mathsf{dmM}(\theta, \mathsf{rnM}(\theta,\delta)) = \mathsf{dmM}(\theta,\delta)$$

Hence, the recursive equation in γ reduces to

$$\gamma = g(\mathsf{dmM}(\mathsf{dmM}(\theta, \delta), \gamma))$$

But by Theorem 4 and definition of f, δ is a fix-point of the above equation

$$g(\mathsf{dmM}(\mathsf{dmM}(\theta, \delta), \delta)) = g(\mathsf{dmM}(\theta, \delta)) = \delta$$

Since fix-points are unique it follows that $\delta = \gamma$. Now, using again Theorem 4 and the above result we obtain

$$\mathsf{rnM}(\mathsf{dmM}(\theta, f(\theta)), \gamma) = \mathsf{rnM}(\mathsf{dmM}(\theta, \mathsf{rnM}(\theta, \delta)), \delta) = \mathsf{rnM}(\theta, \delta)$$

Hence, $f(\mathsf{dmM}(\theta, f(\theta))) = f(\theta)$. □

Lemma 2. $f(\theta) = \mathsf{rnM}(\theta, f(\theta))$.

Proof.

$$\begin{array}{ll} \mathsf{rnM}(\theta, f(\theta)) = & \\ \mathsf{rnM}(\theta, \mathsf{rnM}(\theta, \delta)) = & \{\text{by the definition of } f\} \\ \mathsf{rnM}(\theta, \delta) = & \{\text{by Theorem 4}\} \\ f(\theta) & \{\text{by the definition of } f\} \end{array}$$

This completes the proof. □

Theorem 6. $F_1 \odot F_2$ is a non-empty set of strongly guarded functions if F_1 and F_2 are non-empty sets of strongly guarded functions.

Proof. Since F_1, F_2 and $\mathbb{M}$ are non-empty we may find functions $f_1 \in F_1, f_2 \in F_2$ and $m_1, m_2, m_3 \in \mathbb{M}$. Based on these functions we construct a function f which is strongly guarded and satisfies the recursive equation in Definition 7. Let g be defined as follows.

$$g \in (H \times H) \times H \to H \times H$$
$$g((\varphi, \psi), \theta) = (f_1(m_1(\theta, \psi)), f_2(m_2(\theta, \varphi))$$

The way g is defined in terms of strongly and weakly guarded functions imply that g is strongly guarded. Thus μg is well defined, in which case it follows that μg is strongly guarded. That the function f defined below is also strongly guarded follows accordingly.

$$f \in H \to H$$
$$f(\theta) = m_3(\varphi, \psi) \text{ where } (\varphi, \psi) = (\mu\, g)(\theta)$$

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

Theorem 7. $F_1 \oplus F_2$ is a mobile m2m component if F_1 and F_2 are mobile m2m components.

Proof. Follows from Theorems 5 and 6. □

Theorem 8. $\nu x.F$ is a mobile m2m component if F is a mobile m2m component.

Proof. Follows from Theorem 5. □

C. Proofs: Mobile P2p Case

Theorem 9. The functions pP and aP are strongly guarded and the functions dmP and rnP are weakly guarded.

Proof. The proof is identical to that of Theorem 3. □

Theorem 10. The functions dmP and rnP satisfy the following properties.

$$\begin{array}{rcl} \mathsf{dmP}_{I,O,P}(\theta, \delta) & = & \mathsf{dmP}_{I,O,P}(\mathsf{dmP}_{I,O,P}(\theta, \delta), \delta) \\ & = & \mathsf{dmP}_{I,O,P}(\theta, \mathsf{rnP}_{I,O,P}(\theta, \delta)) \\ \mathsf{rnP}_{I,O,P}(\theta, \delta) & = & \mathsf{rnP}_{I,O,P}(\mathsf{dmP}_{I,O,P}(\theta, \delta), \delta) \\ & = & \mathsf{rnP}_{I,O,P}(\theta, \mathsf{rnP}_{I,O,P}(\theta, \delta)) \end{array}$$

Proof. The proof is similar to that of Theorem 4. □

Theorem 11. If $g \in H \to H$ is a strongly guarded function preserving port uniqueness then $p2p_{I,O,P}(g) \in Mob_{p2p}(I,O,P)$.

Proof. The proof of privacy preservation is similar to the one for the m2m case except that it uses Theorem 10, the p2p equivalent of Theorem 4. That $p2p_{I,O,P}(g)$ preserves port uniqueness follows trivially because dmP and rnP only remove messages. □

Theorem 12. $F_1 \otimes F_2$ is a mobile p2p component if F_1 and F_2 are mobile p2p components.

Proof. That $F_1 \otimes F_2$ is well defined and privacy preserving follows from Theorems 6 and 11. We only have to show that each $f \in F_1 \otimes F_2$ also preserves port uniqueness. With respect to f_1, f_2, m_1, m_2, m_3 and θ of Definition 7 this amounts to proving

$$m_3(\varphi, \psi) \in H_U$$

under the assumption that $\theta \in H_U$ and $\theta = \mathsf{dmP}(\theta, m_3(\varphi, \psi))$. The proof is by induction; the induction hypothesis is formalised by the following lemma. Let

$$\begin{aligned}
\mathsf{aP}^1_n &= \mathsf{aP}_{I_1,O_1,P_1}(m_1(\theta,\psi),\varphi)(n), & \mathsf{pP}^1_n &= \mathsf{pP}_{I_1,O_1,P_1}(m_1(\theta,\psi),\varphi)(n)\\
\mathsf{aP}^2_n &= \mathsf{aP}_{I_2,O_2,P_2}(m_2(\theta,\varphi),\psi)(n), & \mathsf{pP}^2_n &= \mathsf{pP}_{I_2,O_2,P_2}(m_2(\theta,\varphi),\psi)(n)\\
\mathsf{aP}_n &= \mathsf{aP}_{I,O,P}(\theta, m_3(\varphi,\psi))(n), & \mathsf{pP}_n &= \mathsf{pP}_{I,O,P}(\theta, m_3(\varphi,\psi))(n)
\end{aligned}$$

Lemma 3. For all $n \in Nat_+$:

(1) $(\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) \cap (\mathsf{aP}^2_n \cup \mathsf{pP}^2_n) = \{\}$

(2) $\mathsf{aP}^1_n \cup \mathsf{pP}^1_n \cup \mathsf{aP}^2_n \cup \mathsf{pP}^2_n = \mathsf{aP}_n \cup \mathsf{pP}_n$

(3) φ, ψ are port unique until time n

(4) $\mathsf{pts}(\varphi, n) \cap \mathsf{pts}(\psi, n) = \{\}$

Proof. Base Case: By definition

$$\begin{aligned}
\mathsf{aP}^1_1 &= ?I_1 \cup !O_1, & \mathsf{aP}^2_1 &= ?I_2 \cup !O_2, & \mathsf{aP}_1 &= ?(I_1 \setminus O_2) \cup ?(I_2 \setminus O_1) \cup !(O_1 \setminus I_2) \cup !(O_2 \setminus I_1)\\
\mathsf{pP}^1_1 &= ?!P_1, & \mathsf{pP}^2_1 &= ?!P_2, & \mathsf{pP}_1 &= ?!(P_1 \cup P_2 \cup (I_1 \cap O_2) \cup (I_2 \cap O_1))
\end{aligned}$$

The constraints imposed on I_1, O_1, P_1, I_2, O_2 and P_2 by Definition 21 imply the validity of (1) and (2). (3) follows since f_1 and f_2 are strongly guarded and preserves port uniqueness. (4) follows from (1) since f_1 and f_2 are privacy preserving.

Induction step: Expanding the definitions of aP and pP we obtain

$$\begin{aligned}
\mathsf{aP}^1_{n+1} &= (\mathsf{aP}^1_n \cup \mathsf{rP}^1_n \cup \mathsf{gP}^1_n) \setminus (\mathsf{sP}^1_n \cup \mathsf{hP}^1_n), & \mathsf{pP}^1_{n+1} &= (\mathsf{pP}^1_n \cup \mathsf{hP}^1_n) \setminus (\mathsf{sP}^1_n \cup \widetilde{\mathsf{sP}^1_n})\\
\mathsf{aP}^2_{n+1} &= (\mathsf{aP}^2_n \cup \mathsf{rP}^2_n \cup \mathsf{gP}^2_n) \setminus (\mathsf{sP}^2_n \cup \mathsf{hP}^2_n), & \mathsf{pP}^2_{n+1} &= (\mathsf{pP}^2_n \cup \mathsf{hP}^2_n) \setminus (\mathsf{sP}^2_n \cup \widetilde{\mathsf{sP}^2_n})\\
\mathsf{aP}_{n+1} &= (\mathsf{aP}_n \cup \mathsf{rP}_n \cup \mathsf{gP}_n) \setminus (\mathsf{sP}_n \cup \mathsf{hP}_n), & \mathsf{pP}_{n+1} &= (\mathsf{pP}_n \cup \mathsf{hP}_n) \setminus (\mathsf{sP}_n \cup \widetilde{\mathsf{sP}_n})
\end{aligned}$$

where

$$\begin{aligned}
\mathsf{rP}^1_n &= \textstyle\bigcup_{?i \in \mathsf{aP}^1_n} \{c \mid c \in \overline{\mathsf{pP}^1_n \cup \mathsf{aP}^1_n} \cap \mathsf{pt}(m_1(\theta,\psi)(i)(n))\}\\
\mathsf{rP}^2_n &= \textstyle\bigcup_{?i \in \mathsf{aP}^2_n} \{c \mid c \in \overline{\mathsf{pP}^2_n \cup \mathsf{aP}^2_n} \cap \mathsf{pt}(m_2(\theta,\varphi)(i)(n))\}\\
\mathsf{rP}_n &= \textstyle\bigcup_{?i \in \mathsf{aP}_n} \{c \mid c \in \overline{\mathsf{pP}_n \cup \mathsf{aP}_n} \cap \mathsf{pt}(\theta(i)(n))\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{sP}^1_n &= \textstyle\bigcup_{!i \in \mathsf{aP}^1_n} \{c \mid c \in (\mathsf{pP}^1_n \cup \mathsf{aP}^1_n) \cap \mathsf{pt}(\varphi(i)(n))\}\\
\mathsf{sP}^2_n &= \textstyle\bigcup_{!i \in \mathsf{aP}^2_n} \{c \mid c \in (\mathsf{pP}^2_n \cup \mathsf{aP}^2_n) \cap \mathsf{pt}(\psi(i)(n))\}\\
\mathsf{sP}_n &= \textstyle\bigcup_{!i \in \mathsf{aP}_n} \{c \mid c \in (\mathsf{pP}_n \cup \mathsf{aP}_n) \cap \mathsf{pt}(m_3(\varphi,\psi)(i)(n))\}
\end{aligned}$$

$$\begin{aligned}
\mathsf{gP}^1_n &= \{\widetilde{c} \mid c \in \mathsf{sP}^1_n \wedge c \in \mathsf{pP}^1_n\}, & \mathsf{hP}^1_n &= \{c, \widetilde{c} \mid c \in \mathsf{rP}^1_n \wedge \widetilde{c} \in (\mathsf{aP}^1_n \setminus \mathsf{sP}^1_n) \cup \mathsf{rP}^1_n\}\\
\mathsf{gP}^2_n &= \{\widetilde{c} \mid c \in \mathsf{sP}^2_n \wedge c \in \mathsf{pP}^2_n\}, & \mathsf{hP}^2_n &= \{c, \widetilde{c} \mid c \in \mathsf{rP}^2_n \wedge \widetilde{c} \in (\mathsf{aP}^2_n \setminus \mathsf{sP}^2_n) \cup \mathsf{rP}^2_n\}\\
\mathsf{gP}_n &= \{\widetilde{c} \mid c \in \mathsf{sP}_n \wedge c \in \mathsf{pP}_n\}, & \mathsf{hP}_n &= \{c, \widetilde{c} \mid c \in \mathsf{rP}_n \wedge \widetilde{c} \in (\mathsf{aP}_n \setminus \mathsf{sP}_n) \cup \mathsf{rP}_n\}
\end{aligned}$$

(1) holds by the induction hypothesis for n. Any port

$$p \in (\mathsf{aP}^1_{n+1} \cup \mathsf{pP}^1_{n+1}) \setminus (\mathsf{aP}^1_n \cup \mathsf{pP}^1_n)$$

is by definition contained in one of the following two sets.

$$\mathsf{pts}(\theta, n) \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n}, \qquad \mathsf{pts}(\psi, n)$$

In the first case the assumptions on θ and the assumption that (2) holds for n imply that $p \notin \mathsf{aP}^2_{n+1} \cup \mathsf{pP}^2_{n+1}$. In the second case we may deduce the same since the fact that f_2 is privacy preserving implies that by definition

$$\mathsf{sP}^2_n = \mathsf{pts}(\psi, n), \qquad \mathsf{sP}^2_n \cap (\mathsf{aP}^2_{n+1} \cup \mathsf{pP}^2_{n+1}) = \{\}$$

That any new port p contained in $\mathsf{aP}^2_{n+1} \cup \mathsf{pP}^2_{n+1}$ is not contained in $\mathsf{aP}^1_{n+1} \cup \mathsf{pP}^1_{n+1}$ follows accordingly. Hence, (1) holds for $n+1$.

To see that (2) holds for $n+1$, first remember that by definition

$$\begin{aligned}
\mathsf{aP}^1_{n+1} \cup \mathsf{pP}^1_{n+1} &= (\mathsf{aP}^1_n \cup \mathsf{pP}^1_n \cup \mathsf{rP}^1_n) \setminus \mathsf{sP}^1_n \\
\mathsf{aP}^2_{n+1} \cup \mathsf{pP}^2_{n+1} &= (\mathsf{aP}^2_n \cup \mathsf{pP}^2_n \cup \mathsf{rP}^2_n) \setminus \mathsf{sP}^2_n \\
\mathsf{aP}_{n+1} \cup \mathsf{pP}_{n+1} &= (\mathsf{aP}_n \cup \mathsf{pP}_n \cup \mathsf{rP}_n) \setminus \mathsf{sP}_n
\end{aligned}$$

Since (2) holds for n it follows from the definitions of $\mathsf{sP}^1_n, \mathsf{sP}^2_n$ and sP_n that

$$\mathsf{sP}^1_n \cup \mathsf{sP}^2_n = \mathsf{sP}_n$$

Similarly, we have that

$$\mathsf{rP}^1_n \setminus (\mathsf{aP}^2_n \cup \mathsf{pP}^2_n) \cup \mathsf{rP}^2_n \setminus (\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) = \mathsf{rP}_n$$

Hence, (2) holds for $n+1$.

That $m_1(\theta, \varphi)$ and $m_2(\theta, \psi)$ are port unique until n follows straightforwardly from the assumptions that (2) and (3) holds for n, the assumptions on θ and the fact that f_1 and f_2 are privacy preserving. But then the fact that f_1 and f_2 are strongly guarded and preserves port uniqueness implies that (3) holds for $n+1$.

By the induction hypothesis (4) holds for n. This implies that (4) also holds for $n+1$ since (1) holds for $n+1$ and f_1 and f_2 are privacy preserving.

This completes the proof. □

Theorem 13. $F_1 \otimes F_2$ is a restrictive p2p component if F_1 and F_2 are restrictive p2p components.

Proof. Let $f \in F_1 \otimes F_2$. Since f is privacy preserving we may assume that $\theta = \mathsf{dmP}_{I,O,P}(\theta, f(\theta))$. Suppose $?n \in \mathsf{pt}(f(\theta)(o)(t))$. With respect to Definition 7 there are restrictive p2p functions $f_1 \in F_1$, $f_2 \in F_2$ and histories ψ, φ such that

$$?n \in \mathsf{pt}(f_1(m_1(\theta, \psi))(o)(t)) \quad \text{or} \quad ?n \in \mathsf{pt}(f_2(m_2(\theta, \varphi))(o)(t))$$

Suppose that

$$?n \in \mathsf{pt}(f_1(m_1(\theta, \psi))(o)(t))$$

Since f_1 is a restrictive p2p function it follows that

$$f_1(m_1(\theta, \psi))\downarrow_t = f_1(m_1(\theta, \psi)\dagger_n)\downarrow_t = f_1(m_1(\theta\dagger_n, \psi\dagger_n))\downarrow_t$$

Moreover, since $?n$ belongs to f_1 at time $t-1$ and $F_1 \otimes F_2$ is a p2p component, it follows that $?n$ is not among the ports of f_2 at time $t-1$. Hence, either $?n$ has never been a port of f_2 or it has been forwarded by f_2 to another component. In either case, since f_2 is a restrictive p2p function it follows that

$$f_2(m_2(\theta, \varphi))\downarrow_t = f_2(m_2(\theta, \varphi)\dagger_n)\downarrow_t = f_2(m_2(\theta\dagger_n, \varphi\dagger_n))\downarrow_t$$

Hence $f(\theta)\downarrow_t = f(\theta\dagger_n)\downarrow_t$. If $?n$ belongs to f_2 at time $t-1$ the proof is similar.

Suppose that $!n \in \mathsf{pt}(f(\theta)(o)(t))$. Then

$$!n \in \mathsf{pt}(f_1(m_1(\theta, \psi))(o)(t)) \quad \text{or} \quad !n \in \mathsf{pt}(f_2(m_2(\theta, \varphi))(o)(t))$$

Suppose that

$$!n \in \mathsf{pt}(f_1(m_1(\theta,\psi))(o)(t))$$

Since f_1 is a restrictive p2p function it follows that

$$f_1(m_1(\theta,\psi))\downarrow_t = (f_1(m_1(\theta,\psi))\dagger_n)\downarrow_t$$

Moreover, since $!n$ belongs to f_1 at time $t-1$ and $F_1 \otimes F_2$ is a p2p component, it follows that $!n$ is not among the ports of f_2 at time $t-1$. Hence, either $!n$ has never been a port of f_2 or it has been forwarded by f_2 to another component. In either case, since f_2 is a restrictive p2p function it follows that

$$f_2(m_2(\theta,\varphi))\downarrow_t = (f_2(m_2(\theta,\varphi))\dagger_n)\downarrow_t$$

Hence, $f(\theta)\downarrow_t = (f(\theta)\dagger_n)\downarrow_t$. If $!n$ belongs to f_2 at time $t-1$ the proof is similar. Thus, $F_1 \otimes F_2$ is a restrictive p2p component. □

D. Proofs: p2p Implies m2m

Theorem 14. For all $n \in Nat_+$ and $\theta,\delta \in H$:

$$\begin{array}{lll}
(1) & \mathsf{aP}(\theta,\delta)(n) & \subseteq \quad \mathsf{aM}(\theta,\delta)(n) \\
(2) & \mathsf{pM}(\widetilde{\theta,\delta})(n) & \subseteq \quad \mathsf{aP}(\theta,\delta)(n) \cup \mathsf{pP}(\theta,\delta)(n) \\
(3) & \mathsf{aP}(\theta,\delta)(n) \cup \mathsf{pP}(\theta,\delta)(n) & \subseteq \quad \mathsf{aM}(\theta,\delta)(n) \cup \mathsf{pM}(\theta,\delta)(n)
\end{array}$$

Proof. The proof is by induction on n.
Base Case:

$$\begin{array}{lllll}
(1) & \mathsf{aP}_1 & = & ?I \cup\, !O & \subseteq \quad \mathsf{aM}_1 \\
(2) & \widetilde{\mathsf{pM}_1} & = & ?!P & \subseteq \quad \mathsf{aP}_1 \cup \mathsf{pP}_1 \\
(3) & \mathsf{aP}_1 \cup \mathsf{pP}_1 & = & ?!P \cup ?I \cup\, !O & = \quad \mathsf{aM}_1 \cup \mathsf{pM}_1
\end{array}$$

Induction Step: To prove (1) for $n+1$ there are three cases to consider:

(a) $p \in \mathsf{aP}_n$

(b) $p \in \mathsf{rP}_n \setminus \mathsf{hP}_n$

(c) $p \in \mathsf{gP}_n$

That (1) holds for $n+1$ in the case of (a) follows from the definition of aM_{n+1} and the assumption that (1) holds for n.

The assumption that (2) holds for n and the definitions of rP_n, hP_n and rM_n imply that $\mathsf{rP}_n \setminus \mathsf{hP}_n \subseteq \mathsf{rM}_n$. Hence, (1) holds for $n+1$ in the case of (b) since by definition $\mathsf{rM}_n \subseteq \mathsf{aM}_{n+1}$.

It follows from the assumption that (3) holds for n and the definitions of gP_n and gM_n that any port in gP_n is also an element of $\mathsf{aM}_n \cup \mathsf{gM}_n$. Hence, (1) holds for $n+1$ also in the case of (c).

To prove that (2) holds for $n+1$ remember that $\mathsf{pM}_{n+1} = \mathsf{pM}_n \setminus \mathsf{gM}_n$. Hence,

$$\widetilde{\mathsf{pM}_{n+1}} \overset{def}{=} \widetilde{\mathsf{pM}_n} \setminus \widetilde{\mathsf{gM}_n} \overset{def,hyp}{\subseteq} (\mathsf{aP}_n \cup \mathsf{pP}_n) \setminus \mathsf{sP}_n \overset{def}{\subseteq} \mathsf{aP}_{n+1} \cup \mathsf{pP}_{n+1}$$

That (3) holds for $n+1$ follows straightforwardly as follows.

$$\mathsf{pP}_{n+1} \cup \mathsf{aP}_{n+1} \overset{def}{=} (\mathsf{pP}_n \cup \mathsf{aP}_n \cup \mathsf{rP}_n) \setminus \mathsf{sP}_n \overset{def,hyp}{\subseteq} \mathsf{aM}_n \cup \mathsf{pM}_n \cup \mathsf{rM}_n \overset{def}{=} \mathsf{pM}_{n+1} \cup \mathsf{aM}_{n+1} \quad \square$$

Theorem 15. For all $n \in Nat_+$ and $\theta,\delta \in H$:

$$\begin{array}{lll}
(1) & \mathsf{aP}(\theta,\delta)(n) & \subseteq \quad \mathsf{aM}(\theta,\mathsf{rnP}(\theta,\delta))(n) \\
(2) & \mathsf{pM}(\widetilde{\theta,\delta})(n) & \subseteq \quad \mathsf{aP}(\theta,\mathsf{rnP}(\theta,\delta))(n) \cup \mathsf{pP}(\theta,\mathsf{rnP}(\theta,\delta))(n) \\
(3) & \mathsf{aP}(\theta,\delta)(n) \cup \mathsf{pP}(\theta,\delta)(n) & \subseteq \quad \mathsf{aM}(\theta,\mathsf{rnP}(\theta,\delta))(n) \cup \mathsf{pM}(\theta,\mathsf{rnP}(\theta,\delta))(n)
\end{array}$$

Proof. The proof is almost the same as for Theorem 14. □

Theorem 16. The functions dmP and dmM satisfy the following property.

$$\mathsf{dmP}_{I,O,P}(\theta,\delta) = \mathsf{dmP}_{I,O,P}(\mathsf{dmM}_{I,O,P}(\theta,\delta),\delta)$$

Proof. The proof is quite similar to the proofs of Theorems 4 and 10; it is based on the inductive definitions of $\mathsf{aP}, \mathsf{pP}, \mathsf{aM}$ and pM. The difference is that in this case we have to relate the sets of active and passive ports in the m2m and p2p paradigms. As before, to simplify the notation, we define:

$$\begin{array}{lll} \mathsf{aP}_n \equiv \mathsf{aP}_{I,O,P}(\theta,\delta)(n), & \mathsf{aP}'_n \equiv \mathsf{aP}_{I,O,P}(\mathsf{dmM}(\theta,\delta),\delta)(n) \\ \mathsf{pP}_n \equiv \mathsf{pP}_{I,O,P}(\theta,\delta)(n), & \mathsf{pP}'_n \equiv \mathsf{pP}_{I,O,P}(\mathsf{dmM}(\theta,\delta),\delta)(n) \end{array}$$

The induction hypothesis is that $\mathsf{aP}_n = \mathsf{aP}'_n$ and $\mathsf{pP}_n = \mathsf{pP}'_n$.

Base Case: $\mathsf{aP}_1 = \mathsf{aP}'_1 = ?I \cup !O$ and $\mathsf{pP}_1 = \mathsf{pP}'_1 = ?!P$.

Induction Step: By induction hypothesis $\mathsf{aP}_n = \mathsf{aP}'_n$ and $\mathsf{pP}_n = \mathsf{pP}'_n$. By definition:

$$\mathsf{aP}_{n+1} = (\mathsf{aP}_n \cup \mathsf{rP}_n \cup \mathsf{gP}_n) \setminus (\mathsf{sP}_n \cup \mathsf{hP}_n), \quad \mathsf{pP}_{n+1} = (\mathsf{pP}_n \cup \mathsf{hP}_n) \setminus (\mathsf{sP}_n \cup \widetilde{\mathsf{sP}_n})$$

$$\mathsf{aP}'_{n+1} = (\mathsf{aP}'_n \cup \mathsf{rP}'_n \cup \mathsf{gP}'_n) \setminus (\mathsf{sP}'_n \cup \mathsf{hP}'_n), \quad \mathsf{pP}'_{n+1} = (\mathsf{pP}'_n \cup \mathsf{hP}'_n) \setminus (\mathsf{sP}'_n \cup \widetilde{\mathsf{sP}'_n})$$

By induction hypothesis, $\mathsf{aP}_n = \mathsf{aP}'_n$ and $\mathsf{pP}_n = \mathsf{pP}'_n$. This implies:

$$\begin{array}{lll} \mathsf{rP}_n & = & \bigcup_{?i\in\mathsf{aP}_n}\{p \mid p \in \overline{\mathsf{pP}_n \cup \mathsf{aP}_n} \cap \mathsf{pt}(\theta(i)(n))\} \\ \mathsf{rP}'_n & = & \bigcup_{?i\in\mathsf{aP}_n}\{p \mid p \in \overline{\mathsf{pP}_n \cup \mathsf{aP}_n} \cap \mathsf{pt}(\mathsf{dmM}_{I,O,P}(\theta,\delta)(i)(n))\} \end{array}$$

Moreover,

$$\mathsf{hP}_n = \{p,\widetilde{p} \mid p \in \mathsf{rP}_n \wedge \widetilde{p} \in (\mathsf{aP}_n \setminus \mathsf{sP}_n) \cup \mathsf{rP}_n\}, \quad \mathsf{hP}'_n = \{p \mid p \in \mathsf{rP}'_n \wedge \widetilde{p} \in (\mathsf{aP}_n \setminus \mathsf{sP}'_n) \cup \mathsf{rP}'_n\}$$

and

$$\begin{array}{lllll} \mathsf{sP}_n & = & \mathsf{sP}'_n & = & \bigcup_{!i\in\mathsf{aP}_n}\{p \mid p \in (\mathsf{pP}_n \cup \mathsf{aP}_n) \cap \mathsf{pt}(\delta(i)(n))\} \\ \mathsf{gP}_n & = & \mathsf{gP}'_n & = & \{\widetilde{p} \mid p \in \mathsf{sP}_n \wedge p \in \mathsf{pP}_n\} \end{array}$$

As a consequence, we only have to prove that $\mathsf{rP}_n = \mathsf{rP}'_n$. By the definition of dmM:

$$\mathsf{dmM}(\theta,\delta)(i)(n) = \overline{(\widetilde{\mathsf{pM}_n} \cup D)} \circledS \theta(i)(n) \quad \text{if} \quad ?i \in \mathsf{aM}_n$$

By Theorem 14 it follows that $\mathsf{dmM}(\theta,\delta)(i)(n) = \theta(i)(n)$ inside rP'_n. Hence, $\mathsf{aP}_{n+1} = \mathsf{aP}'_{n+1}$ and $\mathsf{pP}_{n+1} = \mathsf{pP}'_{n+1}$. □

Theorem 17. The functions rnM and rnP satisfy the following property.

$$\mathsf{rnP}(\theta,\delta) = \mathsf{rnM}(\theta,\mathsf{rnP}(\theta,\delta))$$

Proof. To simplify the notation, we define:

$$\mathsf{aM}'_n \equiv \mathsf{aM}_{I,O,P}(\theta,\mathsf{rnP}(\theta,\delta))(n), \quad \mathsf{pM}'_n \equiv \mathsf{pM}_{I,O,P}(\theta,\mathsf{rnP}(\theta,\delta))(n)$$

Unfolding the definition of $\mathsf{rnM}(\theta,\mathsf{rnP}(\theta,\delta))$ we obtain:

$$\begin{array}{ll} ((\mathsf{aM}'_n \cup \mathsf{pM}'_n) \cap (\mathsf{aP}_n \cup \mathsf{pP}_n)) \cup D \circledS \delta(i)(n) & \text{if } !i \in \mathsf{aM}'_n \cap \mathsf{aP}_n \\ \langle\rangle & \text{otherwise} \end{array}$$

So we need to show that:

$$\mathsf{aP}_n \subseteq \mathsf{aM}'_n, \qquad \mathsf{aP}_n \cup \mathsf{pP}_n \subseteq \mathsf{aM}'_n \cup \mathsf{pM}'_n$$

This follows immediately from Theorem 15. □

Theorem 18. If $f \in Mob_{p2p}(I,O,P)$ then $f \in Mob_{m2m}(I,O,P)$.

Proof. We split the proof into two lemmas.

Lemma 4. $\mathsf{rnM}(\theta,f(\theta)) = f(\theta)$

Proof. $\mathsf{rnM}(\theta,f(\theta)) \overset{hyp}{=} \mathsf{rnM}(\theta,\mathsf{rnP}(\theta,f(\theta))) \overset{Thm\ 17}{=} \mathsf{rnP}(\theta,f(\theta)) \overset{hyp}{=} f(\theta)$ □

Lemma 5. $f(\mathsf{dmM}(\theta, f(\theta))) = f(\theta)$

Proof. By the hypothesis $f(\theta)$ satisfies

$$f(\theta) = \mathsf{rnP}(\theta, \delta) \text{ where } \delta = f(\mathsf{dmP}(\theta, \delta))$$

Since f is p2p, $f(\theta) = \delta$ is the unique fix-point of the above recursive equation. Let γ be such that

$$f(\mathsf{dmM}(\theta, f(\theta))) = \mathsf{rnP}(\theta, \gamma) \text{ where } \gamma = f(\mathsf{dmP}(\mathsf{dmM}(\theta, \delta), \gamma))$$

It follows that δ satisfies the recursive equation in γ

$$f(\mathsf{dmP}(\mathsf{dmM}(\theta, \delta), \delta)) \overset{Thm\ 16}{=} f(\mathsf{dmP}(\theta, \delta)) \overset{hyp}{=} f(\theta) = \delta$$

Since γ is the unique fix-point, it follows that $\delta = \gamma$. Hence,

$$f(\mathsf{dmM}(\theta, f(\theta))) = \mathsf{rnP}(\theta, \delta) = f(\theta)$$

This completes the proof. □

Received January 1998
Accepted in revised form August 2000 by I. J. Hayes