TECHNISCHE
UNIVERSITÄT
MÜNCHEN

INSTITUT FÜR INFORMATIK

Sonderforschungsbereich 342:
Methoden und Werkzeuge für die Nutzung
paralleler Rechnerarchitekturen

Compositional Specification of Mobile Systems

Radu Grosu Ketil Stølen

TUM-I9748
SFB-Bericht Nr. 342/29/97 A
November 97

TUM–INFO–11-I9748-350/1.–FI

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

Druck: Fakultät für Informatik der
Technischen Universität München

Formal Aspects of Computing (1997) 3: 1–000

Compositional Specification of Mobile Systems

Radu Grosu[1] and Ketil Stølen[2]

[1]TU München, D-80290 München, Germany
[2]Institute for Energy Technology, P.O.Box 173, N-1751 Halden, Norway

Keywords: Denotational Model, Data-flow, Input/Output Relation, Mobile System, Many-to-many Communication, Point-to-point Communication, Specification, Timing.

Abstract. This paper generalizes a specification technique based on input/output relations on streams to describe mobile systems. 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 under-pinned by denotational models. The models are formulated in the context of timed nondeterministic data-flow networks and presented in a step-wise fashion. The emphasis is on capturing the special kind of dynamic hiding characterizing mobile systems. We demonstrate the proposed approach in a number of small examples.

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, EN86, Tho89, BB90, Mes91, MPW92a, MPW92b]. A denotational understanding of mobility is, however, an essential prerequisite for the compositional development of mobile, and consequently object-oriented reactive systems. Recently several researchers have studied mobility in a denotational setting; see for example [JJ95, FMS96, Sta96]. These denotational approaches are all directed towards the π-calculus. 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

Correspondence and offprint requests to: grosu@informatik.tu-muenchen.de, Ketil.Stoelen@hrp.no

component may change its communication partners on the basis of computation and interaction [Mil91]. This means, for example, that this paper is not concerned with the kind of mobility achieved by allowing the components to communicate (migrate) processes (although this can easily be simulated by the communication of ports).

The use of *input/output relations* (I/O-relations) to specify computerized components is well-known. For example, VDM [Jon90] and Z [Spi88] are both based on this approach: A specification of a sequential component C characterizes 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.

Reactive components can be specified in a similar manner. For example, Focus [BS97] is based on I/O-relations: A specification of a reactive component C characterizes the relationship between its 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 reactive systems and mobile systems is the latter's much more sophisticated concept of hiding. In mobile systems the scope of variables changes dynamically during 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.

This paper generalizes traditional I/O-relations on streams to specify mobile systems with respect to these three communication paradigms. The presented approach is fully compositional and semantically under-pinned by denotational models expressed in the context of timed nondeterministic data-flow networks. We consider networks of autonomous components communicating via directed channels in a time-synchronous and message-asynchronous manner. Time-synchrony is achieved by using a global clock that splits 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 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 telephones network discussed in [Mil91]. In Example 3, we specify a variant in the m2m-format; in Example 4 and 5, we specify p2p-variants.

The paper is organized 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. Finally 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 both finite and 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 set 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*. 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. We model *nondeterministic 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 behavior.

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. For $t = 0$, by $r{\downarrow_t}$ 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

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

This definition is naturally extended to functions mapping tuples of input histories to tuples of output histories. A weakly guarded function is *non-expansive* and a strongly guarded function is *contractive* with respect to the Baire metric [Eng77] on streams[1]. Hence, by Banach's fix-point theorem, each strongly guarded function has a unique fix-point (in the case of feedback). It is also well-known [Eng77] that the functional composition of a strongly and a weakly guarded function yields a strongly guarded function.

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 $1 \le j \le n$;
- $\#s$ is the length of s;
- $s(j)$ is the jth element of s if $1 \le j \le \#s$;
- $\langle a_1, \ldots, a_j\rangle$ 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 way: For each t, $(r(t), s(t))$ is filtered away iff it is not in X. For instance:

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

3. Many-to-Many Communication

In this section we consider m2m-communication. We start by explaining what it means for a function to be *privacy preserving*; then we define components in terms

[1] See Appendix A for a definition of this metric.

of such functions and we introduce operators for parallel composition and hiding. On top of this formalism we build a small specification language in an example-driven manner.

3.1. Privacy Preservation

A stream processing function $f \in H \to H$, modeling a component in the m2m-case, is not only required to be strongly guarded, but also to be privacy preserving. The privacy preservation property formalizes the rules for how components may gain access to ports.

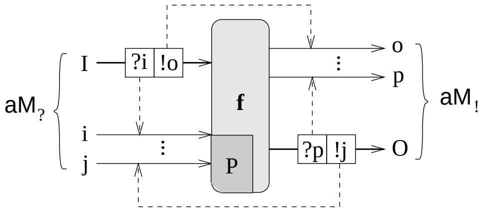

Fig. 1

The behavior of a privacy preserving function f can be described with respect to Figure 1 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 port 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 and by pM_t the set of passive ports at time t. Initially, at time 0 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. After all, 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 the output of f, respectively. The active and passive ports of f can be characterized as in the following definition.

Definition 2. (Active and passive ports) For any $I, O, P \subseteq N$; $\theta, \delta \in H$ and $t \in \mathit{Nat}_+$; 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

$$\mathsf{rM}_t \equiv \textstyle\bigcup_{?i \in \mathsf{aM}_t} \{p \mid p \in \overline{\mathsf{pM}_t} \wedge p \in \theta(i)(t)\}$$

$$\mathsf{gM}_t \equiv \textstyle\bigcup_{!i \in \mathsf{aM}_t} \{p \mid p \in \mathsf{pM}_t \wedge \widetilde{p} \in \delta(i)(t)\}$$

Then the sets of active and passive ports at time t are characterized 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.

In the definition of privacy preservation (Definition 4) we use the functions dmM and rnM to constrain f to maintain the privacy invariant with respect to active and passive ports informally described above. The functions dmM and rnM characterize 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 other words, 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 more sophisticated type-construct; a second alternative is to impose an environment assumption in all definitions characterizing exactly those input histories in which the environment sticks to the rules of the game; 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 satisfactory because we are only interested in environments that can be understood as m2m-components in accordance with Definition 5; 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 characterizing the actual domain and range of a function, also correct environment mistakes. Formally:

Definition 3. (Domain and range) For any $t \in Nat_+$; $I, O, P \subseteq N$; $\theta, \delta \in H$; the domain and range at time t are characterized by:

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

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

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

We can now define what it means for a function to be privacy preserving.

Definition 4. (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 the 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 it never send 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 (respectively 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 *m2m-functions*.

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

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

3.2. M2m-Components

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

Definition 5. (M2m-component) An m2m-component with initial interface (I,O) and initial set of passive port-names P is represented by a nonempty set of m2m-functions $F \subseteq Mob_{m2m}(I, O, P)$ that is closed in the following sense:

$$\forall f \in Mob_{m2m}(I, O, P) : (\forall \theta \in H : \exists f' \in F : f(\theta) = f'(\theta)) \Rightarrow f \in F$$

The closure expresses the black-box view of a component. If an observer is allowed to see only the complete input and output histories, it cannot observe whether the component chooses another function f' in F for each input history θ.

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

3.3. Typed Channels and Tuple Messages

The m2m-model introduced above is both simple and elegant, but not very useful from a practical point of view; it can, however, easily be extended with more practical features. In this section we outline how it can be modified to handle:

- 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 one 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. T is 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 way:

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

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 2, 3, 4, 5 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 parallel composition and hiding in Sections 3.5 and 3.6, and when we prove some results about the m2m-model in Appendix B.

3.4. Elementary M2m-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. In this section we introduce elementary specifications; composite specifications and the use of explicit hiding are treated in Section 3.7.

Since the m2m-model is timed, we can easily handle real-time. Nevertheless, since this paper is concerned with the specification of mobility and not with the specification of real-time requirements, we abstract away the timing and work with untimed streams when we write specifications. $H_A \equiv N \to M^\omega$ is the set of all *untimed* communication histories. For any $\theta \in H_T$, by $\overline{\theta}$ we denote its *time-abstraction*: the element in H_A obtained from θ by concatenating the finite sub-streams in each infinite stream into a stream of messages[2]. For instance, given that $\frown$ is the concatenation operator for streams, we have:

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

We start by specifying the behavior 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 a central that inputs questions and distributes them to the consultants depending on workload, specialization and experience; each question forwarded by the central 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 $c : (Q \times !N)$

out

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

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 central. Its set of output ports is initially empty. The lower-most frame, called the *body*, describes the dynamic behavior by a function *con* defined by the where-clause. In any elementary specification, $\mathsf{in} \in H_A$ represents the input history and

[2] Although we already used over-line for complement, the context should make clear which operator is intended.

$\mathsf{out} \in H_A$ 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 $\{n \mapsto m_1, \ldots, m_k\} \& \theta$ we mean $\{n \mapsto m_1\} \& \ldots \{n \mapsto m_k\} \& \theta$. □

We assume that each specification S has associated a unique, infinite set of private port names P_s[3]. As shown later, in Example 5, this set can be referenced by using the keyword priv. The *semantics* of an elementary specification S with external interface (I, O) and body B is then defined as follows:

$$\begin{aligned} [\![\, S \,]\!] \equiv \{ \quad & g \in Mob_{m2m}(I, O, P_s) \mid \forall\, in' \in H_T : \exists\, out' \in H_T : \\ & out' = g(in') \;\wedge\; B(\mathsf{in}, \mathsf{out}) \qquad \text{where} \\ & \mathsf{in} = \overline{\mathsf{dmM}_{I,O,P}(in', out')}, \quad \mathsf{out} = \overline{out'} \quad \} \end{aligned}$$

In the above definition, the mobility of g enforces that $out' = \mathsf{rnM}_{I,O,P}(in', out')$. Hence, it is enough to define out as the time abstraction of out'.

Note the importance of implicitly assuring time guardedness and privacy preservation in the elementary specification S by imposing it at the semantic level. Time 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. Moreover the implementer is free to develop standard techniques assuring time guardedness and mobility.

Another important observation is that no untimed specification B can violate time guardedness; one can always add the necessary empty sequences to assure it. A similar result is obtained for privacy preservation, if names are made abstract, i.e., if we require that any name constructor and any existential quantification of names is in $?!P \cup ?I \cup !O$.

3.5. M2m-Composition

The *parallel composition* of two m2m-components F_1 and F_2 is illustrated by the network in Figure 2.

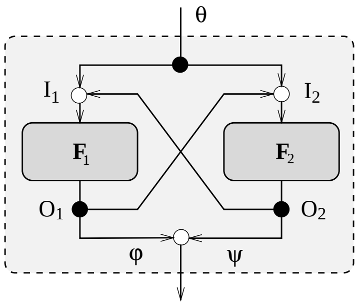

Fig. 2.

The hollow circles denote *interference points*, i.e., points where the environment, F_1 and F_2 may output along the same channel in the same time unit. In our approach,

[3] In practice this is achieved by associating to each specification a unique identifier. This is used to generate the unique set of port names.

interference is modelled by building copies of a *merge node* into the interference points and, therefore, implicitly into the network operators. This allows components to be described in a very abstract and, in our opinion, intuitive way. The merge node M takes two named communication histories as input and yields their merge as output. Any occurrence of 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 real-time, although this plays no role in this paper), M should neither add nor reduce delay. This means that the output history of M for some channel n during time unit k must be a merge of the two finite streams characterizing the input histories on n in time unit k. Moreover, M should not fix the interleaving. Thus, any interleaving of the messages received within a time unit should be allowed. Hence, M is nondeterministic in the sense that a pair of input histories may result in several (often infinitely many) different output histories.

The definition below formalizes 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 6. (Merge function on finite streams) FM is the set-valued function such that

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

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

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

It is now straightforward to define the merge node.

Definition 7. (Merge node) 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 $f \in \mathsf{M}$ is weakly guarded since it considers in a time unit t only on the messages received in the same time unit t. Note also that M is deterministic (it yields a set containing only one output history) 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 m2m-composition. Note the close relationship to the Figure 2.

Definition 8. (M2m-composition) Given two 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:

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

$$F_1 \odot F_2 \equiv \{\ \ f \in H \to H \ \mid \forall \theta \in H : \exists f_1 \in F_1;\ f_2 \in F_2;\ m_1, m_2, m_3 \in \mathsf{M} :$$

$$f(\theta) = m_3(\varphi,\psi) \text{ where } \varphi = f_1(m_1(\theta,\psi)),\ \psi = f_2(m_2(\theta,\varphi)) \ \ \}$$

In Appendix B, Theorem 7, we prove that $F_1 \oplus F_2$ belongs to $Comp_{m2m}(I, O, P)$. The functions $f \in F_1 \odot F_2$ are additionally constrained with 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.

3.6. 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 an explicit *hiding operator*. If Q is a set of port names contained in the initial interface of the m2m-component F, then $\nu Q : F$ is the m2m-component obtained from F by adding Q to the initial set of passive port names and deleting Q from the initial interface. The domain and range of the m2m-functions modeling $\nu Q : F$ are modified accordingly. As a consequence, only components receiving $\widetilde{p} \in ?!Q$ as a message can communicate with F via p later on.

Definition 9. (Hiding) Given an m2m-component $F \subseteq Mob_{m2m}(I', O', P')$ and a set of port names Q. Then $\nu Q : F$ is defined as below:

$$I \equiv I' \setminus Q, \quad O \equiv O' \setminus Q, \quad P \equiv P' \cup Q$$

$$\nu Q : F \equiv \{ \quad m2m_{I,O,P}(f) \quad | \quad f \in F \quad \}$$

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 behavior 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.

3.7. Composite M2m-Specifications

In Section 3.4, we introduced elementary m2m-specifications; in this section, we compose m2m-specifications into composite specifications. We first have a look at a simple example; then we use the m2m-model to define the semantics of such composite specifications.

Example 2. Consultancy network:

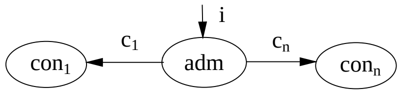

Fig. 3.

In Example 1, we specified a consultant communicating with an administrator and a number of customers; we now specify the administrator and the resulting consultancy network consisting of the administrator and n consultants. The consultancy network, whose initial configuration is illustrated graphically by Figure 3, is described by the composite specification CON_NET.

The consultancy network consists of the m2m-composition of the administrator ADM and of n consultants CON. Their initial input and output ports are renamed to the input and to the output ports within the square brackets to the left and to the right of $\rhd$ respectively. Renaming is positional and defines a new specification.

CON_NET	m2m
in	$i : (Q \times !N)$
out	
loc	$c_1, \ldots, c_n : (Q \times !N)$
$\mathrm{ADM}[i \rhd c_1, \ldots, c_n] \oplus \mathrm{CON}[c_1 \rhd] \oplus \ldots \oplus \mathrm{CON}[c_n \rhd]$	

Initially, the consultancy network has one external input port $?i$ on which it inputs questions from customers. Moreover, it has n local channels $c_1, \ldots, c_n$ 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	$c_1, \ldots, c_n : (Q \times !N)$
$\exists\, p \in \mathcal{P}(\{c_1, \ldots, c_n\})^\infty : adm(p)(\mathsf{in}) = \mathsf{out}$	
where $\forall\, q \in Q \times !N;\ v \in H_A;\ p \in \mathbb{P}(\{c_{r_1}, \ldots, c_{r_n}\})^\infty :$	
$adm(p)(\{i \mapsto q\} \,\&\, v) = (\bigcup_{c \in \mathsf{ft}.p}\{c \mapsto q\}) \,\&\, adm(\mathsf{rt}.p)(v)$	

For any non-empty stream s, we have that $s = \langle \mathsf{ft}.s \rangle \frown \mathsf{rt}.s$. 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 copy of this particular question. Hence, p is used as an *oracle*. □

We assume that the set of local ports P of a composite specification S is uniquely mapped to a new set of port names P_s such that it does not collide with the private ports of the component specifications. This mapping can be defined, for instance, by using the unique identifier of the composite specification. Denote this mapping by η_s. The semantic meaning of a composite specification S with sub-specifications $S_1, \ldots, S_n$ and set of local ports P is then defined as follows:

$$[\![\ S\ ]\!] \equiv \nu P_s : ([\![\ S_1[\eta_s]\ ]\!] \oplus \ldots \oplus [\![\ S_n[\eta_s]\ ]\!])$$

where $S_i[\eta_s]$ is the specification S_i with interface ports renamed according to η_s.

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.

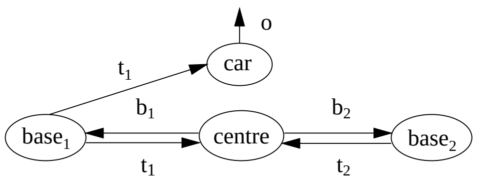

Fig. 4.

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 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. This is demonstrated by the specification below. The system, whose initial configuration is illustrated by Fig. 4, is described by a composite specification as follows:

TLF_NET — m2m

in

out $o : \mathit{Talk}$

loc $t_1, t_2 : \mathit{Talk} \cup ?N;\ b_1, b_2 : ?N \cup \{\mathsf{act}\}$

$$\mathrm{CENTRE}[t_1, t_2 \rhd b_1, b_2] \oplus \mathrm{BASE}[b_1 \rhd t_1] \oplus \mathrm{BASE}[b_2 \rhd t_2] \oplus \mathrm{CAR}[t_1 \rhd o]$$

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 the communication between the base stations and the car is uni-directional. 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 $m \in \mathit{Talk} \subseteq D$ 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 $t_1 : \mathit{Talk} \cup ?N$

out $o : \mathit{Talk}$

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

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

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

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

out $t : \mathit{Talk} \cup ?N$

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

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

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

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 $?t_2$ is 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 synchronize the activity of the two base stations.

CENTRE — m2m

in $t_1, t_2 : \mathit{Talk} \cup ?N$

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

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

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

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

Note that despite of the massive use of 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 with 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. As mentioned in the introduction, we distinguish between p2p-communication with and without channel sharing. We concentrate on the first variant in Sections 4.1-4.5;

the second variant is treated in Section 4.6. To keep the presentation simple, we work in an untyped setting without tuple messages; when we come to the semantics of specifications, however, we assume the model is extended in accordance with Section 3.3.

4.1. Loss of Port Access

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 behavior of the modeling 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:

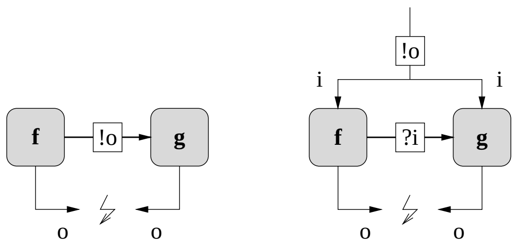

Fig. 5.

- $p = !o$ (see the network on the left hand-side of Figure 5); in that case, f and g may output simultaneously along $!o$;
- $p = ?i$ (see the network on the right hand-side of Figure 5); in that case, 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 interference without losing compositionality, 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 sets of active and passive 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 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 its retransmission 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.

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.

4.2. Privacy Preservation Redefined

We now explain how the privacy invariant described above is imposed formally. First of all, 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 in different channels. Such communication histories are *port-unique*.

Definition 10. (Port-uniqueness) A named communication history $\theta \in H$ is port-unique iff:

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

H_U is the set of all port unique communication histories in H. The merge component M preserves port-uniqueness if its two arguments are without occurrences of the same port within the same time unit. More precisely, if $pt(\theta)(t) \equiv \{p \in ?!N \mid \exists\, i \in N : p \in \theta(i)(t)\}$ we have

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

Definition 11. (Active and passive ports, redefined) For any $I, O, P \subseteq N$; $t \in Nat_+$; $\theta, \delta \in H_U$; aP_t and pP_t are defined recursively as follows:

$$\mathsf{aP}_1 \equiv ?I \cup !O, \qquad \mathsf{pP}_1 \equiv ?!P$$

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

where

$$\mathsf{rP}_t \equiv \bigcup_{?i \in \mathsf{aP}_t} \{p \mid p \in \overline{\mathsf{pP}_t \cup \mathsf{aP}_t} \cap \theta(i)(t)\}, \qquad \mathsf{hP}_t \equiv \{p, \widetilde{p} \mid p \in \mathsf{rP}_t \wedge \widetilde{p} \in \mathsf{aP}_t\}$$

$$\mathsf{sP}_t \equiv \bigcup_{!i \in \mathsf{aP}_t} \{p \mid p \in (\mathsf{pP}_t \cup \mathsf{aP}_t) \cap \delta(i)(t)\}, \quad \mathsf{gP}_t \equiv \{\widetilde{p} \mid p \in \mathsf{sP}_t \wedge p \in \mathsf{pP}_t\}$$

We define the sets of active and passive ports as follows:

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

$\mathsf{rP}_t, \mathsf{sP}_t, \mathsf{gP}_t$ and hP_t are the sets of received, sent, generated and to-be-hidden ports, respectively.

Definition 12. (Domain and range, redefined) For any $I, O, P \subseteq N$; $t \in Nat_+$; $\theta, \delta \in H_U$; let

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

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

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

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

Definition 13. (Privacy preservation, redefined) A function $f \in H_T \rightarrow H_T$ is privacy preserving with respect to $I, O, P \subseteq N$ iff:

$$\forall\, \theta \in H_T : f(\theta) = f(\mathsf{dmP}_{I,O,P}(\theta, f(\theta))) = \mathsf{rnP}_{I,O,P}(\theta, f(\theta))$$
$$[2mm] \forall\, \theta \in H_U : f(\theta) \in H_U$$

Note that we defined the functions f on H_T and not on H_U because we want that p2p-functions are a special case of m2m-functions.

We use $Mob_{p2p}(I,O,P)$ to denote the set of all strongly guarded functions that are privacy preserving in accordance with Definition 13 with respect to (I,O,P). In the sequel we refer to such functions as *p2p-functions*.

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

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

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

4.3. P2p-Components

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

Definition 14. (P2p-component) A p2p-component, with initial interface (I,O) and initial set of passive port-names P, is represented by a nonempty set of p2p-functions $F \subseteq Mob_{p2p}(I,O,P)$ that is closed in the following sense:

$$\forall f \in Mob_{p2p}(I,O,P) : (\forall \theta \in H_U : \exists f' \in F : f(\theta) = f'(\theta)) \Rightarrow f \in F$$

We use $Comp_{p2p}(I,O,P)$ to denote the set of all p2p-components. Note that by definition, any p2p-component is also m2m.

4.4. P2p-Composition

P2p-composition is defined similarly to the m2m-composition. However, feedback channels are in this case hidden both statically and dynamically.

Definition 15. (P2p-composition) Given two 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

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

$$P \equiv P_1 \cup P_2 \cup (I_1 \cap O_2) \cup (I_2 \cap O_1)$$

We define the p2p-composition of F_1 and F_2, as follows:

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

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 in order 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 about this fact.

Note that contrary to $\oplus$, we may use $\otimes$ to hide 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.

4.5. P2p-Specifications

The p2p-model is just a special case of the 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 same holds for composite specifications with the exception that there is no explicit hiding in the p2p-case. The semantics of an elementary p2p-specification S with body B is defined, as follows:

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

By H_{TU} we denote the type-correct sub-set of H_T. H_{AU} is the corresponding set of untimed typed communication histories. The semantics of a composite p2p-specification S with component specifications $S_1, \ldots, S_n$ is defined as follows:

$$[\![\, S \,]\!] \equiv [\![\, S_1 \,]\!] \otimes \ldots \otimes [\![\, S_n \,]\!]$$

Example 4. Mobile telephones — p2p-version:
The 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 telephones 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.

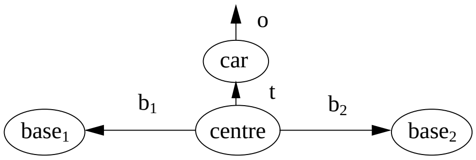

Fig. 6.

This network, whose initial configuration is illustrated by Figure 6, is specified by the composite p2p-specification TLF_NET. 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.

TLF_NET ═══ p2p

in

out $\quad o : Talk$

loc $\quad t : Talk;\ b_1, b_2 : !N$

CENTRE[$\triangleright\, b_1, b_2, t$] $\otimes$ BASE[$b_1\, \triangleright$] $\otimes$ BASE[$b_2\, \triangleright$] $\otimes$ CAR[$t \triangleright o$]

However, 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 $o : Talk$	
$\mathsf{out}(o) = \mathsf{in}(t)$	

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 under-specified by the oracle p.

BASE — p2p

in $b : !N$

out

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

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

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

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

CENTRE — p2p

in

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

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

where $\forall\, v \in H_{UA};\ q \in \mathsf{priv}$:

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

Both $!t$ and $!q$ are used repeatedly by the base stations, i.e., they are shared. However, they are 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 private 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 private. $\Box$

4.6. Restrictive P2p-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. This type of communication is better suited for theorem proving. One can safely assume that no interference ever occurs on any channel. Let $\theta\dagger_n$ denote the history obtained from θ by hiding the information along the channel n, i.e., forall m, n

$$\theta\dagger_n(n) \equiv \langle\rangle^\infty \quad \wedge \quad m \neq n \Rightarrow \theta\dagger_n(m) \equiv \theta(m)$$

Definition 16. (Restrictive p2p-component) A p2p-component F is a restrictive p2p-component if for all $f \in F$; $n, o \in N$; $t \in Nat_+$; $\theta \in H$:

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

Hence, restrictive p2p-communication guarantees that forwarded ports are not used for communication purposes. It is enough to restrict this until time unit t because the privacy constraint of p2p-components guarantees that they are not used afterwards. 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 $Comp_{r\_p2p}(I, O, P)$. 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 way. 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.

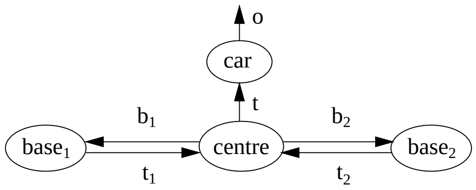

Fig. 7.

The network, whose initial configuration is illustrated by Figure 7, is specified as follows:

TLF_NET ══════════ r_p2p

in

out $o : Talk$

loc $b_1, b_2 : ?!N;\ t_1, t_2 : \{\mathsf{ok}\};\ t : Talk \cup ?N$

$\mathrm{CENTRE}[t_1, t_2 \rhd b_1, b_2, t] \otimes \mathrm{BASE}[b_1 \rhd t_1] \otimes \mathrm{BASE}[b_2 \rhd t_2] \otimes \mathrm{CAR}[t \rhd o]$

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 has to take care to send the read port to the car and the write port to the corresponding base.

$$
\begin{array}{|l}
\hline
\text{CENTRE} \hfill \mathsf{r\_p2p} \\
\mathsf{in} \quad t_1, t_2 : \{\mathsf{ok}\} \\
\mathsf{out} \quad b_1, b_2 : ?!N;\ t : \mathit{Talk} \cup ?N \\
\hline
\exists\, p \in \mathsf{priv}^{\infty} : \mathit{left}(t \,\&\, p)(\mathsf{in}) = \mathsf{out} \\
\mathsf{where}\ \forall\, v \in H_{UA};\ n \in \mathsf{priv};\ p \in \mathsf{priv}^{\infty} : \\
\quad \begin{array}{lcll}
\mathit{left}(n \,\&\, p)(v) & = & \{b_1 \mapsto !n,\ ?\mathsf{ft}.p\} & \&\ \ \mathit{wait\_l}(p)(v) \\
\mathit{wait\_l}(p)(\{t_1 \mapsto \mathsf{ok}\} \,\&\, v) & = & & \mathit{right}(p)(v) \\
\mathit{right}(n \,\&\, p)(v) & = & \{b_2 \mapsto !n,\ ?\mathsf{ft}.p\} & \&\ \ \mathit{wait\_r}(p)(v) \\
\mathit{wait\_r}(p)(\{t_1 \mapsto \mathsf{ok}\} \,\&\, v) & = & & \mathit{left}(p)(v)
\end{array} \\
\hline
\end{array}
$$

Note that the r_p2p constraint enforces (as desired) that all names in p are distinct, i.e., $\forall\, i, j \in \mathit{Nat}_+ : p(i) = p(j) \Rightarrow i = j$. Hence, we dont have to write this axiom explicitly.

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.

$$
\begin{array}{|l}
\hline
\text{BASE} \hfill \mathsf{r\_p2p} \\
\mathsf{in} \quad b : ?!N \\
\mathsf{out} \quad t : \{\mathsf{ok}\} \\
\hline
\exists\, p \in \{1,2\}^{\infty};\ m \in \mathit{Talk}^{\infty} : \mathit{idle}(p, m)(\mathsf{in}) = \mathsf{out} \\
\mathsf{where}\ \forall\, v \in H_{UA};\ c, e \in N;\ p \in \{1,2\}^{\infty};\ m \in \mathit{Talk}^{\infty} : \\
\quad \begin{array}{lcll}
\mathit{idle}(p, m)(\{b \mapsto !e\} \,\&\, v) & = & & \mathit{act}(e)(p, m)(v) \\
\mathit{act}(e)(1 \,\&\, p, m)(v) & = & \{e \mapsto \mathsf{ft}.m\} & \&\ \ \mathit{act}(e)(p, \mathsf{rt}.m)(v) \\
\mathit{act}(e)(2 \,\&\, p, m)(\{b \mapsto ?c\} \,\&\, v) & = & \{e \mapsto ?c,\ t \mapsto \mathsf{ok}\} & \&\ \ \mathit{idle}(p, m)(v)
\end{array} \\
\hline
\end{array}
$$

$\square$

5. Discussion

In this paper we defined a very simple 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 analyzed 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 simple specification formalism that supports the maintenance of the associated invariant. These formalisms allow us to write very high level specifications of mobile systems. Since object creation can be easily modelled with recursion, our formalisms also set the basis for high level specifications of object-oriented systems. The models of the above communication paradigms were defined in a stepwise manner from the most liberal m2m model to the most restrictive r_p2p model. We showed that each model is obtained from the previous one by strengthening the privacy constraints.

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 [AMST92] is an interesting area for future research. For example, we believe that our model can be used to give a denotational semantics for the asynchronous π-calculus. We also believe that the actor languages can be smoothly integrated within our formalism.

Our 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 nondeterminism differs from ours. In [Kok89], where a metric on relations is used, basically only bounded nondeterminism can be handled. In [Kok87], which is not based on metric spaces, an automaton is used to generate the behavior of basic agents. This guarantees the existence of fix-points. We use sets of strongly guarded functions for the same purpose. 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 real-time plays an important role. Second, in an untimed input/output model one cannot define and understand the privacy invariant.

The ideas on mobility of the first author originated in [Gro94]. In that work he defined a semantic model for mobile, deterministic data-flow networks. However, that model is higher-order and mobility is achieved by communicating channels and functions instead of ports. [Bro95] and [Gro94] give also an equational characterization of dynamic reconfiguration. Mobility in the more general framework of nondeterministic systems and where reconfiguration is achieved by sending ports was studied thereafter in [GS95, GS96a, GSB97, GS96b]. This article compares and unifies the models given in those papers. Moreover, it introduces for each communication paradigm a convenient specification formalism. This formalism was applied successfully to give a formal, high level specification of the kernel functionality of an operating system [HS96, Spi97]. In this specification, mobility is used to model resource allocation and recursion is used to model process creation. The m2m-model was also successfully used in [Hin] to give a formal semantics to the object-oriented extension of the ITU-T specification and description language SDL [OFMP$^+$94].

Acknowledgments

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 version of this paper and making helpful comments.

References

[AMST92] G. Agha, I. A. Mason, S. F. Smith, and C. L. Talcott. Towards a theory of actor computation. In *The Third International Conference on Concurrency Theory (CONCUR '92)*, volume 630 of *Springer LNCS*, pages 565–579. Springer Verlag, aug 1992.

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

[Bro95] M. Broy. Equations for describing dynamic nets of communicating systems. In *Proc. 5th COMPASS Workshop, Lecture Notes in Computer Science 906*, pages 170–187, 1995.

[BS97] M. Broy and K. Stølen. Focus on system development. Book manuscript, January 1997.

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

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

[FMS96] Marcelo Fiore, Eugenio Moggi, and Davide Sangiorgi. A fully-abstract model for the π-calculus (extended abstract). In *Eleventh Annual Symposium on Logic in Computer Science (LICS) (New Brunswick, New Jersey)*, pages 43–54. IEEE, Computer Society Press, July 1996.

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

[GS95] R. Grosu and K. Stølen. A Denotational Model for Mobile Point-to-Point Dataflow Networks. Technical Report SFB 342/14/95 A, Technische Universität München, 1995.

[GS96a] R. Grosu and K. Stølen. A Model for Mobile Point-to-Point Data-flow Networks without Channel Sharing . In Martin Wirsing and Maurice Nivat, editors, *Proceedings of the 5th International Conference on Algebraic Methodology and Software Technology, AMAST'96, Munich, Germany*, pages 505–519. Lecture Notes in Computer Science 1101, 1996.

[GS96b] R. Grosu and K. Stølen. A denotational model for mobile many-to-many data-flow networks. Technical Report TUM-I9622, Technische Universität München, 1996.

[GSB97] R. Grosu, K. Stølen, and M. Broy. A model for mobile point-to-point data-flow networks with channel sharing. Technical Report SFB 342/17/97A, Technische Universität München, 1997.

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

[Hin] U. Hinkel. Formale Fundierung und Verifikationsmethodik für SDL. Dissertation.

[HS96] U. Hinkel and K. Spies. Anleitung zur Spezifikation von mobilen, dynamischen Focus-Netzen. TUM-I 9639, Technische Universität München, 1996.

[JJ95] L. J. Jagadeesan and R. Jagadeesan. Causality and true concurrency: A data-flow analysis of the pi-calculus. *Lecture Notes in Computer Science*, 936:277–??, 1995.

[Jon90] C. B. Jones. *Systematic Software Development Using VDM, Second Edition*. Prentice-Hall, 1990.

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

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

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

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

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

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

[OFMP+94] A. Olsen, O. Færgemand, B. Møller-Pedersen, R. Reed, and J. R. W. Smith. *Systems Engineering Using SDL-92*. Elsevier Science, North-Holland, 1994.

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

[Spi97] K. Spies. Modellierung und methodische Entwicklung von Betriebssystemkonzepten. PhD in progress, June 1997.

[Sta96] Ian Stark. A fully abstract domain model for the pi-calculus. In *Eleventh Annual Symposium on Logic in Computer Science (LICS) (New Brunswick, New Jersey)*, pages 36–42. IEEE, Computer Society Press, July 1996.

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

A. Metrics on Streams and Stream Tuples

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

Definition 17. (Metric of streams) The metric of streams (E^∞, d) over a discrete metric (E, ρ) is defined, as follows:

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

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

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

Proof See [Eng77]. □

Definition 18. (Metric of named stream tuples) The metric of *named stream tuples* $(I \to E^\infty, d)$ over a countable set of names I and discrete metric (E, ρ) is defined, as follows:

$$\forall\, \theta, \varphi \in I \to E^\infty : d(\theta, \varphi) = \mathit{inf}\{2^{-t} \mid \theta{\downarrow_t} = \varphi{\downarrow_t}\}$$

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

Proof The metric is equivalent to the Cartesian product metric $\times_{i \in I} E^\infty$ which is complete because E^∞ is complete (see [Eng77]). □

B. Proofs — 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 have 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.

Induction hypothesis:

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

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

To simplify the notation, we define:

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

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

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

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

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

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

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

By definition of dmM and rnM:

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

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

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

inside this union. It is enough to show that

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

under the assumption that $c \notin \mathsf{aM}_n$ and $c \in \overline{\mathsf{pM}_n}$. This follows trivially 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 taken over $?i \in \mathsf{aM}_n = \mathsf{aM}'_n = \mathsf{aM}''_n$. As a consequence

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

inside this union. It is enough to show that

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

under the assumption that $c \in \mathsf{pM}_n$. This follows trivially 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 $\theta(i)(n)$ inside the definition of dmM and $\mathsf{rnM}(\theta, \delta)(i)(n)$ simplifies to $\delta(i)(n)$ inside the definition of rnM. This immediately proves the theorem.

□

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

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

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

The function f is well defined and strongly guarded because g is strongly guarded and dmM and rnM are weakly guarded. The privacy preserving property of f is proved by 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:

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

By Theorem 4 and definition of f we have that:

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

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 $\delta = \gamma$. 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

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

□

This completes the proof of privacy preservation. □

Theorem 6. $F_1 \odot F_2$ is a closed, strongly-guarded component if F_1 and F_2 are strongly-guarded components.

Proof Since F_1 and F_2 are closed components and M is not empty we may find functions $f_1 \in F_1, f_2 \in F_2$ and $m_1, m_2, m_3 \in \mathsf{M}$. Based on these functions we construct a function f which is strongly guarded and satisfies the recursive equation in the definition of $F_1 \odot F_2$. Let g be defined as follows:

$$g \in (H \times H) \times H \to H \times H$$

$$g((\varphi, \psi), \theta) = (f_1(m_1(\vartheta, \psi)), f_2(m_2(\vartheta, \varphi))$$

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

$$f \in H \to H$$

$$f(\theta) = m_3(\varphi, \psi) \quad \mathsf{where} \quad (\varphi, \psi) = (\mu\ g)(\theta)$$

Finally, since $\exists f_1, f_2, m_1, m_2, m_3 : \forall \theta : P$ implies $\forall \theta : \exists f_1, f_2, m_1, m_2, m_3 : P$ it follows that $f \in F_1 \odot F_2$. To see that $F_1 \odot F_2$ is closed, assume that

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

Together with the definition of $\odot$ it follows that for any θ there are $f' \in F_1 \odot F_2$, $f_1 \in F_1, f_2 \in F_2$ and $m_1, m_2, m_3 \in \mathsf{M}$ such that:

$$f(\theta) = f'(\theta) = m_3(\varphi, \psi) \ \mathsf{where}$$

$$\varphi = f_1(m_1(\theta, \psi)), \quad \psi = f_2(m_2(\theta, \varphi))$$

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

Theorem 7. $F_1 \oplus F_2$ is an m2m-component if F_1 and F_2 are m2m-components.

Proof Follows trivially from Theorems 5 and 6. □

Theorem 8. $\nu x : F$ is an m2m-component.

Proof Follows trivially from Theorem 5. □

C. Proofs — 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 the one for the m2m-case. □

Theorem 10. The functions dmP and rnP have the following properties:

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

Proof The proof is is identical to the one for the m2m-case. □

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

Proof The proof of privacy preservation is identical to the one for the m2m-case. The only difference is 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 p2p-component if F_1 and F_2 are p2p-components.

Proof That $F_1 \otimes F_2$ is well defined, closed 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. In order to prove this, we have to show that for each n:

$$pt(\varphi)(n) \cap pt(\vartheta)(n) = pt(\psi)(n) \cap pt(\vartheta)(n) = pt(\varphi)(n) \cap pt(\psi)(n) = \{\}$$

where $\vartheta = \mathsf{dmP}(\theta, m_3(\varphi,\psi))$. The proof is by induction and uses a stronger induction hypothesis, the one given in the following lemma. Let

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

Lemma 3. If $\theta \in H_U$ then $\varphi, \psi, m_1(\vartheta,\psi), m_2(\vartheta,\varphi) \in H_U$ and for all n

$$\begin{aligned}
&\mathsf{aP}^1_n \cap \mathsf{pP}^1_n = \mathsf{aP}^2_n \cap \mathsf{pP}^2_n = \mathsf{aP}_n \cap \mathsf{pP}_n = \{\}\\
&(\mathsf{pP}^1_n \cup \mathsf{aP}^1_n) \cap (\mathsf{pP}^2_n \cup \mathsf{aP}^2_n) = \{\},\\
&\mathsf{aP}_n = (\mathsf{aP}^1_n \setminus \widetilde{\mathsf{aP}^2_n}) \cup (\mathsf{aP}^2_n \setminus \widetilde{\mathsf{aP}^1_n})\\
&\mathsf{pP}_n = (\mathsf{aP}^1_n \cap \widetilde{\mathsf{aP}^2_n}) \cup (\mathsf{aP}^2_n \cap \widetilde{\mathsf{aP}^1_n}) \cup \mathsf{pP}^1_n \cup \mathsf{pP}^2_n
\end{aligned}$$

Proof Suppose the above equalities are our induction hypothesis.
Base case:

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

These values clearly satisfy the above equations for $n = 1$. Together with the port-uniqueness preserving property of f_1 and f_2 they also assure the port uniqueness of

$\varphi, \psi, m_1(\vartheta, \psi), m_2(\vartheta, \varphi)$ and $m_3(\varphi, \psi)$ for $n = 1$.
Induction step: Expanding the definitions of aP and pP we obtain

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

where

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

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

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

We do now a case analysis over the terms of the above expressions. Each term denotes a particular form of input or output.

1. External Input: $?i \in \mathsf{aP}^1_n \cap \mathsf{aP}_n$

By the induction hypothesis $?i \notin \widetilde{\mathsf{aP}^2_n}$. As a consequence $(m_1(\vartheta, \psi))(i)(n) = \vartheta(i)(n)$. Suppose $c \in \vartheta(i)(n)$. Clearly $c \in \overline{\mathsf{aP}_n \cup \mathsf{pP}_n}$. There are two cases $\widetilde{c} \notin \mathsf{aP}^1_n$ and $\widetilde{c} \in \mathsf{aP}^1_n$:

$\underline{\widetilde{c} \notin \mathsf{aP}^1_n}$

$c \in \mathsf{rP}^1_n$,	$\{$by definition$\}$
$c \notin \mathsf{hP}^1_n$,	$\{$by definition$\}$
$c \notin \mathsf{sP}^1_n$,	$\{(\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\}$

Hence $c \in \mathsf{aP}^1_{n+1}$ and $c \notin \mathsf{pP}^1_{n+1}$.

$c \notin \mathsf{aP}^2_n$,	$\{\mathsf{aP}^2_n \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\}$
$c \notin \mathsf{rP}^2_n$,	$\{u(\theta),\ (\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\}$
$c \notin \mathsf{sP}^2_n$,	$\{(\mathsf{aP}^2_n \cup \mathsf{pP}^2_n) \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\}$

Hence $c \notin \mathsf{aP}^2_{n+1}$ and $c \notin \mathsf{pP}^2_{n+1}$.

$c \in \mathsf{rP}_n$,	$\{$by definition$\}$
$c \notin \mathsf{hP}_n$,	$\{a.\ \widetilde{c} \notin \mathsf{aP}^2_n\}$
$c \in \mathsf{hP}_n$,	$\{b.\ \widetilde{c} \in \mathsf{aP}^2_n\}$
$c \notin \mathsf{sP}_n$,	$\{(\mathsf{aP}_n \cup \mathsf{pP}_n) \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\}$

Hence $c \in \mathsf{aP}_{n+1}$ and $c \notin \mathsf{pP}_{n+1}$ in case a and $c \notin \mathsf{aP}_{n+1}$ and $c \in \mathsf{pP}_{n+1}$ in case b.

$\underline{\widetilde{c} \in \mathsf{aP}^1_n}$

$c \in \mathsf{rP}^1_n$	$\{$by definition$\}$
$c \in \mathsf{hP}^1_n$	$\{$by definition$\}$
$c \notin \mathsf{sP}^1_n$	$\{(\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\}$

Hence $c \notin \mathsf{aP}^1_{n+1}$ and $c \in \mathsf{pP}^1_{n+1}$. For f_2 nothing changes. Hence $c \notin \mathsf{aP}^2_{n+1}$ and $c \notin \mathsf{pP}^2_{n+1}$.

$$\begin{array}{ll} c \in \mathsf{rP}_n & \{\text{by definition}\} \\ c \in \mathsf{hP}_n & \{\text{by definition}\} \\ c \notin \mathsf{sP}_n & \{(\mathsf{aP}_n \cup \mathsf{pP}_n) \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\} \end{array}$$

Hence $c \notin \mathsf{aP}_{n+1}$ and $c \in \mathsf{pP}_{n+1}$.

In both cases all the above equations are satisfied. Moreover, the port-uniqueness of $m_1(\vartheta,\psi)$ and $m_2(\vartheta,\varphi)$ at time n, and the port-uniqueness preserving property of f_1 and f_2 imply the port-uniqueness of φ and ψ at time $n+1$. Then the above disjointness equations imply the uniqueness of $m_1(\vartheta,\psi)$, $m_2(\vartheta,\varphi)$ and $m_3(\varphi,\psi)$ at time $n+1$.

2. Internal Input: $?i \in \mathsf{aP}^1_n \cap \widetilde{\mathsf{aP}^2_n}$

By induction hypothesis $?i \notin \mathsf{aP}_n$. Hence $(m_1(\vartheta,\psi))(i)(n) = \psi(i)(n)$. Suppose $c \in \psi(i)(n)$. Then there are two disjoint cases: $c \in \mathsf{aP}^2_n$ or $c \in \mathsf{pP}^2_n$.

$\underline{c \in \mathsf{aP}^2_n}$

$$\begin{array}{ll} c \in \mathsf{rP}^1_n, & \{\text{by definition}\} \\ c \notin \mathsf{hP}^1_n & \{a.\ \widetilde{c} \notin \mathsf{aP}^1_n\} \\ c \in \mathsf{hP}^1_n & \{b.\ \widetilde{c} \in \mathsf{aP}^1_n\} \\ c \notin \mathsf{sP}^1_n, & \{(\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) \cap \mathsf{aP}^2_n = \{\}\} \end{array}$$

Hence $c \in \mathsf{aP}^1_{n+1}$ and $c \notin \mathsf{pP}^1_{n+1}$ in case a or $c \notin \mathsf{aP}^1_{n+1}$ and $c \in \mathsf{pP}^1_{n+1}$ in case b.

$$\begin{array}{ll} c \notin \mathsf{rP}^2_n, & \{\mathsf{aP}^2_n \cap (\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) = \mathsf{aP}^2_n \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\} \\ c \in \mathsf{sP}^2_n, & \{\text{by definition}\} \\ c \notin \mathsf{gP}^2_n, & \{\mathsf{aP}^2_n \cap \mathsf{pP}^2_n = \{\}\} \end{array}$$

Hence $c \notin \mathsf{aP}^2_{n+1}$ and $c \notin \mathsf{pP}^2_{n+1}$.

$$\begin{array}{ll} c \in \mathsf{aP}_n \wedge c \notin \mathsf{pP}_n, & \{a.\ \widetilde{c} \notin \mathsf{aP}^1_n\} \\ c \notin \mathsf{aP}_n \wedge c \in \mathsf{pP}_n & \{b.\ \widetilde{c} \in \mathsf{aP}^1_n\} \\ c \notin \mathsf{rP}_n, & \{\mathsf{aP}^2_n \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\} \\ c \notin \mathsf{sP}_n, & \{u(\psi),\ (\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) \cap \mathsf{aP}^2_n = \{\}\} \end{array}$$

Hence $c \in \mathsf{aP}_{n+1}$ and $c \notin \mathsf{pP}_{n+1}$ in case a or $c \notin \mathsf{aP}_{n+1}$ and $c \in \mathsf{pP}_{n+1}$ in case b.

$\underline{c \in \mathsf{pP}^2_n}$

$$\begin{array}{ll} c \in \mathsf{rP}^1_n, & \{\text{by definition}\} \\ c \notin \mathsf{hP}^1_n & \{c, \widetilde{c} \notin \mathsf{aP}^1_n\} \\ c \notin \mathsf{sP}^1_n, & \{(\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) \cap \mathsf{pP}^2_n = \{\}\} \end{array}$$

Hence $c \in \mathsf{aP}^1_{n+1}$ and $c \notin \mathsf{pP}^1_{n+1}$.

$$\begin{array}{ll} c \notin \mathsf{rP}^2_n, & \{\mathsf{pP}^2_n \cap (\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) = \mathsf{pP}^2_n \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\} \\ c \in \mathsf{sP}^2_n, & \{\text{by definition}\} \\ \widetilde{c} \in \mathsf{gP}^2_n, & \{\text{by definition}\} \end{array}$$

Hence $c \notin \mathsf{aP}^2_{n+1}$ and $c \notin \mathsf{pP}^2_{n+1}$ and $\widetilde{c} \in \mathsf{aP}^2_n$.

$$\begin{array}{ll} c \notin \mathsf{aP}_n & \{\mathsf{pP}^2_n \cap \mathsf{aP}_n = \{\}\} \\ c \in \mathsf{pP}_n & \{\text{induction hypothesis}\} \\ c \notin \mathsf{rP}_n, & \{\mathsf{pP}^2_n \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\} \\ c \notin \mathsf{sP}_n, & \{u(\psi),\ (\mathsf{aP}^1_n \cup \mathsf{pP}^1_n) \cap \mathsf{pP}^2_n = \{\}\} \end{array}$$

Hence $c \notin \mathsf{aP}_{n+1}$ and $c \in \mathsf{pP}_{n+1}$.
It is easy to show that all the above equations hold. Moreover, a similar argument as before, proves the port-uniqueness.

3. External Output: $!i \in \mathsf{aP}^1_n \cap \mathsf{aP}_n$

By the induction hypothesis $!i \notin \widetilde{\mathsf{aP}^2_n}$. The only interesting case is if $c \in \varphi(i)(n)$ is passive and its complement is not sent, i.e., if $c \in \mathsf{pP}^1_n$ and $\widetilde{c} \notin \varphi(i)(n)$.

$$\begin{array}{ll} c \in \mathsf{sP}^1_n, & \{\text{by definition}\} \\ \widetilde{c} \in \mathsf{gP}^1_n, & \{\text{by definition}\} \end{array}$$

Hence $\widetilde{c} \in \mathsf{aP}^1_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pP}^1_{n+1}$.

$$\begin{array}{ll} c \notin \mathsf{aP}^2_n & \{\mathsf{pP}^1_n \cap \mathsf{aP}^2_n = \{\}\} \\ c, \widetilde{c} \notin \mathsf{rP}^2_n, & \{u(\varphi),\ \mathsf{pP}^1_n \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\} \\ c, \widetilde{c} \notin \mathsf{sP}^2_n, & \{\mathsf{pP}^1_n \cap (\mathsf{aP}^2_n \cup \mathsf{pP}^2_n) = \{\}\} \end{array}$$

Hence $c, \widetilde{c} \notin \mathsf{aP}^2_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pP}^2_{n+1}$.

$$\begin{array}{ll} c \in \mathsf{sP}_n, & \{\text{by definition}\} \\ \widetilde{c} \in \mathsf{gP}_n, & \{\text{by definition}\} \end{array}$$

Hence $\widetilde{c} \in \mathsf{aP}^1_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pP}^1_{n+1}$.

It is easy to show that all the above equations hold. Moreover, a similar argument as before, proves the port-uniqueness.

4. Internal Output: $!i \in \mathsf{aP}^1_n \cap \widetilde{\mathsf{aP}^2_n}$

By the induction hypothesis $!i \notin \mathsf{aP}_n$. The only interesting case is if $c \in \varphi(i)(n)$ is passive and its complement is not sent, i.e., if $c \in \mathsf{pP}^1_n$ and $\widetilde{c} \notin \varphi(i)(n)$.

$$\begin{array}{ll} c \in \mathsf{sP}^1_n, & \{\text{by definition}\} \\ \widetilde{c} \in \mathsf{gP}^1_n, & \{\text{by definition}\} \end{array}$$

Hence $\widetilde{c} \in \mathsf{aP}^1_{n+1}$, $c \notin \mathsf{aP}^1_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pP}^1_{n+1}$.

$$\begin{array}{ll} c \in \mathsf{rP}^2_n, & \{\text{by definition}\} \\ \widetilde{c} \notin \mathsf{rP}^2_n, & \{u(\varphi),\ \mathsf{pP}^1_n \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\} \\ \widetilde{c} \notin \mathsf{sP}^2_n, & \{\mathsf{pP}^1_n \cap (\mathsf{aP}^2_n \cup \mathsf{pP}^2_n) = \{\}\} \end{array}$$

Hence $c \in \mathsf{aP}^2_{n+1}$, $\widetilde{c} \notin \mathsf{aP}^2_{n+1}$ and $c, \widetilde{c} \notin \mathsf{pP}^2_{n+1}$.

$$\begin{array}{ll} c \notin \mathsf{aP}_n & \{\mathsf{pP}^1_n \cap \mathsf{aP}_n = \{\}\} \\ c \in \mathsf{pP}_n & \{\text{induction hypothesis}\} \\ c \notin \mathsf{rP}_n, & \{\mathsf{pP}^1_n \cap \overline{\mathsf{aP}_n \cup \mathsf{pP}_n} = \{\}\} \\ c \notin \mathsf{sP}_n, & \{u(\psi),\ (\mathsf{aP}^2_n \cup \mathsf{pP}^2_n) \cap \mathsf{pP}^1_n = \{\}\} \end{array}$$

Hence $c \notin \mathsf{aP}_{n+1}$ and $c \in \mathsf{pP}_{n+1}$.

It is easy to show that all the above equations hold. Moreover, a similar argument as before, proves the port-uniqueness. This also completes all the cases for f_1. A similar argument applies for f_2. □

As a consequence of the above lemma, ϑ, φ and ψ have disjoint domains, are port unique and contain disjoint sets of ports. Consequently, f preserves port-uniqueness. □

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

Proof Suppose $f \in F_1 \otimes F_2$ and $?n \in f(\theta)(o)(t)$. Then there are restrictive p2p-functions $f_1 \in F_1$ and $f_2 \in F_2$ sucht that

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

where ϑ is the restriction of θ with respect to the domain of f. Suppose that

$$?n \in f_1(m_1(\vartheta, \psi))(o)(t)$$

Since f_1 is a restrictive p2p-function it follows that

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

Moreover, since $?n$ belonged to f_1 and $F_1 \otimes F_2$ is a p2p-component, it follows that $?n$ is not among the ports of f_2. As a consequence, since f_2 is p2p

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

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

Suppose that $!n \in f(\theta)(o)(t)$. Then

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

Suppose that

$$!n \in f_1(m_1(\vartheta, \psi))(o)(t)$$

Since f_1 is a restrictive p2p-function it follows that

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

Moreover, since $!n$ belonged to f_1 and $F_1 \otimes F_2$ is a p2p-component, it follows that $!n$ is not among the ports of f_2. As a consequence, since f_2 is p2p

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

Hence $f(\theta)\downarrow_t = (f(\theta)\dagger_n)\downarrow_t$. If $!n$ belongs to f_2 the proof is similar. This completes the proof. $\square$

D. Proofs — P2p Implies M2m

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

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

Proof The proof is by induction. We split it into the following three lemmas.

Lemma 4.

$$\mathsf{aP}_n \subseteq \mathsf{aM}_n$$

Proof
Base Case: $\mathsf{aP}_1 = ?I \cup !O \ \subseteq \mathsf{aM}_1$.

Induction Step: According to the definition of aP_{n+1} and aM_{n+1} we have the following cases:

(a) $p \ \in \ \mathsf{aP}_n$. Then by induction hypothesis $p \in \mathsf{aM}_n \ \subseteq \ \mathsf{aM}_{n+1}$.

(b) $p \in \mathsf{rP}_n$. Since $\mathsf{aP}_n \subseteq \mathsf{aM}_n$ we only have to prove that if $p \notin \mathsf{pP}_n \cup \mathsf{aP}_n$ then it is also the case that $p \in \mathsf{aM}_{n+1}$. By using the induction hypothesis (3), we have the following three cases:

$$b.1 \quad p \in \mathsf{aM}_n \overset{def}{\Rightarrow} p \in \mathsf{aM}_{n+1}$$

$$b.2 \quad p \in \mathsf{pM}_n \Rightarrow \widetilde{p} \in \widetilde{\mathsf{pM}_n} \overset{(2)}{\Rightarrow} \widetilde{p} \in \mathsf{aP}_n \overset{def}{\Rightarrow} p \in \mathsf{hP}_n \overset{def}{\Rightarrow} p \notin \mathsf{aP}_{n+1}$$

$$b.3 \quad p \in \overline{\mathsf{aM}_n \cup \mathsf{pM}_n} \overset{def}{\Rightarrow} p \in \mathsf{rM}_n \ \subseteq \mathsf{aM}_{n+1}$$

(c) $p \in \mathsf{gP}_n$. Since $\mathsf{aP}_n \subseteq \mathsf{aM}_n$ we only have to prove that if $p \in \mathsf{pP}_n$ then it is also the case that $p \in \mathsf{aM}_{n+1}$. By (3) we have that $p \in \mathsf{pM}_n \cup \mathsf{aM}_n$. Hence either $p \in \mathsf{aM}_n \subseteq \mathsf{aM}_{n+1}$ or $p \in \mathsf{gM}_n \subseteq \mathsf{aM}_{n+1}$. □

Lemma 5.

$$\widetilde{\mathsf{pM}}_n \subseteq \mathsf{pP}_n \cup \mathsf{aP}_n$$

Proof
Base Case: $\widetilde{\mathsf{pM}_1} \ = \ ?!P \ \subseteq \ \mathsf{pP}_1 \ \cup \ \mathsf{aP}_1$.

Induction Step: According to the definition $\mathsf{pM}_{n+1} = \mathsf{pM}_n \setminus \widetilde{\mathsf{sM}_n}$. Hence:

$$\widetilde{\mathsf{pM}_{n+1}} \overset{def}{=} \widetilde{\mathsf{pM}_n} \setminus \mathsf{sM}_n \overset{(1,2)}{\subseteq} (\mathsf{aP}_n \cup \mathsf{pP}_n) \setminus \mathsf{sP}_n \overset{def}{\subseteq} \mathsf{aP}_{n+1} \ \cup \ \mathsf{pP}_{n+1}$$

□

Lemma 6.

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

Proof
Base Case: $\mathsf{pP}_1 \ \cup \ \mathsf{aP}_1 \ = \ ?!P \ \cup \ ?I \ \cup \ !O \ \subseteq \ \mathsf{pM}_1 \ \cup \ \mathsf{aM}_1$.

Induction Step: According to the definition:

$$\begin{array}{lll} \mathsf{pP}_{n+1} \ \cup \ \mathsf{aP}_{n+1} & = & (\mathsf{pP}_n \ \cup \ \mathsf{aP}_n \ \cup \ \mathsf{rP}_n) \ \setminus \ \mathsf{sP}_n \\ \mathsf{pM}_{n+1} \ \cup \ \mathsf{aM}_{n+1} & = & \mathsf{pM}_n \ \cup \ \mathsf{aM}_n \ \cup \ \mathsf{rM}_n \end{array}$$

The lemma follows immediately, since $\mathsf{rP}_n \subseteq \mathsf{aM}_{n+1}$ as proved in the first lemma. $\Box$

This completes the proof of the theorem. $\Box$

The above theorem holds even for stronger requirements for δ.

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

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

Proof The proof is almost the same as the proof of Theorem 14. However, it also uses the results of Theorem 14. $\Box$

Theorem 16. The functions dmP and dmM have 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 quite similar to the proofs of the Theorems 4 and 10 and 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 active and passive ports in the may-to-many and in the point-to-point 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 that $\mathsf{pP}_n = \mathsf{pP}'_n$.

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 of aP and pP:

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

By induction hypothesis, $\mathsf{aP}_n = \mathsf{aP}_n$ and $\mathsf{pP}_n = \mathsf{pP}_n$. As a consequence:

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

Moreover

$$\mathsf{hP}_n = \{p \mid p \in \mathsf{rP}_n \wedge \widetilde{p} \in \mathsf{aP}_n\}, \quad \mathsf{hP}'_n = \{p \mid p \in \mathsf{rP}'_n \wedge \widetilde{p} \in \mathsf{aP}_n\}$$

and

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

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

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

Now, by Theorem 14 it follows that $\mathsf{dmM}(\theta,\delta)(i)(n) = \theta(i)(n)$ inside rP'_n. As a consequence $\mathsf{aP}_{n+1} = \mathsf{aP}'_{n+1}$ and $\mathsf{pP}_{n+1} = \mathsf{pP}'_{n+1}$. This immediately proves the theorem. $\Box$

Theorem 17. The functions rnM and rnP have 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:

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

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. □

Now we are ready to prove the main theorem of this appendix.

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

Proof We split the proof in two lemmas.

Lemma 7.

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

Proof

$$\mathsf{rnM}(\theta, f(\theta)) \stackrel{hyp}{=} \mathsf{rnM}(\theta, \mathsf{rnP}(\theta, f(\theta))) \stackrel{Thm\ 17}{=} \mathsf{rnP}(\theta, f(\theta)) \stackrel{hyp}{=} f(\theta)$$

□

Lemma 8.

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

Proof By hypothesis $f(\theta)$ is defined as follows:

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

Since $f(\theta)$ verifies the recursive equation in δ (it is point-to-point), $f(\theta) = \delta$ is the unique fix-point of the above recursive equation. Now:

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

Now we show that δ satisfies the recursive equation in γ:

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

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. □