History-Dependent Automata: An Introduction

Ugo Montanari[1] and Marco Pistore[2]

[1] University of Trento, Italy
marco.pistore@unitn.it
[2] University of Pisa, Italy
ugo@di.unipi.it

Abstract. In this paper we give an overview of History Dependent Automata, an extension of ordinary automata that overcomes their limitations in dealing with named calculi. In a named calculus, the observations labelling the transitions of a system may contain names which represent features such as communication channels, node identifiers, or the locations of the system. An example of named calculus is π-calculus, which has the ability of sending channel names as messages and thus of dynamically reconfiguring process acquaintances and of modeling agents and code mobility. We show that History-Dependent Automata allow for a compact representation of π-calculus processes which is suitable both for theoretical investigations and for practical purposes such as verification.

1 Introduction

In the context of process calculi (e.g., Milner's CCS [Mil89]), *automata* (or *labelled transition systems*) are often used as operational models. They allow for a simple representation of process behavior, and many concepts and theoretical results for these process calculi are independent from the particular syntax of the languages and can be formulated directly on automata. In particular, this is true for the *behavioral equivalences* and preorders which have been defined for these languages, like bisimulation equivalence [Mil89, Par80]: in fact they take into account only the labelled actions a process can perform. Automata are also important from an algorithmic point of view: efficient and practical techniques and tools for verification [IP96, Mad92] have been developed for *finite-state* automata. Finite state verification is successful here, differently than in ordinary programming, since the control part and the data part of protocols and hardware components can be often cleanly separated, and the control part is usually both quite complex and finite state. Particularly interesting is also the possibility to associate to each automaton — and, consequently, to each process — a *minimal realization*, i.e., a minimal automaton which is equivalent to the original one. This is important both from a theoretical point of view — equivalent systems give rise to the same (up to isomorphism) minimal realization — and from a practical point of view — smaller state spaces can be obtained.

This ideal situation, however, does not apply to all process calculi. In the case of *named calculi*, in particular, infinite-state transition systems are generated instead, also

M. Bernardo and A. Bogliolo (Eds.): SFM-Moby 2005, LNCS 3465, pp. 1–28, 2005.

by very simple processes. In a *named calculus*, the observations labelling the transitions of a system may contain names which are used to identify different features of the modeled system, such as the communication channels, the agents participating to the system, or the locations describing the spatial structure of the system. A quite interesting example of named calculus is π-calculus [MPW92, Mil93]. It has the ability of sending channel names as messages and thus of dynamically reconfiguring process acquaintances. More importantly, π-calculus names can model objects (in the sense of object oriented programming [Wal95]) and name sending thus models higher order communication and mobile code [San93b].

The operational semantics of π-calculus is given via a labelled transition system. However labelled transition systems are not fully adequate to deal with the peculiar features of the calculus and complications occur in the creation of new channels. Consider process $p = (\nu y)\,\bar{x}y.y(z).\mathbf{0}$. Channel y is initially a local channel for the process (prefix $(\nu y)\_$ is the operator for scope restriction) and no global communication can occur on it. Action $\bar{x}y$, however, which corresponds to the output of name y on the global channel x, makes name y known also outside the process; after the output has taken place, channel y can be used for further communications, and, in fact, y is used in $y(z).\mathbf{0}$ as the channel for an input transition: so the communication of a restricted name creates a new public channel for the process. The creation of this new channel is represented in the ordinary semantics of the π-calculus by means of an infinite bunch of transitions of the form $p \xrightarrow{\bar{x}(w)} w(z).\mathbf{0}$, where w is any name that is not already in use (i.e., $w \neq x$ in our example, since x is the only name in use by p; notice that $w = y$ is just a particular case). This way to represent the creation of new names has some disadvantages: first of all, also very simple π-calculus processes, like p, give rise to infinite-state and infinite-branching transition systems. Moreover, equivalent processes do not necessarily have the same sets of channel names; so, there are processes q equivalent to p which cannot use y as the name for the newly created channel. Special rules are needed in the definition of bisimulation to take care of this problem and, as a consequence, standard theories and algorithms do not apply to π-calculus.

The ideal situation of ordinary automata can (at least in part) be recovered also in the field of named calculi, by introducing a new operational model which is adequate to deal with these languages, and by extending to this new model (part of) the classical theory for ordinary automata. As model we propose the *history-dependent automata* (*HD-automata* in brief). As ordinary automata, they are composed of states and of transitions between states. To deal with the peculiar problems of named calculi, however, states and transitions are enriched with sets of local names: in particular, each transition can refer to the names associated to its source state but can also generate new names, which can then appear in the destination state. In this manner, the names are not global and static, as in ordinary labelled transition systems, but they are explicitly represented within states and transitions and can be dynamically created.

This explicit representation of names permits an adequate representation of the behavior of named processes. In particular, π-calculus processes can be translated into HD-automata and a first sign of the adequacy of HD-automata for dealing with π-calculus is that a large class of *finitary* π-calculus processes can be represented by finite-state HD-automata. We also give a general definition of bisimulation for HD-automata.

An important result is that this general bisimulation equates the HD-automata obtained from two π-calculus processes if and only if the processes are bisimilar according to the ordinary π-calculus bisimilarity relation. The most interesting result on HD-automata is that they can be minimized. It is possible to associate to each HD-automaton a minimal realization, namely a minimal HD-automaton that is bisimilar to the initial one. As in the case of ordinary automata, this possibility is important from a theoretical but also from a practical point of view.

In this paper we give an introduction to HD-automata. Some of the basic results on ordinary automata and an overview of the π-calculus are briefly presented in Section 2. Section 3 introduces HD-automata, defines bisimulation on HD-automata, and presents the translation of π-calculus processes to HD-automata. Section 4 describes how HD-automata can be minimized by taking into account symmetries on the names enriching states and transitions. Finally, in Section 5 we propose some concluding remarks. Further results on HD-automata (as well as the proofs of the results that we present in this paper) can be found in [MP98b, MP98a, MP99, MP00].

2 Background

2.1 Ordinary Automata

Automata have been defined in a large variety of manners. We choose the following definition since it is very natural and since, as we will see, it can be easily modified to define HD-automata.

Definition 1 (ordinary automata). *An* automaton $\mathcal{A}$ *is defined by:*

- *a set* L *of* labels*;*
- *a set* Q *of* states*;*
- *a set* T *of* transitions*;*
- *two functions* $s, d : T \to Q$ *that associate a* source *and a* destination *state to each transition;*
- *a function* $o : T \to L$ *which associates a label to each transition;*
- *an* initial state $q_0 \in Q$.

Given a transition $t \in T$*, we write* $t : q \xrightarrow{l} q'$ *if* $s(t) = q$*,* $d(t) = q'$ *and* $o(t) = l$.

Notation 2. *To represent the components of an automaton we will use the name of the automaton as subscript; so, for instance,* $Q_{\mathcal{B}}$ *are the states of automaton* $\mathcal{B}$ *and* $d_{\mathcal{B}}$ *is its destination function. In the case of automaton* $\mathcal{A}_x$*, we will simply write* Q_x *and* d_x *rather than* $Q_{\mathcal{A}_x}$ *and* $d_{\mathcal{A}_x}$*. Moreover, the subscripts are omitted whenever there is no ambiguity on the referred automaton.*
Similar notations are also used for the other structures we define in the paper.

Often *labelled transition systems* are used as operational models in concurrency. The difference with respect to automata is that in a labelled transition system no initial state is specified. An automaton describes the behavior of a single system, and hence the initial state of the automaton corresponds to the starting point of the system; a labelled

transition system is used to represent the operational semantics of a whole concurrent formalism, and hence an initial state cannot be defined.

Various notions of behavioral preorders and equivalences have been defined on automata. The most important equivalence is *bisimulation equivalence* [Par80, Mil89].

Definition 3 (bisimulation on automata). *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two automata on the same set L of labels. A relation $\mathcal{R} \subseteq Q_1 \times Q_2$ is a* simulation *for $\mathcal{A}_1$ and $\mathcal{A}_2$ if $q_1 \mathcal{R} q_2$ implies:*

for all transitions $t_1 : q_1 \xrightarrow{l} q_1'$ of $\mathcal{A}_1$ there is some transition $t_2 : q_2 \xrightarrow{l} q_2'$ of $\mathcal{A}_2$ such that $q_1' \mathcal{R} q_2'$.

A relation $\mathcal{R} \subseteq Q_1 \times Q_2$ is a bisimulation *for $\mathcal{A}_1$ and $\mathcal{A}_2$ if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.*
Two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ on the same set of labels are bisimilar*, written $\mathcal{A}_1 \sim \mathcal{A}_2$, if there is some bisimulation $\mathcal{R}$ for $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $q_{01} \mathcal{R} q_{02}$.*

An important result in the theory of automata in concurrency is the existence of *minimal representatives* in the classes of bisimilar automata. Given an automaton, a reduced automaton is obtained by collapsing each class of equivalent states into a single state (and similarly for the transitions). This reduced automaton is bisimilar to the starting one, and any further collapse of states would lead to a non-bisimilar automaton. The reduced automaton is hence "minimal". Moreover, the same minimal automaton (up to isomorphisms) is obtained from bisimilar automata: thus it can be used as a canonical representative of the whole class of bisimilar automata.

In the definition below we denote with $[q]_{\mathcal{R}_\mathcal{A}}$ the class of equivalence of state q with respect to the largest bisimulation equivalence $\mathcal{R}_\mathcal{A}$ on automaton $\mathcal{A}$. With a light abuse of notation, we denote with $[t]_{\mathcal{R}_\mathcal{A}}$ the class of equivalent of transition t, where

$$t_1 \mathcal{R}_\mathcal{A} t_2 \quad \text{iff} \quad s(t_1) \mathcal{R}_\mathcal{A} s(t_2), \quad d(t_1) \mathcal{R}_\mathcal{A} d(t_2) \quad \text{and} \quad o(t_1) = o(t_2).$$

Definition 4 (minimal automata). *The* minimal automaton $\mathcal{A}_{\min}$ *corresponding to automaton $\mathcal{A}$ is defined as follows:*

- $L_{\min} = L$;
- $Q_{\min} = \{[q]_{\mathcal{R}_\mathcal{A}} \mid q \in Q\}$ *and* $T_{\min} = \{[t]_{\mathcal{R}_\mathcal{A}} \mid t \in T\}$;
- $s_{\min}([t]_{\mathcal{R}_\mathcal{A}}) = [s(t)]_{\mathcal{R}_\mathcal{A}}$ *and* $d_{\min}([t]_{\mathcal{R}_\mathcal{A}}) = [d(t)]_{\mathcal{R}_\mathcal{A}}$;
- $o_{\min}([t]_{\mathcal{R}_\mathcal{A}}) = o(t)$;
- $q_{0\min} = [q_0]_{\mathcal{R}_\mathcal{A}}$.

2.2 The π-Calculus

In this section we describe the π-calculus [MPW92, Mil93], a process calculus in which channel names can be used as values in the communications, i.e., channels are first-order values. This possibility of communicating names gives to the π-calculus a rich expressive power: in fact it allows to generate dynamically new channels and to change the interconnection structure of the processes. The π-calculus has been successfully

used to model object oriented languages [Wal95], and also higher-order communications can be easily encoded in the π-calculus [San93a], thus allowing for code migration.

Many versions of π-calculus have appeared in the literature. For simplicity, we consider only the *monadic* π-calculus, and we concentrate on the *ground* variant of its semantics.

Let $\mathcal{N}$ be an infinite, denumerable set of *names*, ranged over by $a, b, \ldots y, z \ldots$, and let Var be a finite set of *process identifiers*, denoted by $A, B, \ldots$; the π-calculus (monadic) *processes*, ranged over by $p, q, \ldots$, are defined by the syntax:

$$p \ ::= \ \mathbf{0} \ \Big| \ \pi.p \ \Big| \ p|p \ \Big| \ p{+}p \ \Big| \ (\nu x)\, p \ \Big| \ A(x_1, \ldots, x_n)$$

where the *prefixes* π are defined by the syntax:

$$\pi \ ::= \ \tau \ \Big| \ \bar{x}y \ \Big| \ x(y).$$

The occurrences of y in $x(y).p$ and $(\nu y)\, p$ are bound; *free* and *bound names* of process p are defined as usual and we denote them with $\mathrm{fn}(p)$ and $\mathrm{bn}(p)$ respectively. For each identifier A there is a definition $A(y_1, \ldots, y_n) \stackrel{\mathrm{def}}{=} p_A$ (with y_i all distinct and $\mathrm{fn}(p_A) \subseteq \{y_1, \ldots, y_n\}$); we assume that, whenever A is used, its arity n is respected. Finally we require that each process identifier in p_A is in the scope of a prefix (guarded recursion).

Some comments on the syntax of π-calculus are now in order. As usual, $\mathbf{0}$ is the terminated process. In process $\pi.p$ the prefix π defines an action to execute before p is activated. The prefix $\tau.p$ describes an internal (invisible) action of the process. The *output* prefix $\bar{x}y.p$ specifies the channel x for the communication and the value y that is sent on x. In the *input* prefixes $x(y).p$, name x represents the channel, whereas y is a formal variable: its occurrences in p are instantiated with the received value. Process $p|q$ is the parallel composition with synchronization of p and q, whereas $p{+}q$ is the nondeterministic choice. Process $(\nu x)\, p$ restricts the possible interactions of process p, disabling communications on channel x.

We use σ, ρ to range over name substitutions, and we denote with $\{y_1/x_1 \cdots y_n/x_n\}$ the substitution that maps x_i into y_i for $i = 1, \ldots, n$ and that is the identity on the other names.

We now introduce a *structural congruence* of π-calculus processes. This structural congruence allows us to identify all the processes which represent essentially the same system and which differ just for syntactical details. The structural congruence $\equiv$ is the smallest congruence which respects the following equivalences

(alpha)	$(\nu x)\, p \equiv (\nu y)\, (p\{y/x\})$ if y does not appear in p
(sum)	$p{+}\mathbf{0} \equiv p \qquad p{+}q \equiv q{+}p \qquad p{+}(q{+}r) \equiv (p{+}q){+}r$
(par)	$p|\mathbf{0} \equiv p \qquad p|q \equiv q|p \qquad p|(q|r) \equiv (p|q)|r$
(res)	$(\nu x)\, \mathbf{0} \equiv \mathbf{0} \qquad (\nu x)\, (\nu y)\, p \equiv (\nu y)\, (\nu x)\, p$
	$(\nu x)\, (p|q) \equiv p|(\nu x)\, q$ if x does not appear in p

The structural congruence is useful in practice to obtain finite state representations for classes of processes. It can be used to garbage-collect terminated component — by

exploiting rule $p|\mathbf{0} \equiv p$ — and unused restrictions — by using the rules above, if α does not appear in p then $(\nu\alpha)\, p \equiv p$: in fact, $(\nu x)\, p \equiv (\nu x)\, (p|\mathbf{0}) \equiv p|(\nu x)\, \mathbf{0} \equiv p|\mathbf{0} \equiv p$.

By exploiting the structural congruence $\equiv$, each π-calculus process can be seen as a set of *sequential processes* that act in parallel, sharing a set of channels, some of which are global (unrestricted) while some other are local (restricted). Each sequential process is represented by a term of the form

$$s \ ::= \ \pi.p \ \Big| \ p{+}p \ \Big| \ A(x_1, \ldots, x_n)$$

that can be considered as a "program" describing all the possible behaviors of the sequential process.

The *ground* semantics of the π-calculus is the simplest operational semantics that can be defined for this language. It differs from other semantics, such as the *early* and *late* semantics, in the management of input transitions [MPW93]. According to the early semantics, process $x(y).p$ can perform a whole bunch of input transitions

$$x(y).p \xrightarrow{xz} p\{z/y\}$$

corresponding to the different names z that the environment can send to the process to instantiate the formal input parameter y. In the ground semantics, instantiation of the input parameters are not taken into account, and process $x(y).p$ can perform only one input transition:

$$x(y).p \xrightarrow{x(y)} p.$$

Ground bisimilarity is easy to check[1]. However, it is less discriminating than early bisimilarity, and does not capture the possibility for the environment of communicating an already existing name during an input transition of a process. For instance,

$$x(y).(\bar{y}y.\mathbf{0}|z(w).\mathbf{0}) \text{ and } x(y).(\bar{y}y.z(w).\mathbf{0} + z(w).\bar{y}y.\mathbf{0})$$

are not equivalent according to the early semantics, since, performing input xz we obtain

$$\bar{z}z.\mathbf{0}|z(w).\mathbf{0} \text{ and } \bar{y}y.z(w).\mathbf{0} + z(w).\bar{y}y.\mathbf{0}$$

and a synchronization (i.e., a τ transition) is possible in the first process but not in the second. However,

$$x(y).(\bar{y}y.\mathbf{0}|z(w).\mathbf{0}) \text{ and } x(y).(\bar{y}y.z(w).\mathbf{0} + z(w).\bar{y}y.\mathbf{0})$$

are equivalent according to the ground semantics since the reception of the already existing name z is not allowed. For simplicity, in this paper we consider only the ground semantics. The presented results, however, can easily be extended to the other π-calculus semantics.

The *ground actions* that a process can perform are defined by the following syntax:

$$\mu \ ::= \ \tau \ \Big| \ x(y) \ \Big| \ \bar{x}y \ \Big| \ \bar{x}(y)$$

and are called respectively *synchronization*, *input*, *free output* and *bound output* actions.

[1] ... and, as we will see, easy to model with HD-automata.

Table 1. Free and bound names of π-calculus actions

μ	$\mathrm{fn}(\mu)$	$\mathrm{bn}(\mu)$	$\mathrm{n}(\mu)$
τ	$\emptyset$	$\emptyset$	$\emptyset$
$x(y)$	$\{x\}$	$\{y\}$	$\{x,y\}$
$\overline{x}y$	$\{x,y\}$	$\emptyset$	$\{x,y\}$
$\overline{x}(y)$	$\{x\}$	$\{y\}$	$\{x,y\}$

Table 2. Ground operational semantics of π-calculus

$$[\text{PREF}]\ \pi.p \xrightarrow{\pi} p \qquad [\text{SUM}]\ \frac{p_1 \xrightarrow{\mu} p'}{p_1+p_2 \xrightarrow{\mu} p'}$$

$$[\text{COMM}]\ \frac{p_1 \xrightarrow{\bar{x}y} p_1' \quad p_2 \xrightarrow{x(z)} p_2'}{p_1|p_2 \xrightarrow{\tau} p_1'|(p_2'\{y/z\})} \qquad [\text{PAR}]\ \frac{p_1 \xrightarrow{\mu} p_1'}{p_1|p_2 \xrightarrow{\mu} p_1'|p_2}\ \text{if } \mathrm{bn}(\mu)\cap\mathrm{fn}(p_2)=\emptyset$$

$$[\text{OPEN}]\ \frac{p \xrightarrow{\bar{x}y} p'}{(\nu y)\,p \xrightarrow{\bar{x}(y)} p'}\ \text{if } x\neq y \qquad [\text{CLOSE}]\ \frac{p_1 \xrightarrow{\bar{x}(y)} p_1' \quad p_2 \xrightarrow{x(y)} p_2'}{p_1|p_2 \xrightarrow{\tau} (\nu y)\,(p_1'|p_2')}$$

$$[\text{RES}]\ \frac{p \xrightarrow{\mu} p'}{(\nu x)\,p \xrightarrow{\mu} (\nu x)\,p'}\ \text{if } x \notin \mathrm{n}(\mu)$$

$$[\text{IDE}]\ \frac{p_A\{y_1/x_1 \ldots y_n/x_n\} \xrightarrow{\mu} p'}{A(y_1,\ldots,y_n) \xrightarrow{\mu} p'}\ \text{if } A(x_1,\ldots,x_n) \stackrel{\text{def}}{=} p_A$$

The *free names*, *bound names* and *names* of an action μ, respectively written $\mathrm{fn}(\mu)$, $\mathrm{bn}(\mu)$ and $\mathrm{n}(\mu)$, are defined as in Table 1.

The transitions for the *ground operational semantics* are defined by the axiom schemata and the inference rules of Table 2. We remind that rule

$$\frac{p \equiv p' \quad p' \xrightarrow{\mu} p'' \quad p'' \equiv p''}{p \xrightarrow{\mu} p''}$$

is implicitly assumed.

Notice that the actions a π-calculus process can perform are different from the prefixes. This happens due to the bound output actions. These actions are specific of the π-calculus; they represent the communication of a name that was previously restricted, i.e., it corresponds to the generation of a new channel between the process and the environment: this phenomenon is called *name extrusion*.

Now we present the definition of the ground bisimulation for the π-calculus.

Definition 5 (ground bisimulation). *A relation $\mathcal{R}$ over processes is an* ground simulation *if whenever $p\ \mathcal{R}\ q$ then:*

> *for each $p \xrightarrow{\mu} p'$ with $\mathrm{bn}(\mu)\cap\mathrm{fn}(p|q)=\emptyset$ there is some $q \xrightarrow{\mu} q'$ such that $p'\ \mathcal{R}\ q'$.*

A relation $\mathcal{R}$ is an ground bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are ground simulations.*

Two processes p and q are ground bisimilar*, written $p \sim_g q$, if $p \mathcal{R} q$ for some ground bisimulation $\mathcal{R}$.*

In the definition above, clause "$\mathrm{bn}(\mu) \cap \mathrm{fn}(p|q) = \emptyset$" is necessary to guarantee that the name, that is chosen to represent the newly created channel in a bound output transition, is fresh for both the processes. This clause is necessary since equivalent processes may have different sets of free names.

As for other process calculi, a labelled transition system is used to give an operational semantics to the π-calculus. However, this way to present the operational semantics has some disadvantages. Consider process $q = (\nu y)\,\bar{x}y.y(z).\mathbf{0}$. It is able to generate a new channel by communicating name y in a bound output. The creation of a new name is represented in the transition system by means of an infinite bunch of transitions $q \xrightarrow{\bar{x}(w)} w(z).\mathbf{0}$, where, in this case, w is any name different from x: the creation of a new channel is modeled by using all the names which are not already in use to represent it. As a consequence, the definition of bisimulation is not the ordinary one: in general two bisimilar process can have different sets free names, and the clause "$\mathrm{bn}(\mu) \cap \mathrm{fn}(p|q) = \emptyset$" has to be added in Definition 5 to deal with those bound output transitions which use a name that is used only in one of the two processes. The presence of this clause makes it difficult to reuse standard theory and algorithms for bisimulation on the π-calculus — see for instance [Dam97].

3 History-Dependent Automata

Ordinary automata are successful basic process calculi like CCS. For more sophisticated calculi, however, they are not: in fact, they are not able to capture the particular structures of these languages, that is represented in ordinary automata only in an implicit way. As a consequence, infinite-state automata are often obtained also for very simple programs. To model these languages, it is convenient to enrich states and labels with (part of) the information of the programs, so that the particular structures manipulated by the languages are represented explicitly. These enriched automata are hence more adherent to the languages than ordinary automata.

Different classes of enriched automata can be defined by changing the kind of additional information. Here we focus on a simple form of enriched automata. They are able to manipulate generic "resources": a resource can be allocated, used, and finally released. At this very abstract level, resources can be represented by names: the allocation of a resource is modeled by the generation of a fresh name, that is then used to refer to the resource; since we do not assume any specific operation on resources, the usage of a resource in a transition is modeled by observing the corresponding name in the label; finally, a resource is (implicitly) deallocated when the corresponding name is no more referenced.

We call this class of enriched automata *History-Dependent Automata*, or *HD-automata* in brief. In fact, the usage of names described above can be considered a way to express dependencies between the transitions of the automaton; a transition that uses a name depends on the past transition that generated that name.

In this section we introduce HD-automata and HD-bisimulation and we show that they are able to capture in a convenient way the ground semantics of π-calculus, where the names are use to represent the communication channels.

3.1 HD-Automata

HD-automata extend ordinary automata by allowing sets of names to appear explicitly in states and labels. We assume that the names that are associated to a state or a label are *local* names and do not have a global identity. This is very convenient, since a single state of the HD-automaton can be used to represent all the states of a system that differ just for a renaming (that is, HD-automata work up to bijective substitutions of names). In this way, however, each transition is required to represent explicitly the correspondences between the names of source, target and label. As the reader can see in Figure 1, to represent these correspondences we associate a set of names also to each transition, and we embed the names of the source and target states, and of the label into the names of the transition.

Technically, we represent states, transitions and labels of a HD-automaton by means of *named sets* and use *named functions* to associate a source state, a target state and a label to each transition.

In a named set E, each element e is enriched with a set of names that we denote with $\mathsf{E}[e]$. A function from named set E to named set F maps each element e of the first in an element f of the second; moreover, it also fixes a correspondence between the names of e and the names of f. More precisely, this correspondence provides an embedding of the names of the target element f into the names of the source element e; that is, the names of f are seen, through the name correspondence, as a subset of the names of e.

Now we introduce some notation on functions that we will use extensively in the following. Then we define formally named sets and, based on them, the HD-automata.

Notation 6. *A relation $\mathcal{R}$ on sets A and B is a subset of $A \times B$. If $(a,b) \in \mathcal{R}$ then we also write $a\ \mathcal{R}\ b$. In this case,* $\mathrm{dom}(\mathcal{R}) = \{a \mid (a,b) \in \mathcal{R}\}$ *is the* domain *of $\mathcal{R}$ and* $\mathrm{cod}(\mathcal{R}) = \{b \mid (a,b) \in \mathcal{R}\}$ *is its* codomain. *We denote with $\mathcal{R}^{-1}$ the* inverse *relation of $\mathcal{R}$; that is, $\mathcal{R}^{-1} = \{(b,a) \mid (a,b) \in \mathcal{R}\} \subseteq B \times A$. If $\mathcal{R}$ is a relation on A and B and $\mathcal{S}$ is a relation on B and C, then we denote with $\mathcal{R};\mathcal{S}$ the* composition *of $\mathcal{R}$ and $\mathcal{S}$; that is, $\mathcal{R};\mathcal{S} = \{(a,c) \mid (a,b) \in \mathcal{R}$ and $(b,c) \in \mathcal{S}\} \subseteq A \times C$.*

Special notations are used for particular classes of relations.

We represent with $f : A \to B$ a function *from set A to set B; that is, $f \subseteq A \times B$ such that for each $a \in A$ there is exists exactly one $a \in A$ such that $(a,b) \in f$.*

We represent with $f : A \rightharpoonup B$ a partial bijection *from set A to set B; that is, $f \subseteq A \times B$ such that if $(a,b),(a',b') \in f$ then $a = a'$ iff $b = b'$.*

We represent with $f : A \rightarrowtail B$ an injection *from set A to set B; that is, $f \subseteq A \times B$ such that for each $a \in A$ there exists exactly one $b \in B$ such that $(a,b) \in f$, and for each $b \in B$ there is at most one $a \in A$ such that $(a,b) \in f$.*

We represent with $f : A \twoheadleftarrow B$ an inverse injection *from set A to set B; that is, $f \subseteq A \times B$ such that for each $b \in B$ there exists exactly one $a \in A$ such that $(a,b) \in f$, and for each $a \in A$ there is at most one $b \in B$ such that $(a,b) \in f$.*

We represent with $f : A \longleftrightarrow B$ *a* total bijection *from set* A *to set* B*; that is,* $f \subseteq A \times B$ *such that for each* $a \in A$ *there exists exactly one* $b \in B$ *such that* $(a, b) \in f$ *and, conversely, for each* $b \in B$ *there exists exactly one* $a \in A$ *such that* $(a, b) \in f$*.*

We use also on these subclasses the notations that we have introduced on relations to denote domain, codomain, inverse and composition.

Definition 7 (named sets). *Let* $\mathcal{N}$ *be an infinite denumerable set of names and let* $\mathcal{P}(\mathcal{N})$ *be the power-set of* $\mathcal{N}$*.*
A named set E *is a set, denoted by* E*, and a family of subset of names indexed by* E*, namely* $\{\mathsf{E}[e] \subseteq \mathcal{N}\}_{e \in E}$*, or, equivalently* $\mathsf{E}[\_]$ *is a map from* E *to* $\mathcal{P}(\mathcal{N})$*.*
Given two named sets E *and* F*, a* named function $\mathsf{m} : \mathsf{E} \to \mathsf{F}$ *is a function on the sets* $m : E \to F$ *and a family of name embeddings indexed by* m*, namely* $\{\mathsf{m}[e] : \mathsf{E}[e] \hookleftarrow \mathsf{F}[f]\}_{(e,f) \in m}$*:*

$$\begin{array}{ccc} \mathsf{E} & \ni & e \\ \downarrow{\scriptstyle \mathsf{m}} & & \downarrow{\scriptstyle m} \\ \mathsf{F} & \ni & f \end{array} \qquad \begin{array}{c} \mathsf{E}[e] \\ \uparrow{\scriptstyle \mathsf{m}[e]} \\ \mathsf{F}[f] \end{array}$$

A named set E *is* finitely named *if* $\mathsf{E}[e]$ *is finite for each* $e \in E$*. A named set* E *is* finite *if it is finitely named and set* E *is finite.*

We remark that, in the definition of named function, we use an inverse injection from $\mathsf{E}[e]$ to $\mathsf{F}[f]$ to represent the correspondence between the names of e and the names of f: this inverse injection, in fact, can be seen as an embedding of the names of f into the names of e.

Now we define HD-automata: essentially, they have the same components of ordinary automata (Definition 1), but named sets and named functions are use rather than plain sets and functions.

Definition 8 (HD-automata). *A* HD-automaton $\mathcal{A}$ *is defined by:*

- *a named set* L *of* labels*;*
- *a named set* Q *of* states*;*
- *a named set* T *of* transitions*;*
- *a pair of named functions* $\mathsf{s}, \mathsf{d} : \mathsf{T} \to \mathsf{Q}$*, which associate to each transition the* source *and* destination *states respectively (and embed the names of the source and of the destination states into the names of the transition);*
- *a named function* $\mathsf{o} : \mathsf{T} \to \mathsf{L}$*, which associates a label to each transition (and embeds the names of the label into the names of the transition);*
- *an* initial state $q_0 \in Q$ *and an* initial embedding $\sigma_0 : \mathsf{Q}[q_0] \hookleftarrow \mathcal{N}$ *of the local names of* q_0 *into the infinite, denumerable set* $\mathcal{N}$ *of* global names*.*

Let $\mathsf{T}[t]_{\mathrm{old}} \stackrel{\mathrm{def}}{=} \{n \in \mathsf{T}[t] \mid n \in \mathrm{dom}(\mathsf{s}[t])\}$ *and* $\mathsf{T}[t]_{\mathrm{new}} \stackrel{\mathrm{def}}{=} \{n \in \mathsf{T}[t] \mid n \notin \mathrm{dom}(\mathsf{s}[t])\}$ *be respectively the* old names *and the* new names *of transition* $t \in T$*.*
A HD-automaton is finitely named *if* L*,* Q *and* T *are finitely named; it is* finite *if, in addition,* Q *and* T *are finite.*

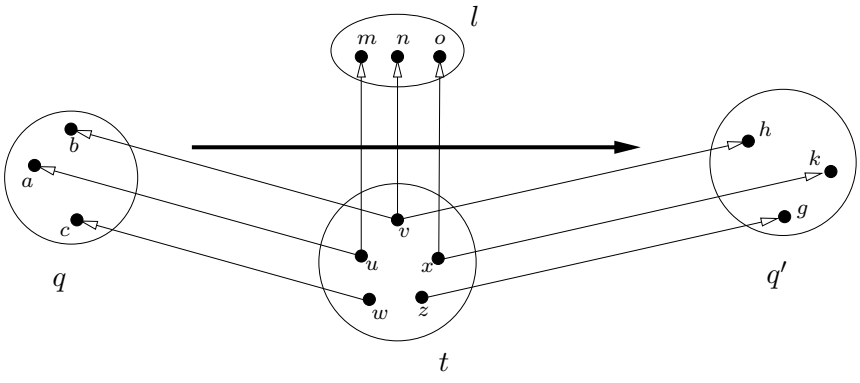

Fig. 1. A transition $t : q \xrightarrow{l} q'$ of a HD-automaton

Let t be a generic transition of a HD-automaton such that $s(t) = q$, $d(t) = q'$ and $o(t) = l$ (in brief $t : q \xrightarrow{l} q'$); one of such transition is represented in Figure 1. Then $\mathsf{s}[t] : \mathsf{T}[t] \hookleftarrow \mathsf{Q}[q]$ embeds, by means of an inverse injection, the names of q into the names of t, whereas $\mathsf{d}[t] : \mathsf{T}[t] \hookleftarrow \mathsf{Q}[q']$ embeds the names of q' into the names of t; in this way, a partial correspondence is defined between the names of the source state and those of the target; so, in the case of the transition in figure, name h of the target state corresponds to name b of the source. The names that appear in the source and not in the target (that is, names a and c in Figure 1) are discarded, or forgotten, during the transition, whereas the names that appear in the target but not in the source (that is, names g and k in figure) are created during the transition.

3.2 From Ground π-Calculus to Basic HD-Automata

We are interested in the representation of the ground π-calculus semantics as HD-automata. First we define the named set of labels L^{π_g} for this language: we have to distinguish between synchronizations, bound inputs, free outputs and bound outputs. Thus the set of labels is

$$L^{\pi_g} = \{\mathtt{tau}, \mathtt{bin}, \mathtt{out}, \mathtt{out}_2, \mathtt{bout}\}$$

where $\mathtt{out}_2$ is used when subject and object names of free outputs coincide (these special labels are necessary, since the function from the names associated to a label into the names associated to a transition must be injective). No name is associated to `tau`, one name (n) is associated to $\mathtt{out}_2$, and two names (n_{sub} and n_{obj}) are associated to `bin`, `out` and `bout`.

In order to associate a HD-automaton to a π-calculus process, we have to represent the derivatives of the process as states of the automaton and their transitions as transitions in the HD-automaton; the names corresponding to a state are the free names of the corresponding process, the names corresponding to a transition are the free names of the source state plus, in the case of a bound input and bound output transition, the

new name appearing in the label of the transition. A label of L^{π_g} is associated to each transition in the obvious way.

This naive construction can be improved to obtain more compact HD-automata. Consider for instance process $p = (\nu z)\,\bar{x}z.B(x, y, z)$; it can perform an infinite number of bound output transitions, depending on the different extruded name. In the case of HD-automata, due to the local nature of names, it is not necessary to consider all the different bound output (and bound input) transitions that differ only on the name used to denote the new created channel. The syntactic identity of that name, in fact, is inessential in the model. A single transition can be chosen from each of these infinite bunches. Here we use transition $p \xrightarrow{\overline{x}(z)} p'$ where $z = \min\big(\mathcal{N} \setminus \mathrm{fn}(p)\big)$. It is worth to stress out that, differently from the case of ordinary automata, where particular care is needed in the choice of this transition, in the case of HD-automata any policy for choosing the fresh name will work: in this case, in fact, we do not have to guarantee that equivalent states choose the same name.

Definition 9 (representative transitions). *A π-calculus transition $p \xrightarrow{\mu} q$ is a* representative transition *if*

$$\mathrm{n}(\mu) \subseteq \mathrm{fn}(p) \cup \big\{\, \min\big(\mathcal{N} \setminus \mathrm{fn}(p)\big)\big\}.$$

According to this definition, all the synchronization and free output transitions are representative (in this case $\mathrm{n}(\mu) \subseteq \mathrm{fn}(p)$). A bound input or a bound output is representative only if the communicated name is the smallest name not appearing free in the process.

The following lemma shows that the representative transitions express, up to α-conversion, all the behaviors of a process.

Lemma 1. *Let $p \xrightarrow{\mu} q$, with $\mu = ax$ (resp. $\mu = \bar{a}(x)$), be a non-representative π-calculus transition. Then there is some representative transition $p \xrightarrow{\mu'} q'$, with $\mu' = ay$ (resp. $\mu' = \bar{a}(y)$), such that $q' = q\{y/x\, x/y\}$.*

If only representative transitions are used when building a HD-automaton from a π-calculus process, the obtained HD-automaton is *finite-branching*, i.e., it has a finite set of transitions from each state.

Another advantage of using local names is that two processes differing only for a bijective substitution can be collapsed in the same state in the HD-automaton: we assume to have a function norm that, given a process p, returns a pair $(q, \sigma) = \mathrm{norm}(p)$, where q is the representative of the class of processes differing from p for bijective substitutions and $\sigma : \mathrm{fn}(p) \longleftrightarrow \mathrm{fn}(q)$ is the bijective substitution such that $q = p\sigma$.

Definition 10 (from π-calculus to HD-automata). *The HD-automaton $\mathcal{A}_p^{\pi_g}$ corresponding to the ground semantics of π-calculus process p is defined as follows:*

- *if* $\mathrm{norm}(p) = (q_0, \sigma_0)$ *then:*
  - *$q_0 \in Q$ is the initial state and $\mathsf{Q}[q_0] = \mathrm{fn}(q_0)$;*
  - *$\sigma_0^{-1} : \mathrm{fn}(q_0) \longleftrightarrow \mathrm{fn}(p)$ is the initial embedding;*

Table 3. Relations between π-calculus labels and labels of HD-automata

μ	τ	$x(y)$		$\bar{x}y$		$\bar{x}x$	$\bar{x}(y)$	
l	`tau`	`bin`		`out`		$\texttt{out}_2$	`bout`	
$\Box = \lambda(\Diamond) \in \mathrm{n}(\mu)$	/	x	y	x	y	x	x	y
$\Diamond = \kappa(\Box) \in \mathsf{L}^{\pi_g}[l]$	/	n_{sub}	n_{obj}	n_{sub}	n_{obj}	n	n_{sub}	n_{obj}

– *if* $q \in Q$, $t : q \xrightarrow{\mu} q'$ *is a representative transition and* $\mathrm{norm}(q') = (q'', \sigma)$, *then:*
  - $q'' \in Q$ *and* $\mathsf{Q}[q''] = \mathrm{fn}(q'')$*;*
  - $t \in T$ *and* $\mathsf{T}[t] = \mathrm{fn}(q) \cup \mathrm{bn}(\mu)$*;*
  - $s(t) = q$, $d(t) = q''$, $\mathsf{s}[t] = \mathrm{id}_{\mathrm{fn}(q)}$ *and* $\mathsf{d}[t] = \sigma$*;*
  - $o(t) = l$ *and* $\mathsf{o}[t] = \kappa$ *are defined as in Table 3.*

Table 3 defines the correspondence between the labels of π-calculus transitions and the HD-automaton labels: so, for instance, an input action $x(y)$ of a π-calculus process is represented in the HD-automaton by means of label `bin`. Moreover, the table also fixes the correspondence between the names that appear in the π-calculus label and the names of the HD-automaton label. This correspondence is defined by means of two functions: function κ maps the names of a π-calculus label μ into the names of the corresponding label l of the HD-automaton, while λ maps the names of l into the names of μ. Both functions are total bijections, and clearly $\kappa = \lambda^{-1}$. In the case of the input action $x(y)$, we have $\mathrm{n}(x(y)) = \{x, y\}$ and $\mathsf{L}^{\pi_g}[\texttt{bin}] = \{n_{\mathrm{sub}}, n_{\mathrm{obj}}\}$; in this case, according to Table 3, functions $\kappa : \{x, y\} \to \{n_{\mathrm{sub}}, n_{\mathrm{obj}}\}$ and $\lambda : \{n_{\mathrm{sub}}, n_{\mathrm{obj}}\} \to \{x, y\}$ are defined as follows: $\kappa(x) = n_{\mathrm{sub}}$ and $\lambda(n_{\mathrm{sub}}) = x$; $\kappa(y) = n_{\mathrm{obj}}$ and $\lambda(n_{\mathrm{obj}}) = y$. We have used function κ in Definition 10; function λ will become useful in the following.

For each π-calculus process p, the HD-automaton $\mathcal{A}_p^{\pi_g}$ is obviously finitely named. Now we identify a class of processes that generate finite HD-automata. This is the class of *finitary* π-calculus processes.

Definition 11 (finitary processes). *The* degree of parallelism $\deg(p)$ *of a* π*-calculus process* p *is defined as*

$$\deg(\mathbf{0}) = 0 \qquad \deg(\pi.p) = 1$$
$$\deg((\nu x)\, p) = \deg(p) \qquad \deg(p|q) = \deg(p) + \deg(q)$$
$$\deg(p{+}q) = \max\{\deg(p), \deg(q)\} \qquad \deg(A(x_1, \ldots, x_n)) = 1$$

A π*-calculus process* p *is* finitary *if* $\max\{\deg(p') \mid p \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_i} p'\} < \infty$.

Theorem 1. *Let* p *be a finitary* π*-calculus process. Then the HD-automaton* $\mathcal{A}_p^{\pi_g}$ *is finite.*

We remark that, it is only semidecidable whether a process is finitary. Also in this case, however, there is a syntactic conditions that guarantees that a π-calculus process is finitary: the *finite-control* condition. A process p has a finite control if no parallel composition appears in the recursive definitions used by p.

Corollary 1. *Let* p *be a finite-control* π*-calculus process. Then the HD-automaton* $\mathcal{A}_p^{\pi_g}$ *is finite.*

3.3 Bisimulation on HD-Automata

We introduce now bisimilarity on HD-automata and give some of its basic properties. We also show that ground bisimilarity of π-calculus processes is captured exactly by the bisimulation on HD-automata.

Due to the private nature of the names appearing in the states of HD-automata, bisimulations cannot simply be relations on the states; they must also deal with name correspondences: a HD-bisimulation is a set of triples of the form $\langle q_1, \delta, q_2 \rangle$ where q_1 and q_2 are states of the automata and δ is a partial bijection between the names of the states. The bijection is partial since we allow for equivalent states with different numbers of names.

Suppose that we want to check if states q_1 and q_2 are bisimilar via the partial bijection $\delta : \mathsf{Q}[q_1] \longleftrightarrow \mathsf{Q}[q_2]$ and suppose that q_1 can perform a transition $t_1 : q_1 \xrightarrow{l} q_1'$: an instance of this situation is represented in Figure 2. Then we have to find a transition $t_2 : q_2 \xrightarrow{l} q_2'$ that matches t_1, i.e., not only the two transitions must have the same label, but also the names associated to the labels must be used consistently. This means that, given a name n of the label:

- either n is *old* in both transitions, i.e., it corresponds to some name n_1 of state q_1 and to some name n_2 of q_2 (via the suitable name embeddings), and these names n_1 and n_2 are in correspondence by δ; this is the case of name h of label l in Figure 2: it corresponds to names a_1 and a_2 in the source states, and these are related by δ;
- or n is *new* in both transitions, i.e., it does not correspond to any name n_1 of state q_1, nor to any name n_2 of q_2; this is the case of name k of label l in Figure 2: in fact, the corresponding names y_1 and y_2 in the transitions are new.

This behavior is obtained by requiring that a partial bijection $\zeta : \mathsf{T}[t_1] \longleftrightarrow \mathsf{T}[t_2]$ exists such that: (i) ζ coincides with δ if restricted to the names of the source states (obviously, via the embeddings $\mathsf{s}[t_1]$ and $\mathsf{s}[t_2]$), and extends δ with a partial correspondence ξ between the new names of t_1 and t_2; (ii) the names associated to the labels are the same, via ζ, and (iii) the destination states q_1' and q_2' are bisimilar via a partial bijection δ' which is compatible with ζ (i.e., if two names are related by δ' in the destination states, then the corresponding names in the transitions are related by ζ). The reader can check that all these requirements are satisfied in Figure 2.

We remark that it is *not* required that two names of the destination states are related by δ' if the corresponding names of the transitions are related by ζ. That is, we allow some of the correspondences that hold in the transitions to be discarded in the destination states. In Figure 2, for instance, names f_1 and f_2 of the target states are not related by δ', even if the corresponding names of the transitions, namely z_1 and z_2, are related by ζ. We will comment further on this choice later in this section. We anticipate that the same equivalence on HD-automata is obtained also by requiring that no correspondence can be discarded in the target states.

Definition 12 (HD-bisimulation). *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two HD-automata. A* HD-simulation *for $\mathcal{A}_1$ and $\mathcal{A}_2$ is a set of triples $\mathcal{R} \subseteq \{\langle q_1, \delta, q_2 \rangle \mid q_1 \in Q_1,\, q_2 \in Q_2,\, \delta : \mathsf{Q}_1[q_1] \longleftrightarrow \mathsf{Q}_2[q_2]\}$ such that, whenever $\langle q_1, \delta, q_2 \rangle \in \mathcal{R}$ then:*

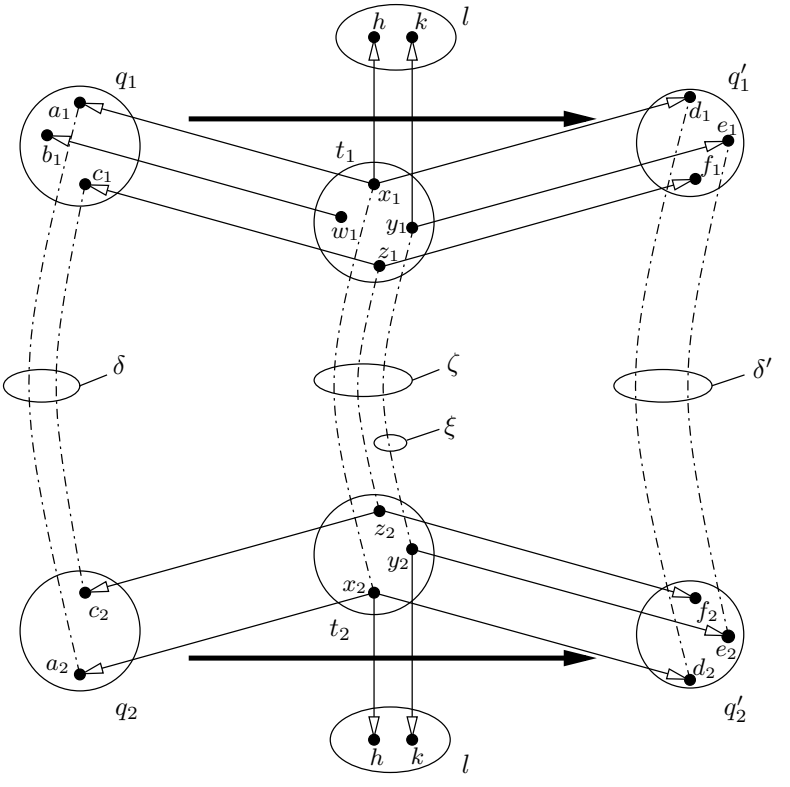

Fig. 2. A step of bisimulation on HD-automata

for each $t_1 : q_1 \xrightarrow{l} q_1'$ *in* $\mathcal{A}_1$ *there exist some* $t_2 : q_2 \xrightarrow{l} q_2'$ *in* $\mathcal{A}_2$*, some* $\xi : \mathsf{T}_1[t_1]_{\text{new}} \longleftrightarrow \mathsf{T}_2[t_2]_{\text{new}}$*, and some* $\zeta : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_2[t_2]$ *such that:*

- $\zeta = \left(\mathsf{s}_1[t_2]; \delta; \mathsf{s}_2[t_2]^{-1}\right) \cup \xi$,
- $\mathsf{o}_1[t_1] = \zeta; \mathsf{o}_2[t_2]$,
- $\langle q_1', \delta', q_2' \rangle \in \mathcal{R}$ *where* $\delta' \subseteq \mathsf{d}_1[t_1]^{-1}; \zeta; \mathsf{d}_2[t_2]$.

A HD-bisimulation *for* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *is a set of triples* $\mathcal{R}$ *such that* $\mathcal{R}$ *is a HD-simulation for* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *and* $\mathcal{R}^{-1} = \{\langle q_2, \delta^{-1}, q_1 \rangle \mid \langle q_1, \delta, q_2 \rangle \in \mathcal{R}\}$ *is a HD-simulations for* $\mathcal{A}_2$ *and* $\mathcal{A}_1$.

A HD-bisimulation for $\mathcal{A}$ *is a HD-bisimulation for* $\mathcal{A}$ *and* $\mathcal{A}$.

The HD-automata $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *are* HD-bisimilar *(written* $\mathcal{A}_1 \sim \mathcal{A}_2$*) if there exists some HD-bisimulation for* $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *such that* $\langle q_{01}, \delta, q_{02} \rangle \in \mathcal{R}$ *for some* $\delta \subseteq \sigma_{01}; \sigma_{02}^{-1}$.

Now we present some basic properties of HD-bisimulations.

Proposition 1. *Let* $\{\mathcal{R}_i \mid i \in I\}$ *be a (finite or infinite) set of HD-bisimulations for* $\mathcal{A}_1$ *and* $\mathcal{A}_2$*. Then* $\bigcup_{i \in I} \mathcal{R}_i$ *is a HD-bisimulation for* $\mathcal{A}_1$ *and* $\mathcal{A}_2$.

This proposition allows us to define the greatest bisimulation between two automata.

Definition 13 (greatest HD-bisimulation). *We denote with $\mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2}$ the greatest HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e.:*

$$\mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2} \stackrel{\text{def}}{=} \{\langle q_1, \delta, q_2\rangle \mid \langle q_1, \delta, q_2\rangle \in \mathcal{R},\ \mathcal{R} \text{ HD-bisimulation for } \mathcal{A}_1 \text{ and } \mathcal{A}_2\}$$

We denote with $\mathcal{R}_{\mathcal{A}}$ the greatest HD-bisimulation for $\mathcal{A}$.

By the previous proposition, $\mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2}$ and $\mathcal{R}_{\mathcal{A}}$ are HD-bisimulations.

Proposition 2. *If $\mathcal{R}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ and $\mathcal{S}$ is a HD-bisimulations for $\mathcal{A}_2$ and $\mathcal{A}_3$ then $\mathcal{R} \frown \mathcal{S}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_3$, where:*

$$\mathcal{R} \frown \mathcal{S} \stackrel{\text{def}}{=} \{\langle q_1, (\delta; \delta'), q_3\rangle \mid \langle q_1, \delta, q_2\rangle \in \mathcal{R}, \langle q_2, \delta', q_3\rangle \in \mathcal{S}\}.$$

Proposition 3. *If $\mathcal{R}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\widehat{\mathcal{R}}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, where:*

$$\widehat{\mathcal{R}} \stackrel{\text{def}}{=} \{\langle q_1, \delta', q_2\rangle \mid \langle q_1, \delta, q_2\rangle \in \mathcal{R}, \delta \subseteq \delta'\}.$$

It is easy to see that $\mathcal{R}_{\mathcal{A}}$ is closed for $\widehat{\ }$ and - $\frown$ -. Moreover, relation $\sim$ is an equivalence on HD-automata: symmetry and reflexivity are immediate, whereas transitivity derives from the previous proposition.

Proposition 3 shows that, whenever two states of an automaton are equivalent via some partial correspondence of names, they also are equivalent for all the correspondences obtained by adding new relations between the names. By exploiting this fact, we can define HD-bisimulation with a stronger condition on the correspondence δ' for the destination states: in fact, we can require $\delta' = \mathsf{d}_1[t_1]^{-1}; \zeta; \mathsf{d}_2[t_2]$. Also with this alternative definition the same equivalence on HD-automata is obtained, and also the greatest bisimulation $\mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2}$ does not change.

The possibility of discarding correspondences in the definition of δ', though, is very convenient. First of all, it permits to exhibit smaller relations to prove HD-bisimilarity of two HD-automata. Furthermore, some important properties of HD-bisimulation do not hold if the discarding is not allowed. This is the case for instance of the concatenation property of Proposition 2: in fact, if we consider the HD-automaton of Figure 3, then relations

$$\mathcal{R} = \{\langle q_1, \delta_{12}, q_2\rangle, \langle q_1', \emptyset, q_2'\rangle\} \qquad \text{with } \delta_{12}(a) = b$$
$$\mathcal{S} = \{\langle q_2, \delta_{23}, q_3\rangle, \langle q_2', \emptyset, q_3'\rangle\} \qquad \text{with } \delta_{23}(b) = c$$

are HD-bisimulations; however, their concatenation

$$\mathcal{R} \frown \mathcal{S} = \{\langle q_1, \delta_{13}, q_3\rangle, \langle q_1', \emptyset, q_3'\rangle\} \qquad \text{with } \delta_{13}(a) = c$$

is *not* a HD-bisimulation if we do not permit to discard name correspondences, since names a' and c' of the target states are not related by $\mathcal{R} \frown \mathcal{S}$, even if the corresponding names a and c of the source states are related.

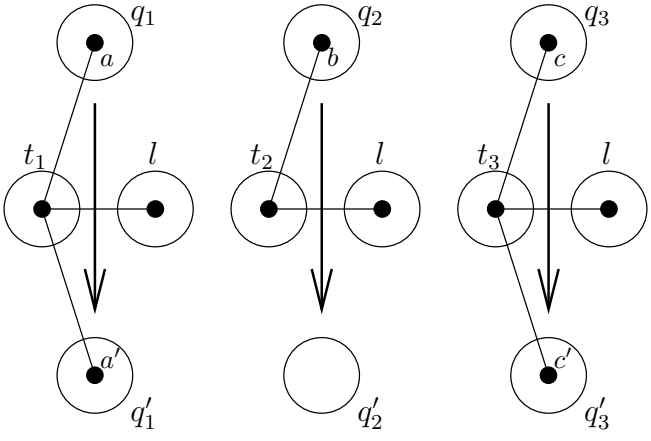

Fig. 3. A tricky example for concatenation of HD-bisimulations

3.4 Global States and Global Bisimulation

Now we give an alternative characterization of HD-bisimulation, which is based on global (rather than local) names. This alternative characterization is very useful to show that HD-bisimulation, when applied to HD-automata obtained from π-calculus processes, coincides with bisimilarity relation $\sim_g$.

We have seen that a state of a HD-automaton is obtained from a π-calculus process by normalizing its names, so that all the processes that differ for a renaming are represented by the same state. Conversely, a particular π-calculus process can be recovered from a state q of the HD-automaton by giving a global identity of the local names of q. Following this intuition, if q is a state of a HD-automaton and $\sigma : \mathsf{Q}[q] \longleftrightarrow \mathcal{N}$, then (q, σ) is a *global state*, i.e., a state where a global identity is assigned to the names. Global transitions are defined similarly.

Definition 14 (global state and global transition). *A* global state *of a HD-automaton $\mathcal{A}$ is a pair $g = (q, \sigma)$, where $q \in Q$ and $\sigma : \mathsf{Q}[q] \longleftrightarrow \mathcal{N}$. We denote with $G_\mathcal{A}$ the set of global states of $\mathcal{A}$. We denote with $\mathsf{G}_\mathcal{A}$ the named set of global state of $\mathcal{A}$, obtained by defining $\mathsf{G}_\mathcal{A}[(q, \sigma)] \stackrel{\text{def}}{=} \sigma(\mathsf{Q}[q])$.*
A global transition *is a pair $u = (t, \rho)$, where $t \in T$ and $\rho : \mathsf{T}[t] \longleftrightarrow \mathcal{N}$. We denote with $U_\mathcal{A}$ the set of global transitions of $\mathcal{A}$. We denote with $\mathsf{U}_\mathcal{A}$ the named set of global transitions of $\mathcal{A}$, obtained by defining $\mathsf{U}_\mathcal{A}[(t, \rho)] \stackrel{\text{def}}{=} \rho(\mathsf{U}[t])$. Moreover we use the notations $\mathsf{U}_\mathcal{A}[(t, \rho)]_{\text{old}} \stackrel{\text{def}}{=} \rho(\mathsf{T}[t]_{\text{old}})$ and $\mathsf{U}_\mathcal{A}[(t, \rho)]_{\text{new}} \stackrel{\text{def}}{=} \rho(\mathsf{T}[t]_{\text{new}})$.*
If $t : q \xrightarrow{l} q'$ then we write $(t, \rho) : (q, \sigma) \xrightarrow{(l,\lambda)} (q', \sigma')$, where $\sigma = \mathsf{s}[t]^{-1}; \rho$, $\lambda = \mathsf{o}[t]^{-1}; \rho$ and $\sigma' = \mathsf{d}[t]^{-1}; \rho$.

For the global states and global transitions of a HD-automaton we use notations similar to those for the components of the HD-automaton; so, the global transitions of HD-automaton $\mathcal{B}$ are denoted by $\mathsf{T}_\mathcal{B}$; also, if we consider two HD-automata $\mathcal{A}_1$ and $\mathcal{A}_2$, then their global states are denoted by G_1 and G_2 respectively.

Now we give the definition of bisimulation which is based on global states and global transitions.

Definition 15 (global bisimulation). *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two HD-automata. A* global simulation *for $\mathcal{A}_1$ and $\mathcal{A}_2$ is a relation $\mathcal{R} \subseteq G_1 \times G_2$ such that whenever $g_1 \mathcal{R} g_2$ then:*

for all $u_1 : g_1 \xrightarrow{k} g_1'$ in U_1 with $\mathsf{U}_1[u_1]_{\mathrm{new}} \cap \mathsf{G}_2[g_2] = \emptyset$ there exists some $u_2 : g_2 \xrightarrow{k} g_2'$ such that $g_1' \mathcal{R} g_2'$.

A global bisimulation *for $\mathcal{A}_1$ and $\mathcal{A}_2$ is a relation $\mathcal{R} \subseteq G_1 \times G_2$ such that both $\mathcal{R}$ is a global simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ and $\mathcal{R}^{-1}$ is a global simulation for $\mathcal{A}_2$ and $\mathcal{A}_1$.*
The HD-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are global-bisimilar *iff there exists some global bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $(q_{01}, \sigma_{01}) \mathcal{R} (q_{02}, \sigma_{02})$.*

Notice the clause "$\mathsf{U}_1[u_1]_{\mathrm{new}} \cap \mathsf{G}_2[g_2] = \emptyset$" in the definition above, that discards all those global transitions of g_1 that use as new name a name which is old in g_2. This is necessary in the global bisimulation, since names have a global identity here; in fact, this clause plays the same role of clause "$\mathrm{bn}(\mu) \cap \mathrm{fn}(p|q) = \emptyset$" in the definitions of bisimulation in π-calculus (Definition 5).

Global bisimilarity coincides with HD-bisimilarity.

Proposition 4. *Two HD-automata are HD-bisimilar if and only if they are global bisimilar.*

We now show that two π-calculus processes are bisimilar if and only if the corresponding HD-automata are bisimilar. To obtain this result we exploit the global characterization of HD-bisimulation presented in the previous section. The following is the main lemma.

Lemma 2. *Let (q, σ) be a global state of the HD-automaton $\mathcal{A}_p^{\pi_g}$ corresponding to a π-calculus process p. Then:*

- *if $q\sigma \xrightarrow{\mu} q''$ is a π-calculus transition with $\mathrm{bn}(\mu) \cap \mathrm{fn}(q\sigma) = \emptyset$, then there is some global transition $(t, \rho) : (q, \sigma) \xrightarrow{(l,\lambda)} (q', \sigma')$ of $\mathcal{A}_p^{\pi_g}$; and*
- *if $(t, \rho) : (q, \sigma) \xrightarrow{(l,\lambda)} (q', \sigma')$ is a global transition of $\mathcal{A}_p^{\pi_g}$, then there is some π-calculus transition $q\sigma \xrightarrow{\mu} q''$*

where in both cases $q'' = q'\sigma'$, and (l, λ) are related to μ as in Table 3.

Theorem 2. *Let p_1 and p_2 be π-calculus processes. Then $p_1 \sim_g p_2$ iff $\mathcal{A}_{p_1}^{\pi_g} \sim \mathcal{A}_{p_2}^{\pi_g}$.*

4 Minimization of HD-Automata

In this section we address the problem of defining minimal realizations for HD-automata. As we have already discussed for ordinary automata, having a minimal canonical representative for a class of bisimilar automata is important both from a theoretical point of view and from a practical point of view. Unfortunately enough, minimization is

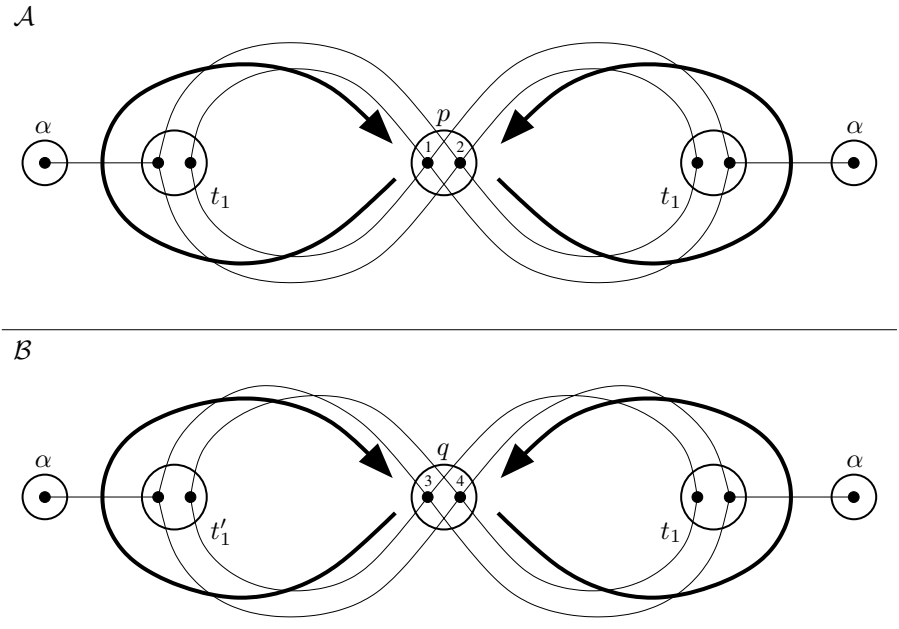

Fig. 4. Two non isomorphic minimal HD-automata

not possible on the HD-automata we introduced in Section 3. In Figure 4 we show two equivalent HD-automata: they are both "minimal", in the sense that it is not possible to reduce them further; however they are not isomorphic. In each of the HD-automata there is a single state with two names, and two transitions: each transition exhibits in the label one of the two names. The difference between the two HD-automata is that the names are switched along the transitions in HD-automaton $\mathcal{B}$, while they are not in $\mathcal{A}$. Still, the HD-automata are equivalent: their behavior is symmetric w.r.t. the two names; and in fact a bisimulation for these HD-automata is:

$$\mathcal{R} = \{\langle p, \delta, q\rangle, \langle p, \delta', q\rangle \mid \delta(1) = 3,\ \delta(2) = 4 \text{ and } \delta'(1) = 4,\ \delta'(2) = 3\}$$

The impossibility of representing explicitly the symmetry between names 1 and 2 (and 3 and 4) is precisely the cause of the impossibility of having a common minimal realization for the two HD-automata. In fact, there is no way to quotient HD-automaton $\mathcal{A}$ with respect to its greatest bisimulation $\mathcal{R}_{\mathcal{A}} = \{\langle p, \delta, p\rangle \mid \delta(1) = 2,\ \delta(2) = 1\}$.

In the following, we show how this problem can be solved by allowing symmetries on names to appear explicitly in the states of the HD-automata.

4.1 Symmetries and HD-Automata with Symmetries

In the following we define an extended version of HD-automata where each state, label, and transition of a HD-automaton is enriched by a set of names *and* by a symmetry on this set of names. We start defining *symmetries* on names and functions between them.

Definition 16 (symmetries). *Let $N \in \mathcal{N}$ be a set of names. A* symmetry *Σ on N is a set of bijections (or permutations) on N that is a group for composition; that is:*

- *$\mathrm{id}_N \in \Sigma$ (i.e., Σ contains the identity bijection);*
- *if $\sigma, \sigma' \in \Sigma$ then $\sigma;\sigma' \in \Sigma$ (i.e., Σ is closed for composition);*
- *if $\sigma \in \Sigma$ then there is some $\sigma' \in \Sigma$ such that $\sigma;\sigma' = \mathrm{id}_N$ (i.e., Σ is closed for inversion).*

We denote the set of all the symmetries on N with $\mathcal{S}ym(N)$ and with $\mathcal{S}ym_{\mathcal{N}}$ the set of all symmetries on all subsets $N \subseteq \mathcal{N}$.
For all $\Sigma \in GSNames$, we denote with $\mathrm{n}(\Sigma)$ the set N of names such that $\Sigma \in \mathcal{S}ym(N)$.

We need to extend the HD-automaton not only adding symmetries to states and transitions, but also defining correspondences between the symmetry that enrich every transition and those that enrich its source state, target state, and label. This is similar to what happens in the case of the HD-automata in Section 3: in that case the correspondences are defined by means of inverse injections; in the case of HD-automata with symmetries these inverse injections are enriches with *embeddings on symmetries*.

Let $\Sigma \in \mathcal{S}ym(N)$ and $\Sigma' \in GS(N')$ be two symmetries and let ρ be an injective function from N' to N. Assume that all the permutations of Σ also appear in Σ' via the function ρ, i.e., that $\rho;\Sigma;\rho^{-1} \subseteq \Sigma'$, where

$$\rho;\Sigma;\rho^{-1} \stackrel{\text{def}}{=} \{\rho;\sigma;\rho^{-1} \mid \sigma \in \Sigma\}.$$

Then ρ is an embedding of Σ into Σ'. We remark that $\rho;\Sigma;\rho^{-1} \subseteq \Sigma'$ can hold only if there is no permutations in Σ which exchange names in the image of ρ with names outside the image: otherwise, $\rho;\Sigma;\rho^{-1}$ would contain partial correspondences on N which are not bijections. Therefore, ρ splits N in two separated sets of names, those in the image and those outside the image. Permutations in Σ can only switch names within such sets, but cannot switch names within the image with names outside it. Notice also that the same embedding is defined, in general, by more than one bijection. In fact, we do not want to distinct between two bijections ρ and ρ' if there is some symmetry $\sigma \in \Sigma'$ such that $\rho' = \rho;\sigma$. Hence, we define an *embedding* from Σ to Σ' as a class of those equivalent bijections.

Definition 17 (embeddings on symmetries). *Let $\Sigma \in GS(N)$ and $\Sigma' \in GS(N')$ be two symmetries on $\mathcal{N}$. An* embedding *f of Σ into Σ' (written $f : \Sigma \to \Sigma'$) is a set of injections from N' to N such that:*

- *if $\rho \in f$, then $\rho;\Sigma;\rho^{-1} \subseteq \Sigma'$ (i.e., all the permutations of Σ also appear, via f, in Σ'); and*
- *if $\rho \in f$ then $f = \Sigma';\rho$ (i.e., f contains all the variants of the same embedding).*

Now we define *named sets with symmetries*: they are similar to named sets (Definition 7), but in this case the elements are enriched with symmetries on names, rather than by sets of names. Based on named sets with symmetries, we then defined HD-automata with symmetries.

Definition 18 (named sets with symmetries). *A* named set with symmetries E *is a set denoted by* E*, and a family of symmetries on* $\mathcal{N}$*, indexed by* E*, namely* $\{\mathsf{E}[e] \in \mathcal{S}ym_{\mathcal{N}}\}_{e \in E}$*, or, equivalently* $\mathsf{E}[\_]$ *is a map from* E *to* $\mathcal{S}ym_{\mathcal{N}}$*.*
Given two named sets with symmetries E *and* F*, a* named function with symmetries $\mathsf{m} : \mathsf{E} \to \mathsf{F}$ *is a function on the sets* $m : E \to F$ *and a family, indexed by* m*, of embeddings on symmetries, namely* $\{\mathsf{m}[e] : \mathsf{E}[e] \to \mathsf{F}[f]\}_{(e,f) \in m}$*:*

Definition 19 (HD-automata with symmetries). *A* HD-automaton with symmetries $\mathcal{A}$ *is defined by:*

- *a named set with symmetries* L *of* labels*;*
- *a named set with symmetries* Q *of* states*;*
- *a named set with symmetries* T *of* transitions*;*
- *a pair of named functions with symmetries* $\mathsf{s}, \mathsf{d} : \mathsf{T} \to \mathsf{Q}$*, which associate to each transition the* source *and* destination *states respectively (and embed the symmetry of the transition into the symmetries of the source and of the destination states);*
- *a named function with symmetries* $\mathsf{o} : \mathsf{T} \to \mathsf{L}$*, which associates a label to each transition (and embeds the symmetry of the transition into the symmetry of the label);*
- *an* initial state $q_0 \in \mathrm{Q}$ *and an* initial embedding $f_0 : \{\mathrm{id}_{\mathcal{N}}\} \to \mathsf{Q}[q_0]$*, that gives a global identity to the local names of* q_0*.*

In the initial embedding, $\{\mathrm{id}_{\mathcal{N}}\}$ is the symmetry on the full set on names that is composed only by the identity permutation. We remark that the initial embedding f_0 gives a global meaning to the names of the initial state q_0 only up to the symmetry $\mathsf{Q}[q_0]$ that is defined on these names.

Each HD-automaton can be "promoted" to a HD-automaton with symmetries, by associating to each state, transition, and label the symmetry consisting only of the identity permutation. As a consequence, we can easily adapt Definition 10 to map π-calculus processes into HD-automata with symmetries.

4.2 Bisimulation on HD-Automata with Symmetries

Now we introduce bisimulation on HD-automata with symmetries and describe some of its basic properties. Similarly to what happens for HD-bisimulations on basic HD-automata (Section 3.3), also a HD-bisimulation on HD-automata with symmetries is a set of triples of the form $\langle q_1, \delta, q_2 \rangle$ where q_1 and q_2 are states of the automata and δ is a partial correspondence between the names of the states.

Let us consider the HD-automata with symmetries in Figure 5. We want to check if states q_1 and q_2 are bisimilar via the bijection δ. State q_1 can perform a transition $t_1 : q_1 \xrightarrow{l} q_1'$. We cannot requite that this transition is matched by a single transition of q_2: in fact, in state q_1 there is a symmetry between names 1 and 2, so, in transition t_1 the name in the label can correspond both to name 1 and to name 2 of the source state. In state q_2 there is no symmetry between names 1 and 2, but there are two transitions, that use name 1 and 2, respectively. We consider bisimilar these two HD-automata with symmetries, proviso the target states are bisimilar according to the correspondences

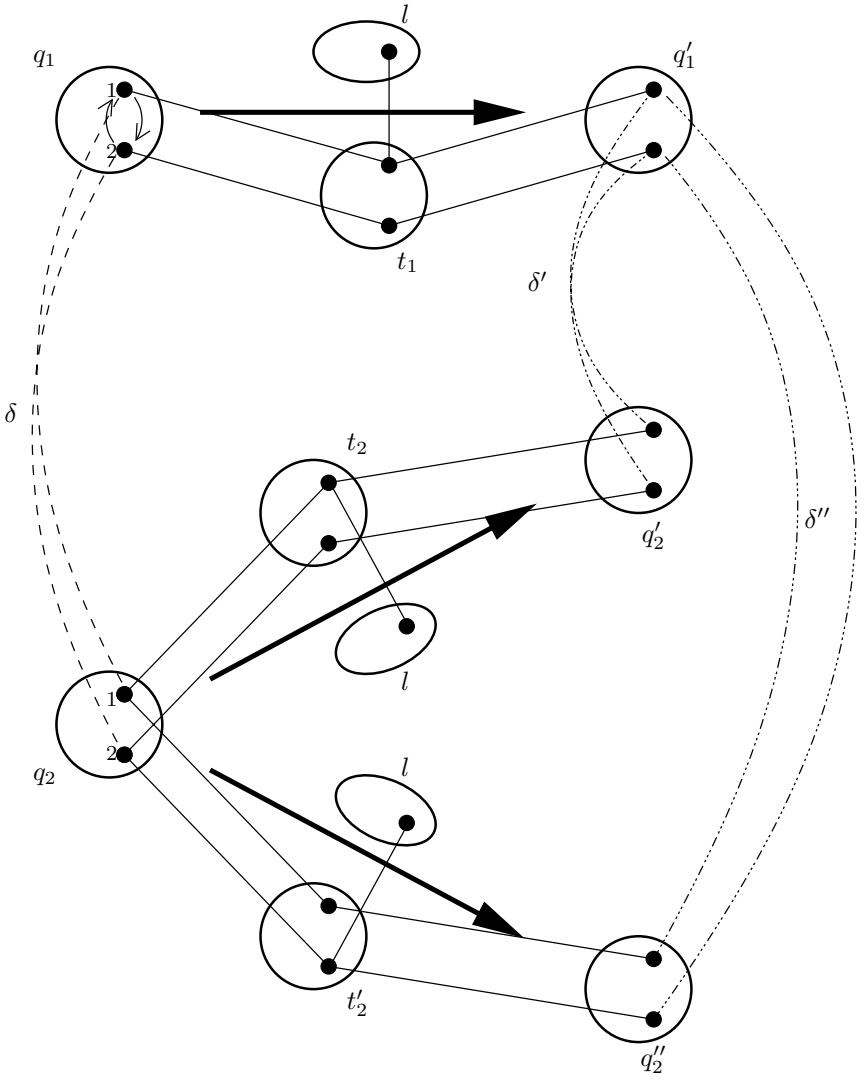

Fig. 5. A step of bisimulation on HD-automata with symmetries

δ' and δ'' represented in figure; in fact, we do not want to distinguish between the symmetries in the behaviors that are "declared" in the states and those that are implicit in the transitions of the HD-automaton with symmetries. So, transition t_1 to be matched by the pair of transitions t_2 and t_2', one for each of the symmetric behaviors of t_1. In the definition of bisimulation for HD-automata with symmetries, this is obtained by requiring that, given transition $t_1 : q_1 \xrightarrow{l} q_2$, for each injection $\alpha_1 \in \mathsf{s}_1[t_1]$ there exist a transition t_2 from q_2 and an injection $\alpha_2 \in \mathsf{s}_2[t_2]$ so that t_1 and t_2 match w.r.t. α_1 and α_2. In the general case, we have to take into account not only the symmetries of

the source state, but also those of the label and of the target state of a transition. So, a matching has to be found for a transition $t_1 : q_1 \xrightarrow{l} q_1'$ and three bijections $\alpha_1 \in \mathsf{s}_1[t_1]$, $\gamma_1 \in \mathsf{l}_1[t_1]$ and $\beta_1 \in \mathsf{d}_1[t_1]$.

Definition 20 (HDS-bisimulation). *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two HD-automata with symmetries. A* HDS-simulation *for $\mathcal{A}_1$ and $\mathcal{A}_2$ is a set of triples $\mathcal{R} \subseteq \{\langle q_1, \delta, q_2\rangle \mid q_1 \in Q_1,\ q_2 \in Q_2,\ \delta : \mathrm{n}(\mathsf{Q}_1[q_1]) \hookleftarrow \mathrm{n}(\mathsf{Q}_2[q_2])\}$ such that, whenever $\langle q_1, \delta, q_2\rangle \in \mathcal{R}$ then:*

for each $t_1 : q_1 \xrightarrow{l} q_1'$ in $\mathcal{A}_1$ and for each $\alpha_1 \in \mathsf{s}_1[t_1]$, $\gamma_1 \in \mathsf{o}_1[t_1]$ and $\beta_1 \in \mathsf{d}_1[t_1]$, there exist some $t_2 : q_2 \xrightarrow{l} q_2'$ in $\mathcal{A}_2$, some injections $\alpha_2 \in \mathsf{s}_2[t_2]$, $\gamma_2 \in \mathsf{o}_2[t_2]$ and $\beta_2 \in \mathsf{d}_2[t_2]$, some $\xi : \mathrm{n}(\mathsf{T}_1[t_1])_{\mathrm{new}} \hookleftarrow \mathrm{n}(\mathsf{T}_2[t_2])_{\mathrm{new}}$, and some $\zeta : \mathrm{n}(\mathsf{T}_1[t_1]) \hookleftarrow \mathrm{n}(\mathsf{T}_2[t_2])$ such that:

- $\zeta = (\alpha_1; \delta; \alpha_2^{-1}) \cup \xi$,
- $\gamma_1 = \zeta; \gamma_2$,
- $\langle q_1', \delta', q_2'\rangle \in \mathcal{R}$ *where δ' is such that* $\zeta = \beta_1; \delta'; \beta_2^{-1}$.

A HDS-bisimulation *for $\mathcal{A}_1$ and $\mathcal{A}_2$ is a set of triples $\mathcal{R}$ such that $\mathcal{R}$ is a HDS-simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ and $\mathcal{R}^{-1} = \{\langle q_2, \delta^{-1}, q_1\rangle \mid \langle q_1, \delta, q_2\rangle \in \mathcal{R}\}$ is a HDS-simulations for $\mathcal{A}_2$ and $\mathcal{A}_1$.*

A HDS-bisimulation for $\mathcal{A}$ is a HDS-bisimulation for $\mathcal{A}$ and $\mathcal{A}$.

The HD-automata with symmetries $\mathcal{A}_1$ and $\mathcal{A}_2$ are HDS-bisimilar *(written $\mathcal{A}_1 \sim \mathcal{A}_2$) if there exists some HDS-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\langle q_{01}, \delta, q_{02}\rangle \in \mathcal{R}$ for $\delta = \sigma_{01}; \sigma_{02}^{-1}$.*

If $\mathcal{R}$ is a HDS-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, and a pair of transitions t_1 in $\mathcal{A}_1$ and t_2 in $\mathcal{A}_2$ satisfy the bisimulation condition in definition above holds, then we write, with a light abuse of notation, that $\langle t_1, \rho, t_2\rangle \in \mathcal{R}$, where $\rho = \alpha_1; \delta; \alpha_2^{-1}$.

It is easy to see that, in the case of HD-automata with symmetries consisting only of identity permutations, this definition of HD-bisimulation coincides with the one of Definition 12.

We now investigate the basic properties of HD-bisimulation. Similarly to the HD-bisimulations defined in Section 3, also the HDS-bisimulations are closes w.r.t. union, concatenation, and operator $\widehat{-}$. As a consequence, greatest HDS-bisimulations exist: we denote with $\mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2}$ the greatest HDS-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, and with $\mathcal{R}_{\mathcal{A}}$ the greatest HDS-bisimulation for $\mathcal{A}$. Moreover, relation $\sim$ is an equivalence on HD-automata with symmetries.

In the case of HDS-bisimulations a new operator can be defined, that closes a bisimulation w.r.t. all the symmetries that are present in the states of the HD-automata.

Proposition 5. *If $\mathcal{R}$ is a HDS-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\widetilde{\mathcal{R}}$ is a HDS-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, where:*

$$\widetilde{\mathcal{R}} \stackrel{\mathrm{def}}{=} \{\langle q_1, \delta', q_2\rangle \mid \langle q_1, \delta, q_2\rangle \in \mathcal{R}, \delta' = \sigma_1; \delta; \sigma_2 \text{ and } \sigma_1 \in \mathsf{Q}_1[q_1],\ \sigma_2 \in \mathsf{Q}_2[q_2]\}.$$

4.3 Minimizing HD-Automata with Symmetries

In this section we show that, given a HD-automaton with symmetries $\mathcal{A}$, it is possible to minimize it, i.e., to define a HD-automaton with symmetries $\mathcal{A}_{\min}$ that is bisimilar to

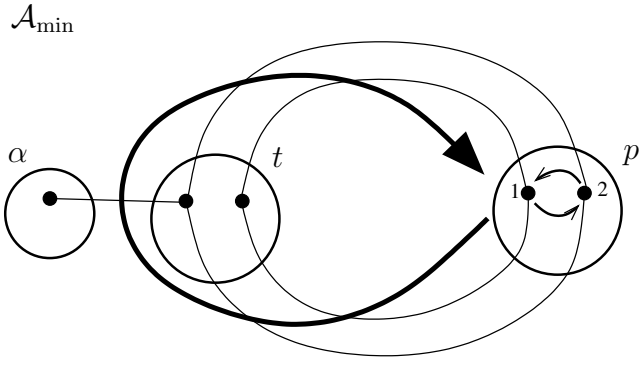

Fig. 6. A minimal realization for the HD-automata of Figure 4

$\mathcal{A}$ and that is "minimal" in the class of HD-automata bisimilar to $\mathcal{A}$ — we define below what is the meaning of "minimal".

We start by showing that the counter-example on the existence of minimal HD-automata presented at the beginning of this section (see Figure 4) does not apply to the case of HD-automata with symmetries. Indeed, the minimal HD-automaton corresponding to the two HD-automata in Figure 4 is represented in Figure 6. HD-automaton $\mathcal{A}_{\min}$ has a single state p, with one infinite repository and two distinct names 1 and 2. Moreover the symmetry associated to state p declares that names 1 and 2 can be switched without affecting the behavior. HD-automaton $\mathcal{A}_{\min}$ has one transition t, that exhibits one of the two names in the label α. Also the transition and the label have one infinite repository. In the figure, we have not represented explicitly that the infinite repositories of p, t and α are in correspondence along the transition. The possibility of declaring the symmetry on the two names 1 and 2 of state p is the key feature for obtaining a canonical minimal HD-automaton with symmetries. Indeed, this symmetry makes it possible to use a single transition t of $\mathcal{A}_{\min}$ to represent both transitions of $\mathcal{A}$ and $\mathcal{B}$ — the two transitions t_1 and t_2 in the HD-automata differ only for the choice of the name to exhibit in the action. Moreover, the symmetry between 1 and 2 makes ephemeral the fact that the two names are exchanged or not along transition t.

We start by describing the fine structure of $\mathcal{R}_{\mathcal{A}}$. This will be useful to guide the construction of the minimal automaton. First of all, relation $\mathcal{R}_{\mathcal{A}}$ is closed for concatenation, so it defines a partition on the states Q of $\mathcal{A}$; that is, relation $\equiv_{\mathcal{A}}$ is an equivalence, where

$$p \equiv_{\mathcal{A}} q \qquad \text{iff} \qquad \langle p, \delta, q\rangle \in \mathcal{R}_{\mathcal{A}} \text{ for some } \delta.$$

Consider two states $p, q \in Q$, and let $\Delta_{\mathcal{A}}(p, q)$ be the set of correspondences that exist, according to $\mathcal{R}_{\mathcal{A}}$, between the names of p and of q:

$$\Delta_{\mathcal{A}}(p, q) \stackrel{\text{def}}{=} \{\delta \mid \langle p, \delta, q\rangle \in \mathcal{R}_{\mathcal{A}}\}.$$

Let us now consider more in detail $\Delta_{\mathcal{A}}(q, q)$. It consists of a set of partial mappings on the names in $\mathsf{Q}[q]$. The fact that these mappings are partial is an evidence that state q

can contain names which do not play any important role in the future behavior: indeed, according to the definition of $\Delta_{\mathcal{A}}$, state q exhibits the same behaviors also if the identity of the names outside the partial mappings is lost. More precisely, let $\mathrm{an}_{\mathcal{A}}(q)$ be the names that appear in the domain of every partial mapping in $\Delta_{\mathcal{A}}(q,q)$. Then these are the only *active names* in state q and all other names can be safely discarded from the state, since they are not relevant for the future behaviors. The following property formalized the notion of active names and investigates some of their properties.

Proposition 6. *Let $\mathcal{A}$ be a HD-automaton, and let*

$$\underset{\mathcal{A}}{\mathrm{an}}(q) \stackrel{\mathrm{def}}{=} \bigcap_{\delta \in \Delta_{\mathcal{A}}(q,q)} \mathrm{dom}(\delta) \quad \textit{and} \quad \Delta_{\mathrm{an}\mathcal{A}}(q) \stackrel{\mathrm{def}}{=} \{\delta \cap (\underset{\mathcal{A}}{\mathrm{an}} \times \underset{\mathcal{A}}{\mathrm{an}}) \mid \delta \Delta_{\mathcal{A}}(q,q)\}$$

for all $q \in Q$. Then:

1. *$\Delta_{\mathrm{an}\mathcal{A}}(q)$ is a symmetry on $\mathrm{an}_{\mathcal{A}}(q)$;*
2. *$\mathsf{Q}[q] \cap (\mathrm{an}_{\mathcal{A}}(q) \times \mathrm{an}_{\mathcal{A}}(q)) \subseteq \Delta_{\mathrm{an}\mathcal{A}}(q,q)$;*
3. *$\Delta_{\mathcal{A}}(q) = \{\delta \in \mathsf{Q}[q] \rightharpoonup\!\!\!\!\!\rightharpoonup \mathsf{Q}[q] \mid (\delta \cap (\mathrm{an}_{\mathcal{A}}(q) \times \mathrm{an}_{\mathcal{A}}(q))) \in \Delta_{\mathrm{an}\mathcal{A}}(q)\}$.*

The results described above for the states of an HD-automaton with symmetries also hold for the transitions. More precisely, let us define

$$t \equiv_{\mathcal{A}} t' \qquad \text{iff} \qquad \langle t, \rho, t' \rangle \in \mathcal{R}_{\mathcal{A}} \text{ for some } \rho.$$

This relation turns out to be an equivalence. Moreover, by defining

$$\Delta_{\mathcal{A}}(t,t') = \{\rho \mid \langle t, \rho, t' \rangle \in \mathcal{R}_{\mathcal{A}}\},$$

the results of Proposition 6 also hold for transitions.

We are now ready to define the minimal HD-automaton corresponding to a given HD-automaton with symmetries $\mathcal{A}$. This minimal realization is obtained by replacing each class of equivalent states and transitions of $\mathcal{A}$ with a single state or transition. The names associated to states and transitions of the minimal HD-automaton are the active names and the associated symmetries are those defined by $\Delta_{\mathrm{an}\mathcal{A}}$: these, in fact, express all the symmetries that exist between the names, not only those "declared" in HD-automaton $\mathcal{A}$. We remark that it is the possibility of representing the symmetries defined by the HDS-bisimulations directly in the states of an automaton that allows for the definition of minimal HD-automata.

In the definition of the minimal HD-automaton, we denote with $[q]_{\equiv_{\mathcal{A}}}$ the equivalence classes of the states w.r.t. $\equiv_{\mathcal{A}}$; that is, $[q]_{\equiv_{\mathcal{A}}} = \{q' \mid q \equiv_{\mathcal{A}} q'\}$. We also assume that a canonical representative is defined for any such class, and we denote with $\lfloor q \rfloor_{\equiv_{\mathcal{A}}}$ the canonical representative of class $[q]_{\equiv_{\mathcal{A}}}$; that is, $\lfloor q \rfloor_{\equiv_{\mathcal{A}}} \in [q]_{\equiv_{\mathcal{A}}}$ and whenever $q \equiv_{\mathcal{A}} q'$ then $\lfloor q \rfloor_{\equiv_{\mathcal{A}}} = \lfloor q' \rfloor_{\equiv_{\mathcal{A}}}$. Similar notations are used for the transitions.

The definition of minimal HD-automaton follows.

Definition 21 (minimal HD-automaton with symmetries). *The minimal HD-automaton with symmetries $\mathcal{A}_{\min}$ for $\mathcal{A}$ is defined as follows:*

- $\mathrm{L}_{\min} = \mathrm{L}$ *and* $\mathsf{L}_{\min}[l] = \mathsf{L}[l]$ *for each* $l \in \mathrm{L}_{\min}$*;*
- $\mathrm{Q}_{\min} = \{\lfloor q \rfloor_{\equiv_{\mathcal{A}}} \mid q \in \mathrm{Q},\ q$ *reachable state*$\}$ *and* $\mathsf{Q}_{\min}[q] = \Delta_{\mathrm{an}_{\mathcal{A}}}(q, q)$ *for every* $q \in \mathrm{Q}_{\min}$*;*
- $\mathrm{T}_{\min} = \{\lfloor t \rfloor_{\equiv_{\mathcal{A}}} \mid t \in \mathrm{T},\ t$ *reachable transition*$\}$ *and* $\mathsf{T}_{\min}[t] = \Delta_{\mathrm{an}_{\mathcal{A}}}(t, t)$ *for every* $t \in \mathrm{T}_{\min}$*;*
- $\mathrm{o}_{\min}(t) = \mathrm{o}(t)$ *and* $\mathsf{o}_{\min}[t] = \mathsf{o}[t]$ *for every* $t \in \mathrm{T}_{\min}$*;*
- $\mathrm{s}_{\min}(t) = \lfloor \mathrm{s}(t) \rfloor_{\equiv_{\mathcal{A}}}$ *and* $\mathsf{s}_{\min}[t] = \big\{\sigma \mid \sigma = (\sigma'; \sigma'') \cap (\mathrm{an}_{\mathcal{A}}(\lfloor \mathrm{s}(t) \rfloor_{\equiv_{\mathcal{A}}}) \times \mathrm{an}_{\mathcal{A}}(t))$ *for* $\sigma' \in \Delta_{\mathcal{A}}(s(t), \lfloor s(t) \rfloor_{\equiv_{\mathcal{A}}})$ *and* $\sigma'' \in \mathrm{s}(t)\big\}$ *for every* $t \in \mathrm{T}_{\min}$*;*
- $\mathrm{d}_{\min}(t) = \lfloor \mathrm{d}(t) \rfloor_{\equiv_{\mathcal{A}}}$ *and* $\mathsf{d}_{\min}[t] = \big\{\sigma \mid \sigma = (\sigma'; \sigma'') \cap (\mathrm{an}_{\mathcal{A}}(\lfloor \mathrm{d}(t) \rfloor_{\equiv_{\mathcal{A}}}) \times \mathrm{an}_{\mathcal{A}}(t))$ *for* $\sigma' \in \Delta_{\mathcal{A}}(d(t), \lfloor d(t) \rfloor_{\equiv_{\mathcal{A}}})$ *and* $\sigma'' \in \mathrm{d}(t)\big\}$ *for every* $t \in \mathrm{T}_{\min}$*;*
- $q_{0\,\min} = \lfloor q_0 \rfloor_{\equiv_{\mathcal{A}}}$ *and* $f_{0\,\min} = \big\{\sigma \mid \sigma = (\sigma'; \sigma'') \cap (\mathrm{an}_{\mathcal{A}}(\lfloor q_0 \rfloor_{\equiv_{\mathcal{A}}}) \times \mathcal{N})$ *for* $\sigma' \in \Delta_{\mathcal{A}}(q_0, q_{\min\,0})$ *and* $\sigma'' \in f_0\big\}$.

In the definition above, by reachable states and reachable transitions we mean those states and transitions that can be reached from the initial state following the transitions in the automaton.

A first, important property of minimal HD-automata is that $\mathcal{A}_{\min}$ is HDS-bisimilar to the original HD-automaton $\mathcal{A}$.

Proposition 7. *Let $\mathcal{A}$ be a HD-automaton with symmetries. Then $\mathcal{A} \sim \mathcal{A}_{\min}$.*

Minimal HD-automata are unique, up to isomorphism, for each class of bisimilar HD-automata.

Theorem 3. *Let $\mathcal{A}$ and $\mathcal{B}$ be two HD-automata with symmetries. Then $\mathcal{A} \sim \mathcal{B}$ if and only if $\mathcal{A}_{\min}$ and $\mathcal{B}_{\min}$ are isomorphic.*

The obtained HD-automaton $\mathcal{A}_{\min}$ is *minimal* since it has the minimum number of states and of transitions among the HD-automata that are bisimilar to $\mathcal{A}$; moreover, it has the maximum set of symmetries in these states and transitions. Notice that increasing the symmetries in states and transitions is considered a step toward minimization: in fact, if larger symmetries are present, then a smaller number of transitions is sufficient to represent the same behaviors. If we collapse further states and transitions of $\mathcal{A}_{\min}$, or if we enlarge symmetries of its states and transitions, a non-equivalent HD-automaton is obtained.

5 Concluding Remarks

We have presented History-Dependent Automata and we have shown that they are an operational model particularly adequate for named calculi such as the π-calculus. An important property that holds only for HD-automata enriched with symmetries is the existence, in each class of equivalent HD-automata, of a minimal representative. As it happens for ordinary automata, this minimal HD-automaton can be considered the semantic object corresponding to the class of equivalent HD-automata.

Several results on HD-automata have not been covered in this paper. An extended version of HD-automata with symmetries has been defined in [Pis99], in order to capture the early and late semantics of the π-calculus. In [MP99] a particular variant of

HD-automata, namely HD-automata with *negative transitions*, is proposed in order to deal with the asynchronous π-calculus [HT91, ACS98]. In [MP00] a co-algebraic semantics for the π-calculus is defined. It is based on the idea of extending states and transitions with an algebra of names and symmetries. A variant of HD-automata is shown to come out naturally as a compact representation of the co-algebraic models. Finally, a categorical characterization of HD-automata and of HD-bisimulation is given in [MP98b, MP98a].

HD-automata also provide the core of HAL [FFG+97, GR97], a verification environment for concurrent systems described in the π-calculus and other named calculi: HD-automata allow for a compact representation of the behaviors of these concurrent systems, and can be used in the algorithms as a common format for named calculi.

References

[ACS98] R. Amadio, I. Castellani, and D. Sangiorgi. On bisimulations for the asynchronous π-calculus. *Theoretical Computer Science*, 192(2):291–324, 1998.

[Dam97] M. Dam. On the decidability of process equivalences for the π-calculus. *Theoretical Computer Science*, 183(2):215–228, 1997.

[FFG+97] G. Ferrari, G. Ferro, S. Gnesi, U. Montanari, M. Pistore, and G. Ristori. An automata based verification environment for mobile processes. In *Proc. TACAS'97*, volume 1217 of *LNCS*. Springer Verlag, 1997.

[GR97] S. Gnesi and G. Ristori. A model checking algorithm for π-calculus agents. In *Proc. ICTL'97*. Kluwer Academic Publishers, 1997.

[HT91] K. Honda and M. Tokoro. On asynchronous communication semantics. In *Proc. ECOOP'91*, volume 612 of *LNCS*. Springer Verlag, 1991.

[IP96] P. Inverardi and C. Priami. Automatic verification of distributed systems: The process algebras approach. *Formal Methods in System Design*, 8(1):1–37, 1996.

[Mad92] E. Madelaine. Verification tools for the CONCUR project. *Bullettin of the EATCS*, 47:110–126, 1992.

[Mil89] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[Mil93] R. Milner. The polyadic π-calculus: a tutorial. In *Logic and Algebra of Specification*, volume 94 of *NATO ASI Series F*. Springer Verlag, 1993.

[MP98a] U. Montanari and M. Pistore. History dependent automata. Technical Report TR-11-98, Università di Pisa, Dipartimento di Informatica, 1998.

[MP98b] U. Montanari and M. Pistore. An introduction to history dependent automata. In *Proc. Second Workshop on Higher-Order Operational Techniques in Semantics (HOOTS II)*, volume 10 of *ENTCS*. Elsevier, 1998.

[MP99] U. Montanari and M. Pistore. Finite state verification for the asynchronous π-calculus. In *Proc. TACAS'99*, LNCS. Springer Verlag, 1999.

[MP00] U. Montanari and M. Pistore. π-calculus, structured coalgebras and minimal hd-automata. In *Proc. MFCS 2000*, volume 1893 of *LNCS*. Springer Verlag, 2000.

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

[MPW93] R. Milner, J. Parrow, and D. Walker. Modal logic for mobile processes. *Theoretical Computer Science*, 114(1):149–171, 1993.

[Par80] D. Park. *Concurrency and Automata on Infinite Sequences*, volume 104 of *LNCS*. Springer Verlag, 1980.

[Pis99] M. Pistore. *History Dependent Automata*. PhD thesis, Università di Pisa, Dipartimento di Informatica, 1999.

[San93a] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms*. PhD thesis, University of Edinburgh, 1993.

[San93b] D. Sangiorgi. From π-calculus to higher-order π-calculus – and back. In *Proc. TAPSOFT'93*, volume 668 of *LNCS*. Springer Verlag, 1993.

[Wal95] D. Walker. Objects in the π-calculus. *Information and Computation*, 116(2):253–271, 1995.