ITC irst

CENTRO PER LA RICERCA
SCIENTIFICA E TECNOLOGICA

38050 Povo (Trento), Italy
Tel.: +39 0461 314312
Fax: +39 0461 302040
e−mail: prdoc@itc.it − url: http://www.itc.it

HISTORY DEPENDENT AUTOMATA

Montanari U., Pistore M.

December 2001

Technical Report # 0112−14

History-Dependent Automata

Ugo Montanari
University of Pisa
Corso Italia 40, 56100 Pisa, Italy
ugo@di.unipi.it

Marco Pistore
ITC-IRST
Via Sommarive 18, 38050 Povo (Trento), Italy
pistore@irst.itc.it

Abstract

In this paper we present *history-dependent automata* (HD-automata in brief). They are an extension of ordinary automata that overcomes their limitations in dealing with history-dependent formalisms. In a history-dependent formalism the actions that a system can perform carry information generated in the past history of the system. The most interesting example is π-calculus: channel names can be created by some actions and they can then be referenced by successive actions. Other examples are CCS with localities and the history-preserving semantics of Petri nets.

Ordinary automata are an unsatisfactory operational model for these formalisms: infinite automata are obtained for all the systems with infinite computations, even for very simple ones; moreover, the ordinary definition of bisimulation does not apply in these cases, thus preventing the reusage of standard theories and algorithms.

In this paper we show that HD-automata are an adequate model for the history-dependent formalisms. We present translations of π-calculus, CCS with localities and Petri nets into HD-automata; and we show that finite HD-automata are obtained for significant classes of systems with infinite computations. We also define HD-bisimulation and show that it captures the standard equivalences of the considered history-dependent formalisms. Moreover, we prove that HD-automata can be minimized, and that the same minimal HD-automaton is associated to each class of bisimilar HD-automata. Finally, we provide a categorical definition of HD-automata and of HD-bisimulation (by exploiting open maps).

Contents

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 an agent 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 CCS agent — 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 *history-dependent calculi*, in particular, infinite-state transition systems are generated instead, also by very simple processes. A calculus is *history-dependent* if the observations labelling the transitions of an agent may refer to previous transitions of the same agent, expressing in this manner a dependence from them. For instance, in the case of CCS with localities [BCHK93], each transition exhibits, in addition to an action, also the location in which the action is supposed to happen, and new locations are generated by fork transitions. A similar case is CCS with causality [DDNM90, DD89, Kie94]. Another quite interesting example 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 [San93b]. New channels between the process and the environment can be created at run-time and referred to in subsequent communications. It is thus evident the history-dependent character of π-calculus.

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 agents, 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 aim of this paper is to show that the ideal situation of ordinary automata can (at least in part) be recovered also in the field of history-dependent 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 history-dependent 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 history-dependent processes. In particular, π-calculus agents 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 agents 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 agents if and only if the agents are bisimilar according to the ordinary π-calculus bisimilarity relation. These results do not hold only for the π-calculus. A similar mapping exists, for instance, for CCS with localities [BCHK93]. HD-automata can be also applied to concurrent formalisms outside the field of process calculi: for instance, we show that they can be applied to

Petri nets, for representing the history-preserving semantics of the nets [BDKP91].

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 order to stress that naturalness of HD-automata and HD-bisimulation, we show that it is possible to define them in a very simple way in a categorical framework. A classical categorical definition of ordinary automata is extended to HD-automata: essentially, the categorical construction is the same, but we use the category of *named sets* as the base category — it was the category of sets in the case of ordinary automata. *Open maps* bisimulation [JNW96] — an uniform approach to define equivalences for concurrent models presented in a categorical framework — can be applied also to HD-automata, thus obtaining a categorical definition of HD-bisimulation. Minimization of HD-automata is captured very naturally in the categorical framework: the minimal model is the final model in the sub-category of equivalent HD-automata.

Outline. CCS and some of the basic results on ordinary automata are briefly presented in Section 2; this section will be used as comparison term for the results on HD-automata. In Section 3 the π-calculus is presented and the problems of using ordinary automata to deal with it are discussed.

In order to have a simpler presentation, we define two families of HD-automata. Section 4 introduces a simplified version of HD-automata, called Basic HD-automata. They can model only some of the history-dependent calculi we consider — notably, they are not adequate for the *early* and *late* π-calculus semantics — and they do not admit minimal models. Section 4 also defines bisimulation on HD-automata and presents the translation of the *ground* semantics of π-calculus agents to HD-automata. In Sections 5 we briefly describe two other history-dependent formalisms that can be represented by Basic HD-automata — namely CCS with localities and Petri nets with history-preserving semantics.

Section 6 describes the complete version of HD-automata, namely HD-automata with Symmetries. They are adequate not only for all the history-dependent calculi already considered for Basic HD-automata, but also for the early and late semantics of the π-calculus. Moreover, they allow for minimization. In Section 7 the categorical characterizations of HD-automata and of HD-bisimulations are presented.

In Section 8 we describe in short some other formalisms that can be captured by HD-automata and some possible extensions, while in Section 9 we propose some concluding remarks.

Previous works. This paper resumes and completes preliminary results on the HD-automata that have been reported in previous papers by the authors. The first, primitive notion of HD-automata appears in [MP95] under the name of π-automata; they are used in an algorithm for checking bisimilarity of π-calculus agents without matching, as a compact algorithmical structure for representing the operational semantics of the agents. There was no notion of bisimulation on the π-automata.

Simplified versions of the HD-automata also appeared in [MPY96], by Daniel Yankelevich and the authors, for the CCS with localities, in [MP97b] for Petri nets, and in [MP97a] for a class of partial-order systems, that includes CCS with localities and Petri nets. HD-automata and HD-bisimulations defined in [MPY96, MP97b, MP97a] are much simpler than those needed for π-calculus, since there is no input of names. Also, the categorical definition of HD-automata and HD-bisimulation is not present in those papers. A categorical characterization of HD-automata is given in [MP98b, MP98a]. This categorical characterization only covers Basic HD-automata.

Other works extend the theory of HD-automata in specific directions. 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, an extended presentation of HD-automata can be found in the PhD Thesis of the second author [Pis99].

2 Ordinary automata and CCS

Automata are a very convenient operational model for process calculi like CCS. In this section we introduce the basic results on automata and their applications to CCS. In the following sections we will often refer to the results presented here for CCS and ordinary automata to draw a comparison with the results which hold for history-dependent calculi and HD-automata.

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 2.1 (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 2.2 *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 2.3 (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 2.4 (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 CCS

The version of CCS we present here is slightly different from the classical one [Mil89] and follows some suggestions of π-calculus. The differences with the classical definition of CCS are not substantial and are introduced to have a more uniform presentation of the various process calculi that appear in this paper.

Let Λ be a set of *atomic actions*, or *channels* (ranged over by $\alpha, \beta, \ldots$), and Var be a finite set of agent identifiers (ranged over by $A, B, \ldots$). CCS agents (ranged over by $p, q, \ldots$) are defined by the syntax:

$$p \ ::= \ \mathbf{0} \ \Big| \ \mu.p \ \Big| \ p|p \ \Big| \ p{+}p \ \Big| \ (\nu\alpha)\, p \ \Big| \ A$$

where *prefixes* (or *actions*) μ are defined by the syntax:

$$\mu \ ::= \ \tau \ \Big| \ \alpha \ \Big| \ \bar{\alpha}.$$

For each agent identifier A there is a definition $A \overset{\text{def}}{=} p_A$ and we assume that each agent identifier in p_A is in the scope of a prefix (guarded recursion).

As usual, $\mathbf{0}$ is the terminated agent; $\mu.p$ prefixes action μ to agent p; $p|q$ is the parallel composition with synchronization of agents p and q, whereas $p{+}q$ is the nondeterministic choice. Following the notation of π-calculus, the restriction of action α in agent p is represented by $(\nu\alpha)\, p$, rather than by the conventional $p\backslash\alpha$. Finally, infinite behaviors are obtained by means of agent identifiers and of their definitions; also in this case, we prefer this solution to the $\mathbf{rec}\ x.p$ construct for analogy with the π-calculus. The set of definitions is assumed to be finite, to avoid agents with an "infinite program".

We give sum and parallel composition the lowest syntactic precedence among the operators. In an agent $\alpha.\mathbf{0}$, we often omit the trailing $\mathbf{0}$.

We now introduce a *structural congruence* in the style of the Chemical Abstract Machine [BB92] and of the π-calculus [Mil93]. This structural congruence allows us to identify all the agents 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)	$(\nu\alpha)\, p \equiv (\nu\beta)\, (p\{^{\beta}/_{\alpha}\})$ if β 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)	$(\nu\alpha)\, \mathbf{0} \equiv \mathbf{0} \qquad (\nu\alpha)\, (\nu\beta)\, p \equiv (\nu\beta)\, (\nu\alpha)\, p$
	$(\nu\alpha)\, (p|q) \equiv p|(\nu\alpha)\, q$ if α does not appear in p

where agent $p\{^{\beta}/_{\alpha}\}$ is obtained from p by replacing all the free occurrences of α with β.

The structural congruence is exploited in the definition of the operational semantics, for instance commutativity of $\_|\_$ is exploited to avoid the duplication of the rules for the parallel composition.

The structural congruence is also necessary in practice to obtain finite state representations for classes of agents. 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, $(\nu\alpha)\, p \equiv (\nu\alpha)\, (p|\mathbf{0}) \equiv p|(\nu\alpha)\, \mathbf{0} \equiv p|\mathbf{0} \equiv p$.

By exploiting the structural congruence $\equiv$, each CCS agent 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

$$s \ ::= \ \mu.p \ \Big| \ p{+}p \ \Big| \ A$$

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

The transitions that CCS agents can perform are defined by the axiom schemata and inference rules of Table 1. Since CCS agents are defined up to structural congruence, the following rule is implicitly assumed:

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

It is easy to associate an automaton to a CCS agent.

Definition 2.5 (from CCS agents to automata) *The automaton $\mathcal{A}_p^{\mathrm{CCS}}$ corresponding to the CCS agent p is defined as follows:*

[PREF] $\mu.p \xrightarrow{\mu} p$	[SUM] $\dfrac{p_1 \xrightarrow{\mu} p'}{p_1{+}p_2 \xrightarrow{\mu} p'}$
[PAR] $\dfrac{p_1 \xrightarrow{\mu} p_1'}{p_1\|p_2 \xrightarrow{\mu} p_1'\|p_2}$	[COMM] $\dfrac{p_1 \xrightarrow{\bar{\alpha}} p_1' \quad p_2 \xrightarrow{\alpha} p_2'}{p_1\|p_2 \xrightarrow{\tau} p_1'\|p_2'}$
[RES] $\dfrac{p \xrightarrow{\mu} p'}{(\nu\alpha)\, p \xrightarrow{\mu} (\nu\alpha)\, p'}$ if $\mu \neq \alpha, \bar{\alpha}$	[IDE] $\dfrac{p_A \xrightarrow{\mu} p'}{A \xrightarrow{\mu} p'}$ if $A \overset{\text{def}}{=} p_A$

Table 1: Operational semantics for CCS

- *the set of the labels is given by all CCS actions;*
- *$p \in Q$ is the initial state;*
- *if $q \in Q$ and $q \xrightarrow{\mu} q'$ is a CCS transition, then $q' \in Q$ and $t = (q, \mu, q') \in T$, with $s(t) = q$, $d(t) = q'$ and $o(t) = \mu$.*

Finite-state automata are obtained for important classes of agents that have infinite behaviors. In particular, if there is a bound for the number of active sequential components of all the derivatives of a given agent, then a finite-state automaton is obtained from that agent. Conversely, if an agent can activate an unbounded number of active sequential components during its evolutions, then it is not possible to represent it with a finite-state automaton.

Definition 2.6 (finitary agents) *The* degree of parallelism $\deg(p)$ *of an agent p is defined as*

$$\begin{array}{rcl@{\qquad}rcl} \deg(\mathbf{0}) & = & 0 & \deg(\mu.p) & = & 1 \\ \deg((\nu\alpha)\, p) & = & \deg(p) & \deg(p|q) & = & \deg(p) + \deg(q) \\ \deg(p{+}q) & = & \max\{\deg(p), \deg(q)\} & \deg(A) & = & 1 \end{array}$$

A CCS agent p is finitary *if* $\max\{\deg(p') \mid p \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_i} p'\} < \infty$.

Proposition 2.7 *Let p be a finitary CCS agent. Then the automaton A_p^{CCS} is finite.*[1]

We would like to remark that it is only semidecidable whether a CCS agent is finitary. In fact, this problem is equivalent to the problem of deciding whether a given Turing machine needs only a finite tape.

A syntactical condition which implies that an agent is finitary is the absence of parallel compositions in the bodies of recursive definitions. These agents have been called *finite-state* in the literature; we prefer to follow the terminology adopted in π-calculus, and to call them *finite control* [Dam97]. In fact, the name "finite-state" is, in our opinion, misleading, since finite-state automata are obtained, according to Definition 2.5, also for non-finite-state agents, like $B \overset{\text{def}}{=} (\nu\delta)\, (a.(b.\delta.d.B|c.\bar{\delta}.\mathbf{0}))$.

Definition 2.8 (finite control) *CCS agent p has a* finite control *if no parallel composition appears in the recursive definitions used by p.*

Bisimulation equivalence on CCS agents is obtained by specializing Definition 2.3 to CCS transitions: two CCS agents are bisimilar if and only if the corresponding automata are bisimilar. Also the results on the existence of minimal automata transfer to CCS: it is possible to associate to each CCS agent a canonical, minimal automaton, so that bisimilar agents correspond to the same canonical automaton.

3 The π-calculus

In this section we describe the π-calculus [MPW92, Mil93], an extension of CCS 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 richer expressive power that CCS: 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. We consider only the *monadic* π-calculus, and we concentrate on the *ground* and on the *early* variants of its semantics.

[1]To obtain this result, the structural axioms are necessary, since they allow for a garbage collecting of terminated components and unused restrictions.

3.1 Syntax

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

$$p ::= \mathbf{0} \;\Big|\; \pi.p \;\Big|\; p|p \;\Big|\; p{+}p \;\Big|\; (\nu x)\, p \;\Big|\; [x{=}y]p \;\Big|\; A(x_1, \dots, x_n)$$

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 agent 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 $A(y_1, \dots, y_n) \overset{\mathrm{def}}{=} p_A$ (with y_i all distinct and $\mathrm{fn}(p_A) \subseteq \{y_1, \dots, y_n\}$); we assume that, whenever A is used, its arity n is respected. Finally we require that each agent 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. It is similar to that of CCS. The most important difference is in the prefixes. 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. The *matching* $[x{=}y]p$ represents a guard for agent p: agent p is enabled only if names x and y coincide.

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 $i = 1, \dots, n$ and that is the identity on the other names.

We define π-calculus agents up to a *structural congruence* $\equiv$, as done for CCS in Section 2.2; the equivalences are those for CCS plus the following new rule that deals with matching:

$$\textbf{(match)} \quad [x{=}x]p \equiv p \qquad [x{=}y]\mathbf{0} \equiv \mathbf{0}$$

Here we have presented the *monadic* version of π-calculus, where a single name in sent or received in any communication. There is also a *polyadic* version of π-calculus, where tuples of names can be communicated: in this case, the output and input prefixes are $\overline{x}\langle y_1, y_2, \dots, y_n\rangle$ and $x(y_1, y_2, \dots, y_n)$, respectively. In [Mil93] it is shown that the polyadic prefixes can be encoded with monadic prefixes: essentially a polyadic communication is represented by a sequence of monadic communications; all these communications occur on a private channel, that is created on purpose to this communication, to avoid interferences with other polyadic communications. Here we consider only the monadic variants of π-calculus, since the definitions are simpler in this case. All the results, however, scale up to the polyadic π-calculus in the expected way.

Often, in π-calculus infinite behaviors are obtained by means of a replication, or bang, operator $!p$, rather than by means of recursive definitions. Agent $!p$ can be intuitively explained as an infinite copies of agent p in parallel. The two methods for defining infinite behaviors have the same expressive power: each of them can be encoded in the other at the cost of additional τ actions. Also in this case, the results do not depend on the chosen method. However, if the bang operator is used, it is difficult to identify a syntactic class of agents that have a finite control (Definition 2.8): in the case of recursive definitions, in fact, if no parallel composition appears inside the recursive definitions, then clearly the number of active parallel components cannot grow unboundedly. If replication is used, however, even very simple agents like $p = !x(y).\bar{z}y$ can activate an unbounded numbed of parallel components.

3.2 The early semantics

The early semantics of π-calculus was first introduced in [MPW93], but we present here a slightly simplified version, following in part the style proposed by [San93a] and [Mil93] for the polyadic π-calculus.

The *early actions* that an agent can perform are defined by the following syntax:

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

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

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 2.

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

Table 2: Free and bound names of π-calculus actions

[TAU] $\tau.p \xrightarrow{\tau} p$	[OUT] $\bar{x}y.p \xrightarrow{\bar{x}y} p$
[IN] $x(y).p \xrightarrow{xz} p\{z/y\}$	[SUM] $\dfrac{p_1 \xrightarrow{\mu} p'}{p_1+p_2 \xrightarrow{\mu} p'}$
[COMM] $\dfrac{p_1 \xrightarrow{\bar{x}y} p_1' \quad p_2 \xrightarrow{xy} p_2'}{p_1 \mid p_2 \xrightarrow{\tau} p_1' \mid p_2'}$	[PAR] $\dfrac{p_1 \xrightarrow{\mu} p_1'}{p_1 \mid p_2 \xrightarrow{\mu} p_1' \mid p_2}$ if $\mathrm{bn}(\mu) \cap \mathrm{fn}(p_2) = \emptyset$
[OPEN] $\dfrac{p \xrightarrow{\bar{x}y} p'}{(\nu y)\, p \xrightarrow{\bar{x}(y)} p'}$ if $x \neq y$	[CLOSE] $\dfrac{p_1 \xrightarrow{\bar{x}(y)} p_1' \quad p_2 \xrightarrow{xy} p_2'}{p_1 \mid p_2 \xrightarrow{\tau} (\nu y)\,(p_1' \mid p_2')}$ if $y \notin \mathrm{fn}(p_2)$
[RES] $\dfrac{p \xrightarrow{\mu} p'}{(\nu x)\, p \xrightarrow{\mu} (\nu x)\, p'}$ if $x \notin \mathrm{n}(\mu)$	[IDE] $\dfrac{p_A\{y_1/x_1 \cdots y_n/x_n\} \xrightarrow{\mu} p'}{A(y_1, \dots, y_n) \xrightarrow{\mu} p'}$ if $A(x_1, \dots, x_n) \overset{\mathrm{def}}{=} p_A$

Table 3: Early operational semantics of π-calculus

The transitions for the *early operational semantics* are defined by the axiom schemata and the inference rules of Table 3. 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, in the case of the π-calculus, the actions an agent can perform are different from the prefixes. This happens due to the free input and to the bound output actions. In the case of the input, the prefix has the form $x(y)$, while the action has the form xz; this different notation is used to remark that, while y is a formal variable, name z is the effectively received value. The bound output 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 agent and the environment: this phenomenon is called *name extrusion*.

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

Definition 3.1 (early bisimulation) *A relation $\mathcal{R}$ over agents is an* early 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 early bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are early simulations.*
Two agents p and q are early bisimilar, *written $p \sim_e q$, if $p \;\mathcal{R}\; q$ for some early 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 agents. This clause is necessary since equivalent agents 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. For instance, an infinite number of transitions correspond even to very simple agents, like $p = x(y).\bar{y}z.\mathbf{0}$: in fact, this agent can perform an infinite number of different input transitions $p \xrightarrow{xw} \bar{w}z.\mathbf{0}$, corresponding to all the possible choices of $w \in \mathcal{N}$. It is clear that, except for x and z, which are the free names of p, all the other names are indistinguishable as input values for the future behavior of p. However, this fact is not reflected in the operational semantics.

Also 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

[PREF] $\pi.p \xrightarrow{\pi} p$	[SUM] $\dfrac{p_1 \xrightarrow{\mu} p'}{p_1+p_2 \xrightarrow{\mu} p'}$
[COMM] $\dfrac{p_1 \xrightarrow{\bar{x}y} p_1' \quad p_2 \xrightarrow{x(z)} p_2'}{p_1 \mid p_2 \xrightarrow{\tau} p_1' \mid (p_2'\{y/z\})}$	[PAR] $\dfrac{p_1 \xrightarrow{\mu} p_1'}{p_1 \mid p_2 \xrightarrow{\mu} p_1' \mid p_2}$ if $\mathrm{bn}(\mu) \cap \mathrm{fn}(p_2) = \emptyset$
[OPEN] $\dfrac{p \xrightarrow{\bar{x}y} p'}{(\nu y)\,p \xrightarrow{\bar{x}(y)} p'}$ if $x \neq y$	[CLOSE] $\dfrac{p_1 \xrightarrow{\bar{x}(y)} p_1' \quad p_2 \xrightarrow{x(y)} p_2'}{p_1 \mid p_2 \xrightarrow{\tau} (\nu y)\,(p_1' \mid p_2')}$
[RES] $\dfrac{p \xrightarrow{\mu} p'}{(\nu x)\,p \xrightarrow{\mu} (\nu x)\,p'}$ if $x \notin \mathrm{n}(\mu)$	[IDE] $\dfrac{p_A\{y_1/x_1 \cdots y_n/x_n\} \xrightarrow{\mu} p'}{A(y_1,\ldots,y_n) \xrightarrow{\mu} p'}$ if $A(x_1,\ldots,x_n) \overset{\text{def}}{=} p_A$

Table 4: Ground operational semantics of π-calculus

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 3.1 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].

3.3 The ground semantics

The ground semantics of the π-calculus differs from the early semantics just considered in the fact that *bound input* transitions are considered rather than *free* inputs. So, according to the early semantics, agent $x(y).p$ can perform free input transitions

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

for each name z, while, according to the ground semantics, agent $x(y).p$ can perform bound input transitions

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

only if z is fresh, i.e., $z \notin \mathrm{fn}(x(y).p)$.

Ground bisimilarity is easy to check[2]. 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 an agent. For instance,

$$x(y).(\bar{y}y.\mathbf{0}|z(w).\mathbf{0}) \not\sim_e x(y).(\bar{y}y.z(w).\mathbf{0} + z(w).\bar{y}y.\mathbf{0})$$

since, performing the free input action xz we obtain

$$\bar{z}z.\mathbf{0}|z(w).\mathbf{0} \not\sim_e \bar{y}y.z(w).\mathbf{0} + z(w).\bar{y}y.\mathbf{0}$$

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

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

since the reception of the already existing name z is not allowed in the ground semantics.

The *ground actions* that an agent 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*, *bound input*, *free output* and *bound output* actions.

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 2.

The transitions for the *ground operational semantics* are defined by the axiom schemata and the inference rules of Table 4.

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

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

Definition 3.2 (ground bisimulation) *A relation $\mathcal{R}$ over agents 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 early simulations.*
Two agents p and q are ground bisimilar*, written $p \sim_g q$, if $p \; \mathcal{R} \; q$ for some ground bisimulation $\mathcal{R}$.*

4 Basic history-dependent automata

Ordinary automata are successful for CCS-like languages. For more sophisticated languages, 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 a simple version of HD-automata, called *Basic HD-automata*. They are sufficient to deal with some of the existing history-dependent formalisms. The paradigmatic example we use in this section to illustrate HD-automata is the ground semantics of π-calculus. In this case, the names represent the communication channels. Other examples are CCS with localities (in this case, the names are the localities where the execution happens) and history preserving semantics of Petri nets (here the names correspond to the events of a computation). We will consider them in Section 5.

The simple mechanism for dealing with names that is introduced in this section, however, is not sufficient for all the history-dependent formalisms we are interested in. For instance it does not capture the early π-calculus semantics. In Section 6 we will present a more sophisticated version of HD-automata that works also for this π-calculus semantic.

4.1 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 4.1 *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 $f : A \leftharpoondown\!\!\rightharpoonup B$ *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 $f : A \hookrightarrow B$ *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 $f : A \hookleftarrow\!\!\rightharpoonup B$ *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 4.2 (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* $\{\mathsf{m}[e] : \mathsf{E}[e] \hookleftarrow\!\!\rightharpoonup \mathsf{F}[f]\}_{(e,f) \in m}$:

$$\begin{tikzcd}
\mathsf{E} \arrow[d, "\mathsf{m}"] & \ni & e \arrow[d, mapsto, "m"] & \mathsf{E}[e] \\
\mathsf{F} & \ni & f & \mathsf{F}[f] \arrow[u, "\mathsf{m}[e]"']
\end{tikzcd}$$

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 2.1), but named sets and named functions are use rather than plain sets and functions.

Definition 4.3 (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 $\sigma_0 : \mathsf{Q}[q_0] \hookrightarrow \mathcal{N}$ *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.*

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 $\mathsf{s}[t] : \mathsf{T}[t] \hookleftarrow\!\!\rightharpoonup \mathsf{Q}[q]$ embeds, by means of an inverse injection, the names of q into the names of t, whereas $\mathsf{d}[t] : \mathsf{T}[t] \hookleftarrow\!\!\rightharpoonup \mathsf{Q}[q']$ 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.

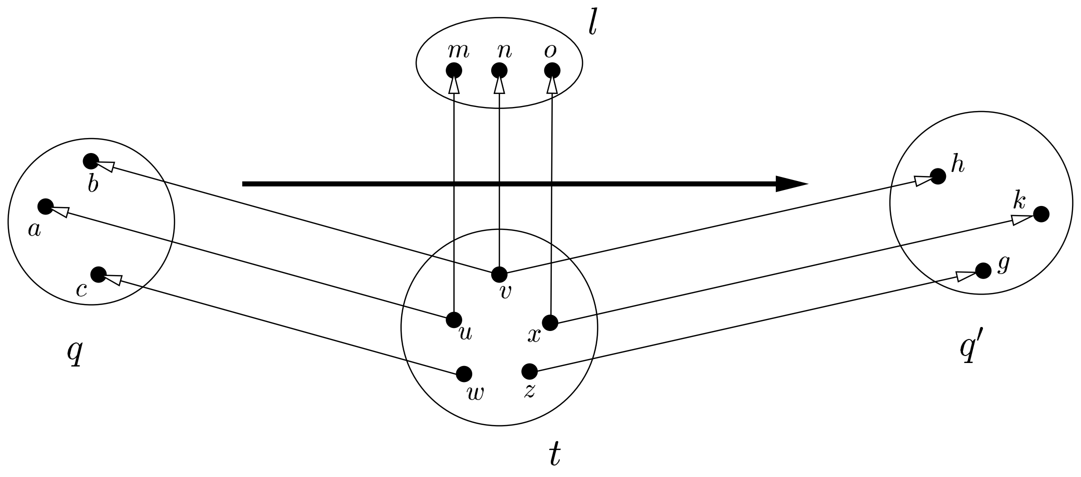

Figure 1: A transition $t : q \xrightarrow{l} q'$ of a HD-automaton

4.1.1 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 $\mathtt{tau}$, one name (n) is associated to $\mathtt{out}_2$, and two names (n_{sub} and n_{obj}) are associated to $\mathtt{bin}$, $\mathtt{out}$ and $\mathtt{bout}$.

In order to associate a HD-automaton to a π-calculus agent, we have to represent the derivatives of the agent 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 agent, 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 agent $p = (\nu z)\, \bar{x}z.B(x,y,z)$; 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 $z = \min\big(\mathcal{N} \setminus \text{fn}(p)\big)$. 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 (see definition of ground bisimulation in Section 3.3), 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 4.4 (representative transitions) *A π-calculus transition $p \xrightarrow{\mu} q$ is a* representative transition *if*

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

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

The following lemma shows that the representative transitions express, up to α-conversion, all the behaviors of an agent. The proof is omitted, since it is standard for the π-calculus.

Lemma 4.5 *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 $q' = q\{y/x\; x/y\}$.*

μ	τ	$x(y)$		$\bar{x}y$		$\bar{x}x$	$\bar{x}(y)$	
l	`tau`	`bin`		`out`		$\mathtt{out}_2$	`bout`	
$\Box = \lambda(\Diamond) \in \mathrm{n}(\mu)$	/	x	y	x	y	x	x	y
$\Diamond = \kappa(\Box) \in \mathsf{L}^{\pi_\mathrm{g}}[l]$	/	n_sub	n_obj	n_sub	n_obj	n	n_sub	n_obj

Table 5: Relations between π-calculus labels and labels of HD-automata

If only representative transitions are used when building a HD-automaton from a π-calculus agent, 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 agents 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 an agent p, returns a pair $(q, \sigma) = \mathrm{norm}(p)$, where q is the representative of the class of agents 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 4.6 (from π-calculus agents to HD-automata) *The HD-automaton $\mathcal{A}_p^{\pi^g}$ corresponding to the ground semantics of π-calculus agent 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;*
- *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 5.*

Table 5 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 agent 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 $\mathsf{L}^{\pi_\mathrm{g}}[\mathtt{bin}] = \{n_\mathrm{sub}, n_\mathrm{obj}\}$; in this case, according to Table 5, 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 4.6; function λ will become useful in the following.

For each π-calculus agent p, the HD-automaton $\mathcal{A}_p^{\pi^g}$ is obviously finitely named. Now we identify a class of agents that generate finite HD-automata. This is the class of *finitary* π-calculus agents, which is defined like the corresponding class of CCS agents.

Definition 4.7 (finitary agents) *The* degree of parallelism $\deg(p)$ *of a π-calculus agent p is defined by the clauses of Definition 2.6 plus the following clause for matching:*

$$\deg([x{=}y]p) = \deg(p)$$

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

Theorem 4.8 *Let p be a finitary π-calculus agent. Then the HD-automaton $\mathcal{A}_p^{\pi^g}$ is finite.*

Proof. Let $n_0 = \max\{\deg(q) \mid p \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_i} q\}$ and let q be any agent reached from p in the construction of the HD-automaton $\mathcal{A}_p^{\pi^g}$. It must be

$$q \equiv (\nu x_1)\,(\nu x_2)\,\cdots(\nu x_m)\,(s_1|s_2|\cdots|s_n)$$

where q_i are sequential processes, $n \leq n_0$, $x_i \neq x_j$ if $i \neq j$, and $x_i \in \mathrm{fn}(s_1|\cdots|s_n)$.
First of all, we notice that — due to the operational semantics of the π-calculus — each component s_i must appear, up to substitutions on the names, either in p or in one of the definitions used by p. More formally, for each $i \in 1, \ldots, n$ there exists some agent p_i and some substitution σ_i such that:

- p_i appears in p or in p_A for some A; and
- $s_i = p_i\sigma_i$.

Since Var is finite, there is just a finite number of different possible candidates for p_i; so also the candidates for s_i are finite up to bijective substitutions (since the names in each p_i are finite, the substitutions σ_i can generate a number of different s_i which is finite up to bijective substitutions).
Since n is bounded by n_0, also the possible candidates for $s_1|s_2|\cdots|s_n$ are finite up to bijective substitutions. Finally, also the set of restricted names $\{x_1, x_2, \ldots, x_m\}$ is finite due to the requirements that $x_i \neq x_j$ if $i \neq j$ and that $x_i \in \mathrm{fn}(s_1|\cdots|s_n)$.
Therefore, the number of agents that can be reached in the construction of the HD-automaton $\mathcal{A}_p^{\pi g}$ is finite up to bijective substitutions. Since all the agents that are used as states in the HD-automaton are normalized, only a representative for each class of agents up to bijective substitutions appears in the HD-automaton, and the set of states has thus to be finite. To show that also the set of transitions is finite it is sufficient to notice that, since the recursion is guarded, any π-calculus agent can perform only a finite number of representative transitions. □

We remark that, as we discussed for CCS in Section 2.2, it is only semidecidable whether an agent is finitary. Also in this case, however, there is a syntactic conditions that guarantees that a π-calculus agent is finitary: the *finite-control* condition. According to Definition 2.8, an agent p has a finite control if no parallel composition appears in the recursive definitions used by p.

Corollary 4.9 *Let p be a finite-control π-calculus agent. Then the HD-automaton $\mathcal{A}_p^{\pi g}$ is finite.*

4.2 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 agents 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 4.10 (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:*

> *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 $\xi : \mathsf{T}_1[t_1]_{\mathrm{new}} \longleftrightarrow \mathsf{T}_2[t_2]_{\mathrm{new}}$, and some $\zeta : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_2[t_2]$ such that:*

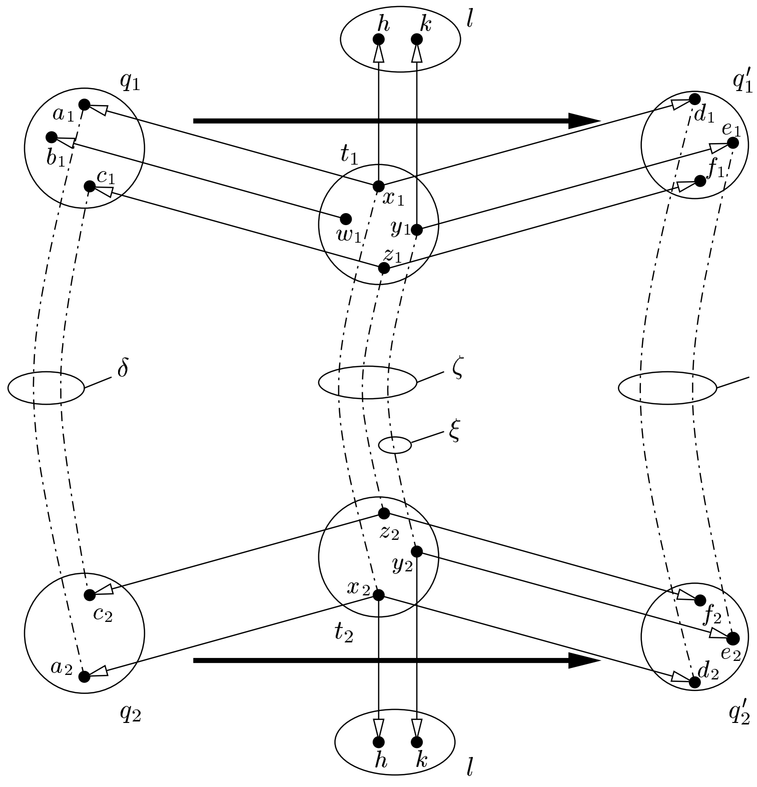

Figure 2: A step of bisimulation on HD-automata

- $\zeta = \left(\mathsf{s}_1[t_2]; \delta; \mathsf{s}_2[t_2]^{-1}\right) \cup \xi$,
- $\mathsf{o}_1[t_1] = \zeta; \mathsf{o}_2[t_2]$,
- $\langle q_1', \delta', q_2' \rangle \in \mathcal{R}$ *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 $\delta \subseteq \sigma_{01}; \sigma_{02}^{-1}$.*

4.2.1 Some basic properties of HD-bisimulation

Now we present some basic properties of HD-bisimulations.

Proposition 4.11 *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 4.12 (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.:*

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

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 4.13 *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:*

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

Proof. Suppose $\langle q_1, \delta, q_3\rangle \in \mathcal{R} \frown \mathcal{S}$. By definition of $\mathcal{R} \frown \mathcal{S}$, there are $\langle q_1, \delta', q_2\rangle \in \mathcal{R}$ and $\langle q_2, \delta'', q_3\rangle \in \mathcal{S}$ such that $\delta = \delta'; \delta''$. Suppose also that $t_1 : q_1 \xrightarrow{l} q_1'$. Since $\mathcal{R}$ is a HD-simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, there exist some $t_2 : q_2 \xrightarrow{l} q_2'$, $\xi' : \mathsf{T}_1[t_1]_{\mathrm{new}} \longleftrightarrow \mathsf{T}_2[t_2]_{\mathrm{new}}$ and $\zeta' : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_2[t_2]$ such that:

$$\zeta' = (\mathsf{s}_1[t_1]; \delta'; \mathsf{s}_2[t_2]^{-1}) \cup \xi \tag{1}$$
$$\mathsf{o}_1[t_1] = \zeta'; \mathsf{o}_2[t_2] \tag{2}$$

and $\langle q_1', \gamma', q_2'\rangle \in \mathcal{R}$ for

$$\gamma' \subseteq \mathsf{d}_1[t_1]^{-1}; \zeta'; \mathsf{d}_2[t_2]. \tag{3}$$

Since $t_2 : q_2 \xrightarrow{l} q_2'$ and $\mathcal{S}$ is a HD-simulation for $\mathcal{A}_2$ and $\mathcal{A}_3$, there are some $t_3 : q_3 \xrightarrow{l} q_3'$, $\xi'' : \mathsf{T}_1[t_1]_{\mathrm{new}} \longleftrightarrow \mathsf{T}_2[t_2]_{\mathrm{new}}$ and $\zeta'' : \mathsf{T}_2[t_2] \longleftrightarrow \mathsf{T}_3[t_3]$ such that:

$$\zeta'' = (\mathsf{s}_2[t_2]; \delta''; \mathsf{s}_3[t_3]^{-1}) \cup \xi'' \tag{4}$$
$$\mathsf{o}_2[t_2] = \zeta''; \mathsf{o}_3[t_3] \tag{5}$$

and $\langle q_2', \gamma'', q_3'\rangle \in \mathcal{S}$ for

$$\gamma'' \subseteq \mathsf{d}_2[t_2]^{-1}; \zeta''; \mathsf{d}_3[t_3]. \tag{6}$$

Let us define $\xi : \mathsf{T}_1[t_1]_{\mathrm{new}} \longleftrightarrow \mathsf{T}_3[t_3]_{\mathrm{new}}$ and $\zeta : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_3[t_3]$ as follows:

$$\xi \overset{\mathrm{def}}{=} \xi'; \xi'' \qquad \zeta \overset{\mathrm{def}}{=} \zeta'; \zeta''.$$

Now we are ready to show that transition t_3 satisfies all the condition of Definition 4.10 w.r.t. transition t_1. First of all we prove $\zeta = (\mathsf{s}_1[t_1]; \delta; \mathsf{s}_3[t_3]^{-1}) \cup \xi$:

$$\begin{aligned}
\zeta &= \zeta'; \zeta'' && \text{by definition of } \zeta\\
&= (\mathsf{s}_1[t_1]; \delta'; \mathsf{s}_2[t_2]^{-1} \cup \xi')\,;\,(\mathsf{s}_2[t_2]; \zeta''; \mathsf{s}_3[t_3]^{-1} \cup \xi'') && \text{by (1) and (4)}\\
&= (\mathsf{s}_1[t_1]; \delta'; \mathsf{s}_2[t_2]^{-1}; \mathsf{s}_2[t_2]; \zeta''; \mathsf{s}_3[t_3]^{-1}) \cup (\xi'; \xi'') && \text{since } \mathsf{s}_1[t_1]; \delta'; \mathsf{s}_2[t_2]^{-1}; \xi'' = \emptyset\\
& && \text{and } \xi'; \mathsf{s}_2[t_2]; \zeta''; \mathsf{s}_3[t_3]^{-1} = \emptyset\\
&= (\mathsf{s}_1[t_1]; \delta'; \delta''; \mathsf{s}_3[t_3]^{-1}) \cup (\xi'; \xi'') && \text{since } \mathsf{s}_2[t_2]^{-1} \text{ is a total injection}\\
&= (\mathsf{s}_1[t_1]; \delta; \mathsf{s}_3[t_3]^{-1}) \cup \xi && \text{since } \delta'; \delta'' = \delta \text{ and } \xi'; \xi'' = \xi.
\end{aligned}$$

Then we prove $\mathsf{o}_1[t_1] = \zeta; \mathsf{o}_3[t_3]$:

$$\begin{aligned}
\mathsf{o}_1[t_1] &= \zeta'; \mathsf{o}_2[t_2] && \text{by (2)}\\
&= \zeta'; \zeta''; \mathsf{o}_3[t_3] && \text{by (5)}\\
&= \zeta; \mathsf{o}_3[t_3] && \text{by definition of } \zeta.
\end{aligned}$$

Finally $\langle q_1', (\gamma'; \gamma''), q_3'\rangle$ holds by definition of $\mathcal{R} \frown \mathcal{S}$; finally $\gamma'; \gamma'' \subseteq \mathsf{d}_1[t_1]^{-1}; \zeta; \mathsf{d}_3[t_3]$. In fact:

$$\begin{aligned}
\gamma'; \gamma'' &\subseteq \mathsf{d}_1[t_1]^{-1}; \zeta'; \mathsf{d}_2[t_2]; \mathsf{d}_2[t_2]^{-1}; \zeta''; \mathsf{d}_3[t_3] && \text{by (3) and (6)}\\
&\subseteq \mathsf{d}_1[t_1]^{-1}; \zeta'; \zeta''; \mathsf{d}_3[t_3] && \text{since } \mathsf{d}_2[t_2]; \mathsf{d}_2[t_2]^{-1} \subseteq \mathrm{id}_{\mathsf{T}_2[t_2]}\\
&= \mathsf{d}_1[t_1]; \zeta; \mathsf{d}_3[t_3]^{-1} && \text{by definition of } \zeta.
\end{aligned}$$

This concludes the proof that $\mathcal{R} \frown \mathcal{S}$ is a HD-simulation. Since $(\mathcal{R} \frown \mathcal{S})^{-1} = \mathcal{S}^{-1} \frown \mathcal{R}^{-1}$ we also have that if $\mathcal{R}$ and $\mathcal{S}$ are HD-bisimulations then $\mathcal{R} \frown \mathcal{S}$ is a HD-bisimulation. $\square$

It is now simple to prove that relation $\sim$ is an equivalence on HD-automata: symmetry and reflexivity are immediate, whereas transitivity derives from the previous proposition.

Corollary 4.14 *If $\mathcal{A}_1 \sim \mathcal{A}_2$ and $\mathcal{A}_2 \sim \mathcal{A}_3$ then also $\mathcal{A}_1 \sim \mathcal{A}_3$.*

Proposition 4.15 *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:*

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

Proof. Suppose $\langle q_1, \delta, q_2\rangle \in \widehat{\mathcal{R}}$; then $\langle q_1, \bar{\delta}, q_2\rangle \in \mathcal{R}$ for some $\bar{\delta} \subseteq \delta$.
Suppose moreover that $t_1 : q_1 \xrightarrow{l} q_1'$. Since $\mathcal{R}$ is a HD-bisimulation, there exist $t_2 : q_2 \xrightarrow{l} q_2'$, $\xi : \mathsf{T}_1[t_1]_{\rm new} \longrightarrow \mathsf{T}_2[t_2]_{\rm new}$ and $\bar{\zeta} : \mathsf{T}_1[t_1] \longrightarrow \mathsf{T}_2[t_2]$ such that

$$\bar{\zeta} = (\mathsf{s}_1[t_1]; \bar{\delta}; \mathsf{s}_2[t_2]^{-1}) \cup \xi \qquad (7)$$
$$\mathsf{o}_1[t_1] = \bar{\zeta}; \mathsf{o}_2[t_2] \qquad (8)$$

and $\langle q_1', \delta', q_2'\rangle \in \mathcal{R}$ where

$$\delta' \subseteq \mathsf{d}_1[t_1]^{-1}; \bar{\zeta}; \mathsf{d}_2[t_2]. \qquad (9)$$

Now, define $\zeta : \mathsf{T}_1[t_1] \longrightarrow \mathsf{T}_2[t_2]$ as follows.

$$\zeta \overset{\rm def}{=} (\mathsf{s}_1[t_1]; \delta; \mathsf{s}_2[t_2]^{-1}) \cup \xi. \qquad (10)$$

Relation ζ is a partial correspondence $\zeta : \mathsf{T}_1[t_1] \longrightarrow \mathsf{T}_2[t_2]$, since $\mathrm{dom}(\mathsf{s}_1[t_1]\delta; \mathsf{s}_2[t_2]^{-1}) \cap \mathrm{dom}(\xi) = \emptyset$.
Since $\bar{\delta} \subseteq \delta$, by (7) we also have $\bar{\zeta} \subseteq \zeta$. So, $\mathsf{o}_1[t_1] = \zeta; \mathsf{o}_2[t_2]$; in fact,

$$\begin{aligned} \mathsf{o}_1[t_1] &= \bar{\zeta}; \mathsf{o}_1[t_1] && \text{by (8)} \\ &\subseteq \zeta; \mathsf{o}_1[t_1] && \text{since } \bar{\zeta} \subseteq \zeta \end{aligned}$$

which implies $\mathsf{o}_1[t_1] = \zeta; \mathsf{o}_2[t_2]$ since $\mathrm{cod}(\mathsf{o}_1[t_1]) = \mathrm{cod}(\mathsf{o}_2[t_1]) = \mathsf{L}[l]$ and $\mathsf{o}_1[t_1]$, $\mathsf{o}_2[t_2]$ are injective.
Finally, $\langle q_1', \delta', q_2'\rangle \in \widehat{\mathcal{R}}$ holds since $\langle q_1', \delta', q_2'\rangle \in \mathcal{R}$ and $\mathcal{R} \subseteq \widehat{\mathcal{R}}$; finally, $\delta' \subseteq \mathsf{d}_1[t_1]^{-1}; \zeta; \mathsf{d}_2[t_2]$ by (9) and $\bar{\zeta} \subseteq \zeta$.
This concludes the proof that $\widehat{\mathcal{R}}$ is a HD-simulation. Since $\left(\widehat{\mathcal{R}}\right)^{-1} = \widehat{\mathcal{R}^{-1}}$, we also have that if $\mathcal{R}$ is a HD-bisimulation then also $\widehat{\mathcal{R}}$ is a HD-bisimulation. ☐

Corollary 4.16 *$\mathcal{R}_{\mathcal{A}}$ is closed for $\widehat{\cdot}$ and $\cdot \frown \cdot$.*

Proposition 4.15 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 4.13: in fact, if we consider the HD-automaton of Figure 3, then relations

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

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.

4.2.2 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 agents, coincides with bisimilarity relation $\sim_g$.

We have seen that a state of a HD-automaton is obtained from a π-calculus agent by normalizing its names, so that all the agents that differ for a renaming are represented by the same state. Conversely, a particular π-calculus agent 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 $\sigma : \mathsf{Q}[q] \hookrightarrow \mathcal{N}$, 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.

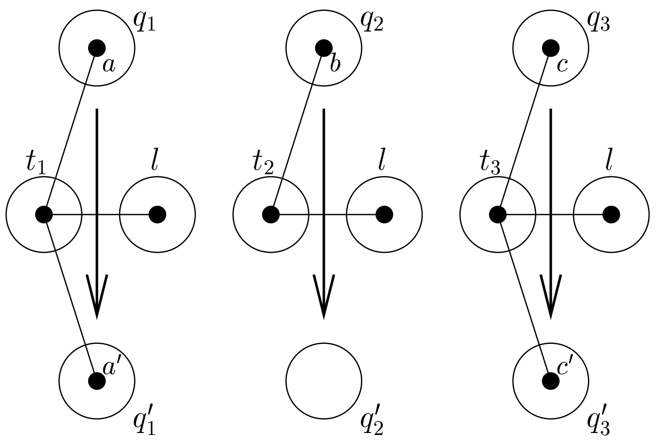

Figure 3: A tricky example for concatenation of HD-bisimulations

Definition 4.17 (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 $\mathsf{G}_{\mathcal{A}}[(q,\sigma)] \overset{\text{def}}{=} \sigma(\mathsf{Q}[q])$.*
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 $\mathsf{U}_{\mathcal{A}}[(t,\rho)] \overset{\text{def}}{=} \rho(\mathsf{U}[t])$. Moreover we use the notations $\mathsf{U}_{\mathcal{A}}[(t,\rho)]_{\text{old}} \overset{\text{def}}{=} \rho(\mathsf{T}[t]_{\text{old}})$ and $\mathsf{U}_{\mathcal{A}}[(t,\rho)]_{\text{new}} \overset{\text{def}}{=} \rho(\mathsf{T}[t]_{\text{new}})$.*
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 4.18 (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 $\mathsf{U}_1[u_1]_{\text{new}} \cap \mathsf{G}_2[g_2] = \emptyset$ 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 "$\mathsf{U}_1[u_1]_{\text{new}} \cap \mathsf{G}_2[g_2] = \emptyset$" 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 "$\text{bn}(\mu) \cap \text{fn}(p|q) = \emptyset$" in the definitions of bisimulation in π-calculus (Definition 3.2).

Global bisimilarity coincides with HD-bisimilarity.

Proposition 4.19 *Two HD-automata are HD-bisimilar if and only if they are global bisimilar.*

Proof. We prove the two implications separately.
Proof of the "only if" implication. It is sufficient to prove that if $\mathcal{R}$ is a HD-simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\mathcal{R}'$ is a global simulation, where

$$\mathcal{R}' \overset{\text{def}}{=} \{\langle (q_1,\sigma_1),(q_2,\sigma_2)\rangle \mid \langle q_1,\delta,q_2\rangle \in \widehat{\mathcal{R}} \text{ with } \delta = \sigma_1;\sigma_2^{-1}\}.$$

Suppose $(q_1,\sigma_1) \;\mathcal{R}'\; (q_2,\sigma_2)$ and $(t_1,\rho_1) : (q_1,\sigma_1) \xrightarrow{(l,\lambda)} (q_1',\sigma_1')$ with

$$\rho_1(\mathsf{T}_1[t_1]_{\text{new}}) \cap \sigma_2(\mathsf{Q}_2[q_2]) = \emptyset. \tag{11}$$

We have to show that there is some $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(l,\lambda)} (q_2', \sigma_2')$ with $(q_1', \sigma_1') \ \mathcal{R}' \ (q_2', \sigma_2')$.
From $(q_1, \sigma_1) \ \mathcal{R}' \ (q_2, \sigma_2)$ we obtain, by definition of $\mathcal{R}'$, that $\langle q_1, \delta, q_2 \rangle \in \widehat{\mathcal{R}}$ where $\delta \overset{\text{def}}{=} \sigma_1; \sigma_2^{-1}$.
From $(t_1, \rho_1) : (q_1, \sigma_1) \xrightarrow{(l,\lambda)} (q_1', \sigma_1')$ by definition of global transition we obtain $t_1 : q_1 \xrightarrow{l} q_1'$ and:

$$\sigma_1 = \mathsf{s}_1[t_1]^{-1}; \rho_1 \qquad \lambda = \mathsf{o}_1[t_1]^{-1}; \rho_1 \qquad \sigma_1' = \mathsf{d}_1[t_1]^{-1}; \rho_1.$$

Since $\mathcal{R}$ is a HD-simulation, there exist some $t_2 : q_2 \xrightarrow{l} q_2'$, $\xi : \mathsf{T}_1[t_1]_{\text{new}} \longleftrightarrow \mathsf{T}_2[t_2]_{\text{new}}$ and $\zeta : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_2[t_2]$ such that

$$\zeta = (\mathsf{s}_1[t_1]; \delta; \mathsf{s}_2[t_2]^{-1}) \cup \xi \tag{12}$$
$$\mathsf{o}_1[t_1] = \zeta; \mathsf{o}_2[t_2] \tag{13}$$
$$\langle q_1', \delta', q_2' \rangle \in \widehat{\mathcal{R}} \text{ with } \delta' = \mathsf{d}_1[t_1]^{-1}; \zeta; \mathsf{d}_2[t_2]. \tag{14}$$

Let

$$\rho_2 \overset{\text{def}}{=} (\mathsf{s}_2[t_2]; \sigma_2) \cup (\xi^{-1}; \rho_1).$$

Since $\mathrm{dom}(\xi^{-1}) \subseteq \mathsf{T}_2[t_2]_{\text{new}}$ and, by definition, $\mathsf{T}_2[t_2]_{\text{new}} = \mathsf{T}_2[t_2] \setminus \mathrm{dom}(\mathsf{s}_2[t_2])$, we conclude that $\rho_2 : \mathsf{T}_2[t_2] \longleftrightarrow \mathcal{N}$.
Property

$$\zeta = \rho_1; \rho_2^{-1} \tag{15}$$

can be shown by considering separately the names $n \in \mathrm{dom}(\mathsf{s}_1[t_1])$ and the names $n \in \mathsf{T}_1[t_1]_{\text{new}}$, and exploiting (11).
Moreover, by definition of ρ_2, $\sigma_2 = \mathsf{s}_2[t_2]^{-1}; \rho_2$ and $\lambda = \mathsf{o}_2[t_2]^{-1}; \rho_2$. Therefore, by definition of global transition, there is some $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(l,\lambda)} (q_2', \sigma_2')$, where

$$\sigma_2' \overset{\text{def}}{=} \mathsf{d}_2[t_2]^{-1}; \rho_2. \tag{16}$$

Finally,

$$\begin{aligned}
\delta' &= \mathsf{d}_1[t_1]^{-1}; \zeta; \mathsf{d}_2[t_2] && \text{by (14)} \\
&= \mathsf{d}_1[t_1]^{-1}; \rho_1; \rho_2^{-1}; \mathsf{d}_2[t_2] && \text{by (15)} \\
&= \sigma_1'; \sigma_2'^{-1} && \text{by (13) and (16).}
\end{aligned}$$

Since $\langle q_1', \delta', q_2' \rangle \in \mathcal{R}$, by definition of $\mathcal{R}'$, it holds $(q_1', \sigma_1') \ \mathcal{R}' \ (q_2', \sigma_2')$. This concludes the proof of the "only if" implication.
Proof of the "if" implication. It is sufficient to prove that if $\mathcal{R}$ is a global simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\mathcal{R}'$ is a HD-bisimulation, where

$$\mathcal{R}' \overset{\text{def}}{=} \{\langle q_1, \delta, q_2 \rangle \mid (q_1, \sigma_1) \ \mathcal{R} \ (q_2, \sigma_2) \text{ and } \delta = \sigma_1; \sigma_2^{-1}\}.$$

Suppose $\langle q_1, \delta, q_2 \rangle \in \mathcal{R}'$, $t_1 : q_1 \xrightarrow{l} q_1'$. We have to show that there exist some $t_2 : q_2 \xrightarrow{l} q_2'$, 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:

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

Since $\langle q_1, \delta, q_2 \rangle \in \mathcal{R}'$ we have $(q_1, \sigma_1) \ \mathcal{R} \ (q_2, \sigma_2)$ and

$$\delta = \sigma_1; \sigma_2^{-1}. \tag{17}$$

Let $\rho_1 : \mathsf{T}_1[t_1] \longleftrightarrow \mathcal{N}$ be such that

$$\sigma_1 = \mathsf{s}_1[t_1]^{-1}; \rho_1 \tag{18}$$
$$\rho_1(\mathsf{T}_1[t_1]_{\text{new}}) \cap \sigma_2(\mathsf{Q}_2[q_2]) = \emptyset \tag{19}$$

To obtain such a function, it is sufficient to define $\rho_1\big|_{\mathsf{T}_1[t_1]_{\text{old}}} = \mathsf{s}_1[t_1]; \sigma_1$ and to let $\rho_1\big|_{\mathsf{T}_1[t_1]_{\text{new}}} : \mathsf{T}_1[t_1]_{\text{new}} \longleftrightarrow \mathcal{N} \setminus \big(\mathrm{cod}(\sigma_1) \cup \mathrm{cod}(\sigma_2)\big)$.
By exploiting (18), $(t_1, \rho_1) : (q_1, \sigma_1) \xrightarrow{(l,\lambda)} (q_1', \sigma_1')$ with

$$\lambda = \mathsf{o}_1[t_1]^{-1}; \rho_1 \tag{20}$$
$$\sigma_1' = \mathsf{d}_1[t_1]^{-1}; \rho_1. \tag{21}$$

By definition of global bisimulation, exploiting (19), there exists some $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(l,\lambda)} (q'_2, \sigma'_2)$ such that $(q'_1, \sigma'_1) \mathcal{R} (q'_2, \sigma'_2)$,

$$\sigma_2 = \mathsf{s}_2[t_2]^{-1}; \rho_2 \tag{22}$$
$$\mathsf{o}_1[t_1]^{-1}; \rho_1 = \lambda = \mathsf{o}_2[t_2]^{-1}; \rho_2 \tag{23}$$
$$\sigma'_2 = \mathsf{d}_2[t_2]^{-1}; \rho_2. \tag{24}$$

Hence $t_2 : q_2 \xrightarrow{l} q'_2$. Let

$$\zeta \overset{\text{def}}{=} \rho_1; \rho_2^{-1} : \mathsf{T}_1[t_1] \longrightarrow \mathsf{T}_2[t_2].$$

Notice that $n \in \mathsf{T}_1[t_1]_{\text{new}}$ if and only if $\zeta(n) \in \mathsf{T}_2[t_2]_{\text{new}}$: this is due to (19). Hence, ζ consists of a partial correspondence between the old names of the two transitions and of a partial correspondence between their new names. Moreover, by defining

$$\xi \overset{\text{def}}{=} \zeta\big|_{\mathsf{T}_1[t_1]_{\text{new}}},$$

and by exploiting (17), (18) and (22), we have:

$$\zeta = (\mathsf{s}_1[t_1]; \delta; \mathsf{s}_2[t_2]^{-1}) \cup \xi$$

Also, by exploiting (20) and (23)

$$\mathsf{o}_1[t_1] = \zeta; \mathsf{o}_2[t_2].$$

Finally, $(q'_1, \sigma'_1) \mathcal{R} (q'_2, \sigma'_2)$ implies $\langle q'_1, \delta', q'_2 \rangle \in \mathcal{R}'$ for $\delta' \overset{\text{def}}{=} \sigma'_1; {\sigma'_2}^{-1}$, and

$$\delta' = \mathsf{d}_1[t_1]^{-1}; \zeta; \mathsf{d}_2[t_2]$$

follows by (21) and (24). This concludes the proof. $\square$

4.2.3 Relating π-calculus ground bisimulation and HD-bisimulation

Now we show that two π-calculus agents 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 4.20 *Let (q, σ) be a global state of the HD-automaton $\mathcal{A}_p^{\pi_g}$ corresponding to a π-calculus agent p. Then:*

- *if $q\sigma \xrightarrow{\mu} q''$ is a π-calculus transition with $\text{bn}(\mu) \cap \text{fn}(q\sigma) = \emptyset$, 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 5.

Theorem 4.21 *Let p_1 and p_2 be π-calculus agents. Then $p_1 \sim_g p_2$ iff $\mathcal{A}_{p_1}^{\pi_g} \sim \mathcal{A}_{p_2}^{\pi_g}$.*

Proof (Sketch). Suppose $p_1 \sim_g p_2$; let the relation $\mathcal{R}$ on the global states of the two HD-automata $\mathcal{A}_{p_1}^{\pi_g}$ and $\mathcal{A}_{p_2}^{\pi_g}$ be defined as follows:

$$\mathcal{R} = \{\langle (q_1, \sigma_1), (q_2, \sigma_2) \rangle \in \mathsf{G}_{p_1} \times \mathsf{G}_{p_2} \mid q_1\sigma_1 \sim_g q_2\sigma_2\}.$$

By exploiting Lemma 4.20, relation $\mathcal{R}$ is a global bisimulation for $\mathcal{A}_{p_1}^{\pi_g}$ and $\mathcal{A}_{p_2}^{\pi_g}$. Moreover $(q_{01}, \sigma_{01}) \mathcal{R} (q_{02}, \sigma_{02})$, since $q_{01}\sigma_{01} = p_1 \sim_g p_2 = q_{02}\sigma_{02}$, so the two HD-automata are bisimilar.
Conversely, suppose $\mathcal{A}_{p_1}^{\pi_g} \sim \mathcal{A}_{p_2}^{\pi_g}$ and let $\mathcal{S}$ be the largest global bisimulation for the two automata, i.e., the one corresponding to HD-bisimulation $\mathcal{R}_{\mathcal{A}_{p_1}^{\pi_g};\mathcal{A}_{p_2}^{\pi_g}}$. By exploiting Lemma 4.20,

$$\mathcal{R} = \{\langle q_1\sigma_1, q_2\sigma_2 \rangle \mid (q_1, \sigma_1) \mathcal{S} (q_2, \sigma_2)\}$$

is a ground bisimulation. In particular, since $(q_{01}, \sigma_{01}) \mathcal{S} (q_{02}, \sigma_2)$ and $p_1 = q_{01}\sigma_{01}$ and $p_2 = q_{02}\sigma_{02}$, we obtain $p_1 \sim_g p_2$. $\square$

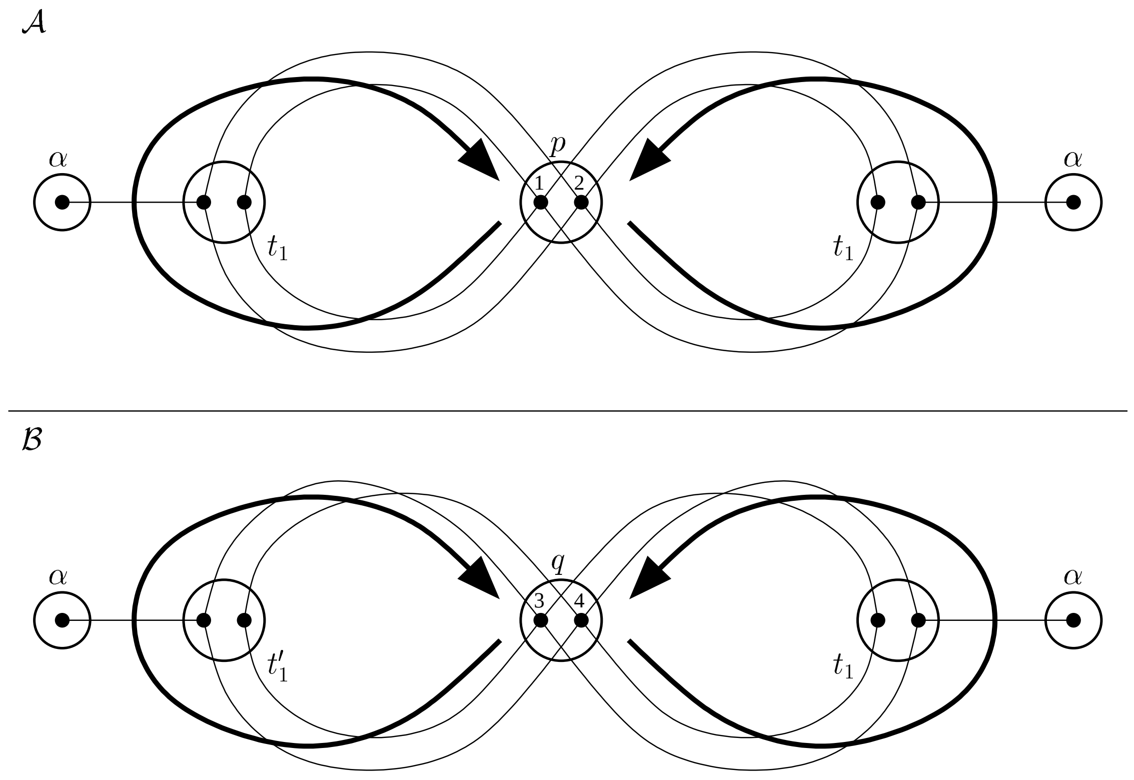

Figure 4: Two non isomorphic minimal HD-automata

4.3 Minimization of HD-automata

We conclude this section with a comment on the impossibility of defining minimal 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 not possible on Basic HD-automata. 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 Section 6 we define an enhanced version of HD-automata that solves this problem by allowing symmetries on names to appear explicitly in the states of the HD-automata. We will show that canonical minimal realizations exist for this enhanced class of HD-automata (see Section 6.5).

5 Basic HD-automata for other history-dependent calculi

In this section we present two other examples of history-dependent formalisms that can be successfully mapped into Basic HD-automata. The first formalism (Sections 5.1 and 5.2) is CCS with locality [BCHK93], an extension of CCS that takes into account the distributed nature of concurrent systems: the underlying idea is that each action occurs at a particular location of the system, and this location is observed in the labels of the transitions. The second (Sections 5.3 and 5.4) is an example of history-dependent formalism outside the filed of process calculi: Petri nets equipped with history-preserving bisimulation [GR83].

5.1 CCS with localities

Here we recall in brief the approach to location semantics for CCS that has been introduced in [BCHK93] and [Kie94]. In Section 5.2 we show how to map this semantics of CCS on HD-automata.

Location semantics is one of the so-called *truly concurrent* semantics of CCS, which discriminate systems not only according to the *sequences* of actions the systems perform, but also considering aspects like the degree of parallelism in the computations and the distribution of the actions in the space. In the location semantics, in particular, localities are assigned to the parallel components of a system, and the location in which an action occurs is observed in the corresponding transition. Hence, agents $\alpha.\mathbf{0}|\beta.\mathbf{0}$ and $\alpha.\beta.\mathbf{0}+\beta.\alpha.\mathbf{0}$, that are identified according to the interleaving semantics, are distinguished in the location semantics: the first agent may perform actions α and β in different places, while the second executes both actions in the same location. This behavior is obtained by extending the syntax of CCS with *location prefixes* $l :: p$ and by observing in the operational semantics the localities in which the actions take place. For instance, in $l :: p|m :: q$ the actions of p are observed to happen in location l, whereas the actions of q are observed in location m. Differently from the *static* approach to localities of [Ace94] — where the distributed nature of agents is made explicit by assigning different locations to their parallel components, as in $\alpha.(l :: p|m :: q)$ — in [BCHK93] and [Kie94] a more observational point of view is preferred. Location names are assigned *dynamically*, during the process of observation: the meaning of transition $l :: \alpha.p \underset{lm}{\xrightarrow{\alpha}} l :: m :: p$ is that the observer sees an action α emanating from a particular sub-location of l and associates name m to this sub-location. Here we follow the dynamic approach. We refer to [Cas93] for further comparisons of static and dynamic approach.

Agents $\alpha.\mathbf{0}|\beta.\mathbf{0}$ and $\alpha.\beta.\mathbf{0}+\beta.\alpha.\mathbf{0}$ are distinguished by this approach: in fact,

$$\alpha.\mathbf{0}|\beta.\mathbf{0} \underset{l}{\xrightarrow{\alpha}} l :: \mathbf{0}|\beta.\mathbf{0} \underset{m}{\xrightarrow{\beta}} l :: \mathbf{0}|m :: \mathbf{0}$$

whereas

$$\alpha.\beta.\mathbf{0} + \beta.\alpha.\mathbf{0} \underset{l}{\xrightarrow{\alpha}} l :: \beta.\mathbf{0} \underset{lm}{\xrightarrow{\beta}} l :: m :: \mathbf{0}$$

and m is a sub-location of l only in the second agent.

Given a set *Loc* of locations (ranged over by $l, m, \ldots$; sequence of locations $l_1 l_2 \cdots l_n$ are ranged over by $u, v, \ldots$), the *CCS location agents* are defined by extending CCS syntax as follows:

$$p \ ::= \ \cdots \ \Big| \ l :: p.$$

The set of location names that occur in p is denoted by $\mathrm{loc}(p)$ and an agent p is *pure* if $\mathrm{loc}(p) = \emptyset$.

The following equivalences are added to the ones in Section 2.2 to define structural equivalence $\equiv$ on location agents:

$$\textbf{(loc)} \quad l :: \mathbf{0} \equiv \mathbf{0} \qquad l :: (p|q) \equiv (l :: p)|(l :: q) \qquad l :: (\nu\alpha)\, p \equiv (\nu\alpha)\, l :: p.$$

Two kinds of transitions are defined on location agents. There are the *standard CCS transitions*, which are generated by the axioms schemata and inference rules of Table 1, extended with the following rule for location prefixes:

$$[\textsc{loc}]\frac{p \xrightarrow{\mu} p'}{l :: p \xrightarrow{\mu} l :: p'}$$

And there are the *location transitions*, of the form $p \underset{u}{\xrightarrow{a}} p'$, which are generated by the axiom schemata and by the inference rules of Table 6. Notice that there is no synchronization rule for the location transitions: since the invisible transitions do not occur in a particular location, the standard transitions are used for them.

The definition of location bisimulation follows.[3]

Definition 5.1 (location bisimulation) *A relation $\mathcal{R}$ over location agents is a* location simulation *if whenever $p \ \mathcal{R} \ q$ then:*

- *for each $p \underset{ul}{\xrightarrow{a}} p'$, with $l \notin \mathrm{loc}(p|q)$, there is some $q \underset{ul}{\xrightarrow{a}} q'$ such that $p' \ \mathcal{R} \ q'$; and*
- *for each $p \xrightarrow{\tau} p'$ there is some $q \xrightarrow{\tau} q'$ such that $p' \ \mathcal{R} \ q'$;*

[3]In [BCHK93] CCS is equipped with a *weak* location bisimulation, while here we consider the *strong* case. The approach defined here also applies to the *weak* case — see Section 8.4.

$$
\boxed{
\begin{array}{ll}
[\text{PREF}]\ a.p \xrightarrow[l]{a} l::p & [\text{LOC}]\ \dfrac{p \xrightarrow[u]{a} p'}{l::p \xrightarrow[lu]{a} l::p'} \\[2ex]
[\text{SUM}]\ \dfrac{p_1 \xrightarrow[u]{a} p'}{p_1+p_2 \xrightarrow[u]{a} p'} & [\text{PAR}]\ \dfrac{p_1 \xrightarrow[u]{a} p_1'}{p_1|p_2 \xrightarrow[u]{a} p_1'|p_2} \\[2ex]
[\text{RES}]\ \dfrac{p \xrightarrow[u]{a} p'}{(\nu\alpha)\,p \xrightarrow[u]{a} (\nu\alpha)\,p'}\ \text{if } a \neq \alpha, \bar{\alpha} & [\text{IDE}]\ \dfrac{p_A \xrightarrow[u]{a} p'}{A \xrightarrow[u]{a} p'}\ \text{if } A \stackrel{\text{def}}{=} p_A
\end{array}
}
$$

Table 6: Location transitions for CCS with localities

A relation $\mathcal{R}$ is a location bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are location simulations.*
Two location agents p and q are location bisimilar*, written $p \sim_l q$, if $p \ \mathcal{R}\ q$ for some location bisimulation $\mathcal{R}$.*

The problems in using labelled transition systems to give semantics to CCS with localities are the classical problems of history-dependent calculi. First of all, an infinite number of location names can be used when a new location is generated. In this case, moreover, even if just one location name is chosen at each point, still an infinite number of states is necessary for agents with infinite behaviors: consider for instance $A \stackrel{\text{def}}{=} \alpha.A$; staring from $l_0 :: A$, an infinite number of different states is reachable:

$$l_0 :: A \xrightarrow[l_0 l_1]{\alpha} l_0 :: l_1 :: A \xrightarrow[l_0 l_1 l_2]{\alpha} l_0 :: l_1 :: l_2 :: A \cdots$$

This happens since location prefixes are created but never forgotten.

Also the definition of bisimulation is not the ordinary one, due to the clause "$l \notin \mathrm{loc}(p|q)$" in Definition 5.1.

We conclude the section by showing that the transitions that a location agent can perform do not depend on the particular instantiation of the location names.

Let p be a CCS agent with localities and let σ be any renaming of the locations, i.e., $\sigma : \mathit{Loc} \longleftrightarrow \mathit{Loc}$; then we represent with $p\sigma$ the agent p whose localities have been updated according to σ.

Lemma 5.2 *Let p be a location agent and let $\sigma : \mathit{Loc} \longleftrightarrow \mathit{Loc}$ be a renaming of the locations. Then*

- *if $p \xrightarrow[u]{\alpha} p'$ then $p\sigma \xrightarrow[v]{\alpha} p'\sigma$, and*
- *if $p\sigma \xrightarrow[v]{\alpha} q$ then $p \xrightarrow[u]{\alpha} p'$ and $q = p'\sigma$,*

where in both cases $v = \sigma(l_1)\sigma(l_2)\cdots\sigma(l_k)$ whenever $u = l_1 l_2 \cdots l_k$. Similarly,

- *if $p \xrightarrow{\tau} p'$ then $p\sigma \xrightarrow{\tau} p'\sigma$, and*
- *if $p\sigma \xrightarrow{\tau} q$ then $p \xrightarrow{\tau} p'$ and $q = p'\sigma$.*

The proof of this lemma can be found in [BCHK93].

With a little abuse of notation, we will also write $p\sigma$ with $\sigma : \mathrm{loc}(p) \longrightarrow \mathit{Loc}$, with the obvious meaning.

5.2 Representing agents with localities as basic HD-automata

Now we show that HD-automata can be applied to CCS with localities. The simplest way to represent a location agent with a HD-automaton would be to use states of the automata to represent derivatives of the agent and to associate to a state the location names of the corresponding agent. This encoding has some advantages w.r.t. the ordinary operational semantics: all the transitions corresponding to different choices in the name for a new locations could be identified. Moreover, the same state of the HD-automaton could be used to represent a whole class of location agents which differ for a renaming of the locations.

However, this is not sufficient to obtain finite HD-automata for the class of finitary location agents: whenever the agent can perform infinite computations, the number of location prefixes continues to grow during the computations, thus leading to an infinite number of derivatives, even up to renamings.

First of all, we have to avoid this unbounded growth of the location prefixes, i.e., we have to find a way to discard these prefixes. An axiom like

$$(\mathbf{del}) \quad l :: m :: p \equiv m :: p$$

would help. However, this axiom is not correct for the location equivalence of Definition 5.1, since, for instance,

$$l :: m :: a.p \xrightarrow[lmn]{a} l :: m :: n :: p$$

whereas

$$m :: a.p \xrightarrow[mn]{a} m :: n :: p$$

and the two labels do not correspond; this happens because the whole sequence of locations is observed in the label of a transition.

Now we present a slightly different definition of location equivalence in which only the newly created location and its direct parent are observed. It can be shown that this new location equivalence coincides with the classical one, at least for the class of pure CCS agents.

Definition 5.3 (incremental location equivalence) *A relation $\mathcal{R}$ on location agents is an* incremental location simulation *if $p \; \mathcal{R} \; q$ implies:*

- *for each $p \xrightarrow[umn]{a} p'$, with $n \notin \mathrm{loc}(p,q)$, there exists some $q \xrightarrow[vmn]{a} q'$ with $p' \; \mathcal{R} \; q'$;*
- *for each $p \xrightarrow{\tau} p'$ there exists some $q \xrightarrow{\tau} q'$ with $p' \; \mathcal{R} \; q'$.*

A relation $\mathcal{R}$ is a incremental location bisimulation *if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are incremental location simulations.*
Two location agents p and q are incremental location equivalent *(written $p \sim_{\Delta l} q$) if $p \; \mathcal{R} \; q$ for some location bisimulation $\mathcal{R}$.*

Proposition 5.4 *Let p_0 and q_0 be two pure CCS agents. Then $p_0 \sim_l q_0$ if and only if $m :: p_0 \sim_{\Delta l} m :: q_0$ for some location m.*

Proof (Sketch). The proof that $p_0 \sim_l q_0$ implies $m :: p_0 \sim_{\Delta l} m :: q_0$ is easy. In fact, standard results for location bisimilarity ensure that $p_0 \sim_l q_0$ implies $m :: p_0 \sim_l m :: q_0$. Moreover, each location bisimulation is by definition also an incremental location bisimulation so $m :: p_0 \sim_{\Delta l} m :: q_0$.
To prove the converse, i.e., that $m :: p_0 \sim_{\Delta l} m :: q_0$ implies $p_0 \sim_l q_0$, we should show that relation $\mathcal{R}$ defined as follows is a location bisimulation:

$$\mathcal{R} = \{(p_n, q_n) \mid p_n \sim_{\Delta l} q_k, \; l_0 :: p_0 \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_k} p_k, \; l_0 :: q_0 \xrightarrow{\mu_1} \cdots \xrightarrow{\mu_k} q_k\}$$

where $\xrightarrow{\mu_i}$ is either $\xrightarrow{\tau}$ or $\xrightarrow[u_i m_i]{a_i}$ with $m_i \notin \mathrm{loc}(p_{i-1}, q_{i-1})$. Hence, $m :: p_0 \sim_l m :: q_0$ and, by exploiting standard results for location equivalence, $p_0 \sim_l q_0$. □

Del is a correct axiom for this alternative characterization and it allows us, combined with the other equivalences of the structural congruence, to associate to each agent a flat structure of locations. Conceptually, these axioms show that agents can be seen in location semantics as collections (multisets) of sequential sub-agents acting in different locations. This intuitive fact, used in [CN94] to represent location agents, gets, in this way, a formal foundation using simple structural axioms.

We denote with $\equiv_\Delta$ the smallest congruence which contains $\equiv$ and which respects equivalence **del**.

Proposition 5.5 *By exploiting $\equiv_\Delta$, every location agent p can be written in the following form:*

$$p \equiv_\Delta (\nu\alpha_1) \cdots (\nu\alpha_m)\,(p_0 | l_1 :: p_1 | \cdots | l_n :: p_n)$$
$$p_i = s_{i1} | \cdots | s_{in_i}$$

where locations l_i are all distinct and s_{ij} are sequential processes.

Lemma 5.6 *If $p' \equiv_\Delta p$, $p \sim_{\Delta l} q$ and $q \equiv_\Delta q'$ then $p' \sim_{\Delta l} q'$.*

We are ready to map location agents into HD-automata. We can assume that the set *Loc* of locations is used as set $\mathcal{N}$ of global names in the HD-automata. We assume also to have a function norm which, given a location agent p, returns a pair (q, σ) defined as follows. Agent q is the representative of the class of agents that differ from p only by the structural congruence $\equiv_\Delta$ and by a renaming of the locations. Agent q can be obtained by first transforming p in an agent p' that is normalized w.r.t. $\equiv_\Delta$; for instance, by resorting on Proposition 5.5, we could have

$$p' = (\nu\alpha_1) \cdots (\nu\alpha_m)\,(l_1 :: p_1 | \cdots | l_n :: p_n).$$

Then the locations of p' can be renamed so that l_i is the i-th location in Loc. Renaming $\sigma : \mathrm{loc}(p) \longleftrightarrow \mathrm{loc}(q)$ associates a location of p to each location of q, so that $q \equiv_\Delta p\sigma$.

When the HD-automaton is built, there is no reason to include all the transitions that differ just in the name assigned to the new location; it is sufficient to use representative transitions: a location transition $p \xrightarrow[ulm]{a} q$ is *representative* if $m = \min\{Loc \setminus \mathrm{loc}(p)\}$.

The named set L^l of the labels in the case of CCS with localities is so defined: the set L^l coincides with the set of the CCS actions. No name is associated to action τ, whereas two names n_{loc} and n_{new} are associated to the visible actions.

Definition 5.7 (from location agents to HD-automata) *The HD-automaton $\mathcal{A}^l_p$ corresponding to the pure CCS agent p, according to the semantics with localities, is the smallest HD-automaton that satisfies the following rules:*

- *let m be a fixed location of Loc; if $\mathrm{norm}(m :: p) = (q_0, \sigma_0)$ then:*
  - *$q_0 \in Q$ is the initial state and $\mathsf{Q}[q_0] = \mathrm{loc}(q_0)$;*
  - *σ_0^{-1} is the initial embedding;*
- *if $t : q \in Q$, $q \xrightarrow[ulm]{a} q'$ is a representative transition and $\mathrm{norm}(q') = (q'', \sigma)$, then:*
  - *$q'' \in Q$ and $\mathsf{Q}[q''] = \mathrm{loc}(q'')$;*
  - *$t \in T$ and $\mathsf{T}[t] = \mathrm{loc}(q) \cup \{m\}$;*
  - *$s(t) = q$, $d(t) = q''$, $\mathsf{s}[t] = \mathrm{id}_{\mathrm{loc}(q)}$ and $\mathsf{d}[t] = \sigma$;*
  - *$o(t) = a$ and $\mathsf{o}[t](l) = n_{\mathrm{loc}}$, $\mathsf{o}[t](m) = n_{\mathrm{new}}$;*
- *if $q \in Q$, $t : q \xrightarrow{\tau} q'$ and $\mathrm{norm}(q') = (q'', \sigma)$ then:*
  - *$q'' \in Q$ and $\mathsf{Q}[q''] = \mathrm{loc}(q'')$;*
  - *$t \in T$ and $\mathsf{T}[t] = \mathrm{loc}(q)$;*
  - *$s(t) = q$, $d(t) = q''$, $\mathsf{s}[t] = \mathrm{id}_{\mathrm{loc}(q)}$ and $\mathsf{d}[t] = \sigma$;*
  - *$o(t) = \tau$.*

Also in this case the HD-automaton corresponding to location agent p is finite whenever agent p is finitary.

Theorem 5.8 *Given a pure location agent p, the HD-automaton $\mathcal{A}^l_p$ is finite if and only if p is finitary according to Definition 2.6.*

The proof is similar to that of Theorem 4.8.

Two pure agents are location bisimilar if and only if the corresponding HD-automata are bisimilar.

Theorem 5.9 *Let p_1 and p_2 be two pure CCS agents. Then $p_1 \sim_l p_2$ iff $\mathcal{A}^l_{p_1} \sim \mathcal{A}^l_{p_2}$.*

Like in the case of the ground semantics of π-calculus, the proof of this theorem relies on the global characterization of HD-bisimulation. It exploits the following lemma, that is analogous to Lemma 4.20.

Lemma 5.10 *Let (q, σ) be a global state of the HD-automaton $\mathcal{A}^l_p$ corresponding to a pure location agent p. Then:*

- *if $q\sigma \xrightarrow[umn]{a} q''$ is a location transition with $n \notin \mathrm{loc}(q\sigma)$, and $(q', \sigma') = \mathrm{norm}(q'')$, then there is some global transition $(t, \rho) : (q, \sigma) \xrightarrow{(a,\lambda)} (q', \sigma')$ of $\mathcal{A}^l_p$ and $q'' = q'\sigma'$; and*
- *if $(t, \rho) : (q, \sigma) \xrightarrow{(a,\lambda)} (q', \sigma')$ is a global transition of $\mathcal{A}^l_p$, then there is some location transition $q\sigma \xrightarrow[umn]{a} q'\sigma'$*

where in both cases $\lambda(n_{\mathrm{loc}}) = m$ and $\lambda(n_{\mathrm{new}}) = n$. A similar property holds for the τ transitions.

5.3 Petri nets

In this section we consider an example of history-dependent formalism outside the field of process calculi, namely Petri nets with labelled transitions. In the case of Petri nets, automata have been used to represent the *case graphs* of the nets, i.e., the graphs of the reachable markings of the nets; a transition of the automaton represents the simultaneous firing of a set of transitions of the net. Bisimulation can then be applied to the case graphs. In the case of Petri nets, however, a refined notion of equivalence is preferred, that also considers the causal relations on the firings of the net.

Processes have been defined in [GR83] to represent concurrent runs of nets. From a process, it is possible to derive a partial order of the events of the run, which represents the dependencies between them. A notion of bisimulation, called *history-preserving bisimulation,* that takes into account the partial order behavior has been defined in [RT88] for event structures. The same notion has been introduced before in [DDNM90] for process calculi, using mixed ordering observations. History-preserving bisimulation has been applied to Petri nets in [BDKP91]: for two nets to be equivalent, it is required not only that they perform the same sequence of actions (with the same branching structure), as in ordinary bisimulation, but also that the partial orders corresponding to two matching computations are isomorphic, i.e., that the causal dependencies between the actions are the same in the two nets. As we will see, also in this case the definition of bisimulation is not the standard one, and suffers of problems similar to the ones of history-dependent calculi.

Now we present the basic definitions on Petri nets. Most of the definitions and of the notations are from [GR83].

Definition 5.11 (net) *A* net N *is a tuple* (S, T, F) *where:*

- S *is a set of* places *and* T *is a set of* transitions*; we assume* $S \cap T = \emptyset$*;*
- $F \subseteq (S \times T) \cup (T \times S)$ *is the* flow relation.

If $x \in S \cup T$ *then* $^\bullet x = \{y \mid (y, x) \in F\}$ *and* $x^\bullet = \{y \mid (x, y) \in F\}$ *are called respectively the* pre-set *and the* post-set *of* x. *Let* $^\circ N = \{x \in S \cup T \mid {}^\bullet x = \emptyset\}$ *and* $N^\circ = \{x \in S \cup T \mid x^\bullet = \emptyset\}$.
A net N *is* finite *if* S *and* T *are finite sets.*

Definition 5.12 (P/T nets) *A* (labelled, marked) place/transition net *(*P/T net *in brief)* N *is a tuple* (S, T, F, W, l, m_0), *where* (S, T, F) *is a net and:*

- $W : F \to \omega^+$ *assigns a positive* weight *to each arc of the net; we sometimes assume that* W *is defined on* $(S \times T) \cup (T \times S)$ *by requiring* $W(x, y) = 0$ *if* $(x, y) \notin F$*;*
- $l : T \to Act$ *is the* labeling function*, where* Act *is a fixed set of action labels;*
- $m_0 : S \to \omega$ *is the* initial marking.

A marking *is a mapping* $m : S \to \omega$. *It represents a distribution of the* tokens *in the places of the net.*
Transition $t \in T$ *is* enabled *at marking* m *if* $m(s) \geq W(s, t)$ *for all* $s \in {}^\bullet t$. *In this case, the* firing *of* t *at* m *produces the marking* m' *with* $m'(s) = m(s) + W(t, s) - W(s, t)$, *and we write* $m \xrightarrow{t} m'$.

Definition 5.13 (n-safe nets) *A P/T net* N *is* n*-safe if for each reachable marking* m *(i.e., for each* m *such that* $m_{0N} \longrightarrow \cdots \longrightarrow m$*) we have:*

$$m(s) \leq n \qquad \forall s \in S_N.$$

Definition 5.14 (occurrence net) *An occurrence net is a net* $K = (C, E, G)$ *(in this case, states are also called* conditions *and transitions are also called* events*) such that:*

- *for all* $c \in C$, $|{}^\bullet c| \leq 1$ *and* $|c^\bullet| \leq 1$ *(conditions are not branching), and*
- *the transitive closure* G^+ *of* G *is irreflexive (the net is acyclic).*

Definition 5.15 (process) *A* process π *of a P/T net* N *is a tuple* (C, E, G, p), *where* $K = (C, E, G)$ *is a finite occurrence net and* $p : (C \cup E) \to (S_N \cup T_N)$ *is such that:*

- $p(C) \subseteq S_N$ *and* $p(E) \subseteq T_N$*;*
- $m_0(s) = |p^{-1}(s) \cap {}^\circ K|$ *for all* $s \in S_N$*;*
- $W_N(s, p(e)) = |\{c \in {}^\bullet e \mid p(c) = s\}|$ *and* $W_N(p(e), s) = |\{c \in e^\bullet \mid p(c) = s\}|$ *for all* $e \in E$ *and all* $s \in S_N$.

We write $^\circ\pi$ *for* $^\circ K$ *and* π° *for* K°.
The initial process *of net* N *is the*[4] *process* $\pi_0(N)$ *with an empty set of events.*
Let $\pi = (C, E, G, p)$ *and* $\pi' = (C', E', G', p')$ *be two processes of* N. *If:*

- $E' = E \cup \{\bar{e}\}$ *for some* $\bar{e} \notin E$;
- $C' \supseteq C$;
- $p'\big|_{C \cup E} = p$

then we write $\pi \xrightarrow{\bar{t}} \pi'$, *where* $\bar{t} = p'(\bar{e})$.

Now we define history-preserving bisimulation. We follow a classical characterization, as it appears in [BDKP91] under the name of *fully concurrent bisimulation*.

Definition 5.16 (event structure) *The* (deterministic) event structure *for process* $\pi = (C, E, G, p)$ *of net* N *is the tuple* $\mathrm{ev}(\pi) = (E, F^+\big|_E, l_N \circ p\big|_E)$. *An* isomorphism *between two event structures is a bijective function between their events which respects ordering and labels.*

Definition 5.17 (history-preserving bisimulation) *A set* $\mathcal{R}$ *of triples is a* history-preserving bisimulation *for nets* N_1 *and* N_2 *if:*

- *whenever* $(\pi_1, f, \pi_2) \in \mathcal{R}$ *then* π_1 *is a process of* N_1, π_2 *is a process of* N_2 *and* f *is an isomorphism between* $\mathrm{ev}(\pi_1)$ *and* $\mathrm{ev}(\pi_2)$;
- $(\pi_0(N_1), \emptyset, \pi_0(N_2)) \in \mathcal{R}$;
- *whenever* $(\pi_1, f, \pi_2) \in \mathcal{R}$ *and* $\pi_1 \xrightarrow{t_1} \pi_1'$ *then* $\pi_2 \xrightarrow{t_2} \pi_2'$ *with* $(\pi_1', f', \pi_2') \in \mathcal{R}$ *and* $f'\big|_{\mathrm{ev}(\pi_1)} = f$;
- *whenever* $(\pi_1, f, \pi_2) \in \mathcal{R}$ *and* $\pi_2 \xrightarrow{t_2} \pi_2'$ *then* $\pi_1 \xrightarrow{t_1} \pi_1'$ *with* $(\pi_1', f', \pi_2') \in \mathcal{R}$ *and* $f'\big|_{\mathrm{ev}(\pi_1)} = f$.

Two nets N_1 *and* N_2 *are* history-preserving bisimilar, *written* $N_1 \sim_{hp} N_2$, *if there is a history-preserving bisimulation for them.*

Notice that the definition of bisimulation which is applied in this case is not the standard one on labelled transition systems. First of all, the bisimulation must deal with isomorphisms between partial orders and hence it cannot be simply a relation on states. Moreover, since processes and partial orders grow during a computation, it is possible to associate finite-state systems only to nets which cannot perform infinite sequences of actions.

5.4 Representing Petri nets as basic HD-automata

Now we show that also the history-preserving semantics for Petri nets can be modeled by HD-automata. In order to obtain finite HD-automata also for nets with infinite behaviors we propose now an alternative characterization of history-preserving bisimulation on Petri nets, in which part of the past history can be forgotten, like we have done in the previous section for CCS with localities.

The first step consists of the definition of *configurations*, which are suitable to represent in a compact way the relevant part of the past history for generic P/T nets. We also show how processes can be mapped into configurations.

Definition 5.18 (configurations) *Let* N *be a P/T net. A* configuration *for* N *is a tuple* $c = (E, \leq, \rho)$, *where:*

- $E \subseteq \mathcal{N}$ *is a set of* events *and* $\leq \subseteq E \times E$ *is a partial ordering for* E;
- $\rho : S_N \times (E \cup \{\mathrm{init}\}) \to \omega$.

We require that $\sum_{s \in S_N} \rho(s, e) > 0$ *for each* $e \in E$.
The initial configuration *of net* N *is configuration* $c_0(N) = (\emptyset, \rho_0, \emptyset)$, *with* $\rho_0(s, \mathrm{init}) = m_0(s)$ *for all* $s \in S$.
Let $c = (E, \leq, \rho)$ *and* $c' = (E', \leq', \rho')$ *be two configurations for* N *and* $\bar{t} \in T$ *be a transition of* N. *If*[5]*:*

- $E' \subseteq E \cup \{\bar{e}\}$ *for some* $\bar{e} \notin E$;

[4] Notice that the initial process of a net is unique only up to isomorphism of the set of initial conditions.
[5] For simplicity, in this definition we suppose that $\rho(s, e) = 0$ if $e \notin E$ and, similarly, $\rho'(s, e) = 0$ if $e \notin E'$.

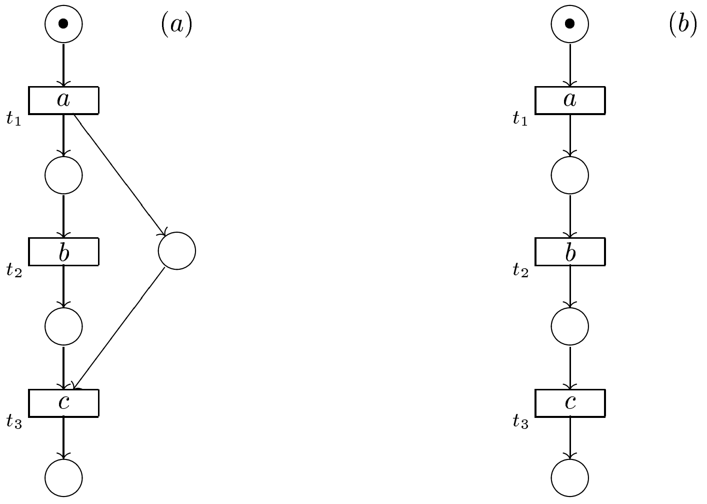

Figure 5: Two bisimilar nets with different sets of immediate causes

- $\leq' = \big(\leq \cup \{(e,\bar{e}) \mid \exists e' \in \mathcal{IC}(c \xrightarrow{\bar{t}}_{\bar{e}} c').\, e \leq e'\}\big) \cap (E' \times E')$*;*
- $\rho(s,e) \geq \rho'(s,e)$ *for all* $s \in S_N$ *and* $e \in (E \cup \{\mathrm{init}\})$*, and* $\sum_{e \in (E \cup \{\mathrm{init}\})}(\rho(s,e) - \rho'(s,e)) = W_N(s,\bar{t})$ *for all* $s \in S_N$*;*
- $\rho'(s,\bar{e}) = W_N(\bar{t},s)$ *for all* $s \in S_N$*;*

then we write $c \xrightarrow{\bar{t}}_{\bar{e}} c'$*, where the set* $\mathcal{IC}(c \xrightarrow{\bar{t}}_{\bar{e}} c')$ *of the* immediate causes *of the transition is:*

$$\mathcal{IC}(c \xrightarrow{\bar{t}}_{\bar{e}} c') = \{e \in E \mid \exists s \in S : \rho'(s,e) < \rho(s,e)\}.$$

The set $\mathcal{MC}(c \xrightarrow{\bar{t}}_{\bar{e}} c')$ *of the* maximal causes *of the transition contains the elements of* $\mathcal{IC}(c \xrightarrow{\bar{t}}_{\bar{e}} c')$ *which are maximal w.r.t.* $\leq$*.*
The transition $c \xrightarrow{\bar{t}}_{\bar{e}} c'$ *is* representative *if* $\bar{e} = \min\{\mathcal{N} \setminus E\}$*.*

In a configuration, the set E of events represents the past events which are still referenced. Since we are interested in a partial order semantics, a partial order is defined on E, which represents the causal dependencies between the past events. Function ρ represents the current marking of the net; rather than simply defining how many tokens are in each place of the net, it also remembers which events generated these tokens (init is a special mark used for the tokens in the initial marking).

We require that in a configuration only the events are remembered which generated tokens still present in the net. This is important to obtain a finite number of different configurations also for certain classes of nets with infinite behaviors.

A transition between two configurations corresponds to the firing of a transition of the net. A new event $\bar{e}$ is generated: it directly depends on those events of the source configuration which correspond to the tokens consumed by the transition (these events are called the *immediate causes* of the transition) and the partial order in the destination configuration respects these dependencies. The marking of the destination configuration is obtained from the marking of the source by discarding tokens according to the pre-set of the transition and by adding new tokens according to the post-set (these tokens are associated to the new event $\bar{e}$). Events with no tokens in the marking are discarded and do not appear in the destination configuration.

It is important to remark that corresponding events of history-preserving bisimilar nets can have different sets of immediate causes. In fact, if we consider the net in Figure 5(a), we see that both t_1 and t_2 are immediate causes of t_3, whereas in the net in Figure 5(b) t_1 is not a direct cause of t_3. It is possible to prove, instead, that two matching events must have the same sets of maximal causes. In fact, notice that t_1 is not a maximal cause of t_3 in both nets of our example.

The transitions between configurations correspond to firings between the corresponding markings:

Definition 5.19 (from configurations to markings) *Let* N *be a P/T net and let* $c = (E, \leq, \rho)$ *be a configuration for* N*. The marking* m_c *corresponding to* c *is defined as follows:*

$$m_c(s) = \sum_{e \in E} \rho(s,e) \quad \text{for each } s \in S.$$

Proposition 5.20 *Let c be a process for net N and let m_c be the corresponding marking. If $c \xrightarrow{t}_{\bar{e}} c'$ then $m_c \xrightarrow{t} m_{c'}$ and, conversely, if $m_c \xrightarrow{t} m'$ then there exists some configuration c' such that $c \xrightarrow{t}_{\bar{e}} c'$ and $m' = m_{c'}$.*

The transitions between configurations correspond also to the transitions between the processes of the net.

Definition 5.21 (from processes to configurations) *Let N be a P/T net and let $\pi = (C, E, G, p)$ be a process on N. The configuration corresponding to π is $c_\pi = (E', \leq, \rho)$ which is defined as follows:*

- *$E' = \{e \in E \mid \pi^\circ \cap e^\bullet \neq \emptyset\}$ and $\leq = G^*\big|_{E'}$;*
- *$\rho(s, e) = |p^{-1}(s) \cap \pi^\circ \cap e^\bullet|$ for all $s \in S_N$ and $e \in E$; moreover $\rho(s, \text{init}) = |p^{-1}(s) \cap \pi^\circ \cap {}^\circ\pi|$ for all $s \in S_N$.*

Notice that the configuration corresponding to the initial process for N is precisely the initial configuration for N. The following proposition shows that the transitions on configurations exactly match the transitions on processes.

Proposition 5.22 *Let π be a process for net N and let c_π be the corresponding configuration. If $\pi \xrightarrow{t} \pi'$ then $c_\pi \xrightarrow{t}_{\bar{e}} c_{\pi'}$ and, conversely, if $c_\pi \xrightarrow{t}_{\bar{e}} c'$ then there exists some process π' such that $\pi \xrightarrow{t} \pi'$ and $c' = c_{\pi'}$.*

Now we introduce an alternative notion of history-preserving bisimulation, which is based on configurations rather than on processes.

Definition 5.23 (incremental history-preserving bisimulation) *A set $\mathcal{R}$ of triples is an* incremental history-preserving bisimulation *for nets N_1 and N_2 if:*

- *whenever $(c_1, f, c_2) \in \mathcal{R}$ then c_1 is a configuration of N_1, c_2 is a configuration of N_2 and $f : E_{c_1} \hookrightarrow E_{c_2}$ is a partial correspondence between the events of c_1 and c_2;*
- *$(c_0(N_1), \emptyset, c_0(N_2)) \in \mathcal{R}$;*
- *whenever $(c_1, f, c_2) \in \mathcal{R}$ and $c_1 \xrightarrow{t_1}_{\bar{e}_1} c_1'$ then $c_2 \xrightarrow{t_2}_{\bar{e}_2} c_2'$ with:*
  - *$f(\mathcal{MC}(c_1 \xrightarrow{t_1}_{\bar{e}_1} c_1')) = \mathcal{MC}(c_2 \xrightarrow{t_2}_{\bar{e}_2} c_2')$,*
  - *$l_1(t_1) = l_2(t_2)$, and*
  - *$(c_1', f', c_2') \in \mathcal{R}$, and $f' = (f \cup (\bar{e}_1, \bar{e}_2)) \cap (E_{c_1'} \times E_{c_2'})$;*
- *the converse, starting from the transitions of c_2.*

Two nets N_1 and N_2 are incremental history-preserving bisimilar*, written $N_1 \sim_{\Delta hp} N_2$, if there is an incremental history-preserving bisimulation for them.*

Proposition 5.24 *Let N_1 and N_2 be two P/T nets. Then $N_1 \sim_{hp} N_2$ if and only if $N_1 \sim_{\Delta hp} N_2$.*

Proof (Sketch). Let $\mathcal{R}$ be a history-preserving bisimulation for N_1 and N_2; then:

$$\mathcal{R}' = \{(c_{\pi_1}, g, c_{\pi_2}) \mid (\pi_1, f, \pi_2) \in \mathcal{R},\ g = f \cap (E_{c_{\pi_1}} \times E_{c_{\pi_2}})\}$$

is an incremental history-preserving bisimulation. This is sufficient to prove that $N_1 \sim_{hp} N_2$ implies $N_1 \sim_{\Delta hp} N_2$.
For the converse, let $\mathcal{R}$ be an incremental history-preserving bisimulation. Then

$$\mathcal{R}' = \{(\pi_1, f, \pi_2) \mid (c_{\pi_1}, g, c_{\pi_2}) \in \mathcal{R}, \\ f \text{ is an isomorphism for } \text{ev}(\pi_1) \text{ and } \text{ev}(\pi_2) \text{ such that } g = f \cap (E_{c_{\pi_1}} \times E_{c_{\pi_2}})\}$$

is a history-preserving bisimulation. This is sufficient to prove that $N_1 \sim_{\Delta hp} N_2$ implies $N_1 \sim_{hp} N_2$. □

We would like to remark that configurations and incremental history preserving bisimulation are strongly related to generalized OM-markings and generalized OM-bisimulation defined in [Vog95]. Also generalized OM-markings are obtained from processes by discarding most of the past history and by retaining only the information on the events that are still active, and on their causal relation. In that paper, OM-bisimulation is proved decidable for 1-safe nets. Essentially, what we have done so far in this section is to extend the definitions of [Vog95] to the more general case of P/T nets.

When a HD-automaton is generated from a net, states of the automaton correspond to configurations of the net. However, to obtain a compact HD-automaton, it is important to identify configurations which are isomorphic.

Definition 5.25 (isomorphic configurations) *Two configurations $c = (E, \leq, \rho)$ and $c' = (E', \leq', \rho')$ of a P/T net N are* isomorphic *if there exists some bijective function $i : E \longleftrightarrow E'$ such that:*

- *$e \leq e'$ if and only if $i(e) \leq' i(e')$ for all $e, e' \in E$, and*
- *$\rho(e, s) = \rho'(i(e), s)$ for all $e \in E$, $s \in S_N$.*

We assume to fix a representative for each class of isomorphic configurations and to have a function norm *such that $\mathrm{norm}(c) = (c', \sigma)$ where c' is the representative of the class of configurations isomorphic to c and σ is the bijection between E_c and $E_{c'}$.*
Let c and c' be two isomorphic configurations and let σ be the total bijection between E_c and $E_{c'}$ which corresponds to the isomorphism. Then $\sigma(c) \overset{\mathrm{def}}{=} c'$.

Now we are ready to show how, given a net, it is possible to build the HD-automaton corresponding to it, by exploiting its behavior on configurations.

In this case, the named set L^{pn} of labels is defined as follows: $L^{\mathrm{pn}} = \{(\alpha, k) \mid \alpha \in Act \wedge k \in \omega\}$ and the $k+1$ names $n_1, \ldots, n_k, n_{\mathrm{new}}$ correspond to (α, k) for any $\alpha \in Act$.

Definition 5.26 (from Petri nets to HD-automata) *The HD-automaton $\mathcal{A}_N^{\mathrm{pn}}$ that corresponds to the P/T net N is the smallest HD-automaton that satisfies the following rules:*

- *let c_0 be the initial configuration for N and $(c'_0, \sigma_0) = \mathrm{norm}(c_0)$; then:*
  - *$c'_0 \in Q$ is the initial state and $\mathsf{Q}[c'_0] = E_{c'_0}$;*
  - *σ_0^{-1} is the initial embedding;*
- *if $c \in Q$, $c \xrightarrow{\bar{t}}_{\bar{e}} c'$ is a representative transition, $\mathcal{MC}(c \xrightarrow{\bar{t}}_{\bar{e}} c') = \{e_1, \ldots, e_k\}$, and $\mathrm{norm}(c') = (c'', \sigma)$, then:*
  - *$c'' \in Q$ and $\mathsf{Q}[c''] = E_{c''}$;*
  - *there is some $t \in T$ such that $\mathsf{T}[t] = E_c \cup \{e_0\}$, where $e_0 = \min\{\mathcal{N} \setminus E_c\}$; moreover*
  - *$s(t) = c$, $d(t) = c''$, $\mathsf{s}[t] = \mathrm{id}_{E_c}$ and $\mathsf{d}[t] = \sigma$;*
  - *$o(t) = (\alpha, k)$, where $\alpha = l_N(t)$; moreover $\mathsf{o}[t](e_j) = n_j$ for $j = 1, \ldots, k$ and $\mathsf{o}[t](\bar{e}) = n_{\mathrm{new}}$.*

For each representative transition $c \xrightarrow{\bar{t}}_{\bar{e}} c'$ there are in general many transitions in the HD-automaton, that differ for the order of the elements of $\mathcal{MC}(c \xrightarrow{\bar{t}}_{\bar{e}} c')$: if k elements appear in $\mathcal{MC}(c \xrightarrow{\bar{t}}_{\bar{e}} c')$, then there are $k!$ ways to assign them to the names $n_1, \ldots, n_k$ of the label. All these different transitions have the same target state, so no grow of the number of states occurs due to this inefficient encoding of the labels. Moreover, it is possible to generate a single transition in the HD-automaton, at the cost of changing slightly the definition of HD-bisimulation, so that two transitions are allowed to match even if their labels differ for a permutation of names $n_1, \ldots, n_k$.

The construction in Definition 5.26 generates finite HD-automata for the finite nets that are n-safe for some n. We emphasize that it is decidable whether a finite P/T net is n-safe for some n. A possible procedure can be found in [VJ85].

Theorem 5.27 *Given a finite P/T net N, the HD-automaton $\mathcal{A}_N^{\mathrm{pn}}$ is finite if and only if N is n-safe for some n.*

Proof. We show that $\mathcal{A}_N^{\mathrm{pn}}$ has a finite number of states. Since a finite number of steps is possible in a net from a particular marking, also the number of transitions exiting from each state of the HD-automaton has to be finite, which concludes the proof.
Each state of $\mathcal{A}_N^{\mathrm{pn}}$ is a configuration $c = (E, \leq, \rho)$ of N, where:

- $E \subseteq \mathcal{N}$ and $\leq \subseteq E \times E$ is a partial ordering;
- $\rho : S \times (E \cup \{\mathrm{init}\}) \to \omega$.

By Proposition 5.20, moreover m_c is a marking of N which is reachable from the initial marking. Since the net is n-safe for some n, there can be just a finite number of different markings m_c corresponding to reachable configurations.
Since we require that $\sum_{s \in S} \rho(s, e) > 0$, the number of events $e \in E$ cannot be greater than the total number of the tokens in m_c, so there is also a bound to the cardinality of E and also the possible partial orders of E are finite. Since we work up to isomorphism of the configurations, this is sufficient to conclude the proof. $\square$

Two P/T nets are history-preserving bisimilar if and only if the corresponding HD-automata are bisimilar.

Theorem 5.28 *Let N_1 and N_2 be two P/T nets. Then $N_1 \sim_{hp} N_2$ iff $\mathcal{A}_{N_1}^{\mathrm{pn}} \sim \mathcal{A}_{N_2}^{\mathrm{pn}}$.*

To prove this result we exploit the global characterization of HD-bisimulation; it is based on the incremental characterization of history-preserving bisimulation and on the following lemma, which corresponds to Lemma 4.20 in the case of the ground semantics of π-calculus.

Lemma 5.29 *Let (c, ρ) be a global state of the HD-automaton $\mathcal{A}_N^{\mathrm{pn}}$ corresponding to a P/T net N. Then:*

- *if $\sigma(c) \xrightarrow{t}_{\bar{e}} c''$ and $(c', \sigma') = \mathrm{norm}(c'')$, then there is some global transition $(t', \rho) : (c, \sigma) \xrightarrow{(\alpha,\lambda)} (c', \sigma')$ of $\mathcal{A}_N^{\mathrm{pn}}$ and $c'' = \sigma'(c')$; and*

- *if $(t', \rho) : (c, \sigma) \xrightarrow{(\alpha,\lambda)} (c', \sigma')$ is a global transition of $\mathcal{A}_N^{\mathrm{pn}}$, then there is some $\sigma(c) \xrightarrow{t}_{\bar{e}} \sigma'(c')$*

where in both cases $\alpha = l_N(t)$ and $\mathrm{cod}(\lambda) = \mathcal{MC}(c\sigma \xrightarrow{t}_{\bar{e}} c'\sigma') \cup \{\bar{e}\}$.

6 HD-automata with symmetries

Basic HD-automata, presented in Section 4, have some limitations. The most important are that they apply only to the *ground* semantics of the π-calculus, and do not apply to the *early* (and *late*) semantics; and that they do not admit minimal realizations. In order to overcome these limitations, in this section we define an extended class of HD-automata, namely HD-automata with Symmetries (HDS-automata).

We start by describing the problems of mapping the early π-calculus semantics on Basic HD-automata (Section 6.1). Then we define symmetries on names (Section 6.2) and, based on them, HDS-automata (Section 6.3); we also map the early semantics of π-calculus into HDS-automata. Then we introduce HDS-bisimulation and study its basic properties (Section 6.4). Finally, we define minimization on HDS-automata (Section 6.5).

6.1 Motivations

The π-calculus semantics provides two ways to introduce fresh names in an agent: name extrusion in bound output transitions, and fresh name reception in input transitions. These two forms of name generation are similar in the ground semantics, and we have seen that Basic HD-automata model them in the same way. In the case of the early semantics instead, input transitions are different, and Basic HD-automata are not able to model them. In fact, in the early π-calculus we have to take into account that the name received in an input transition may be either a name already present in the source agent or a fresh name. Difficulties occur when two equivalent agents have different sets of free names, since they do not agree on which names are "already present in the source" and which names are "fresh". Consider for instance agents

$$p = x(y).A(x, y, z) \qquad \text{and} \qquad q = x(y).B(x, y, w)$$

where we assume that $A(x, y, z) \sim B(x, y, w)$. The two agents are bisimilar, but they have different sets of free names. If we want to map these agents on HD-automata, we do not want to generate all the input transitions: it is sufficient to consider those corresponding to the reception of a name already present in the source state, and one additional transition where a fresh name is received. However, since agents p and q have different sets of free names, this approach leads to different sets of transitions: the transition corresponding to name z is considered only for agent p, while the one corresponding to name w is considered only for q. Therefore the two obtained HD-automata are not HD-bisimilar according to the definition of HD-bisimulation given in Section 4.2.

The fact that Basic HD-automata are not able to distinguish the two forms of fresh name generation of π-calculus agents is already recognized in [MP98a]. There, this distinction is obtained by typing the fresh names introduced in a transition: bound-output-like names are typed *new*, while fresh-input-like names are typed *universal*. HD-bisimulation has to be complicated to deal with these two different kinds of fresh names; in fact, to be sure that the two agents p and q above are equivalent, we have to match transition $p \xrightarrow{xz} A(x, z, z)$ of agent p against the transition $q \xrightarrow{xy} B(x, y, y)$ of q that corresponds to the reception of the universal name. The intuition is that this transition of q corresponds to the reception of any name different from x and w (that are the free names of q) and should hence be used to match the transition of p for name z.

Here we present a generalization of the approach of [MP98a]. The idea is to extend the sets of names that enrich states, transitions and labels of the HD-automata with *name repositories*: these are infinite sources, from which it is possible to extract fresh names when needed. Different classes of fresh names are represented by different repositories, and the approach of [MP98a] can be seen as a particular case, where only two repositories are allowed, one for the universal names and one for the new names. In Figure 6 we represent two π-calculus transition: the upper corresponds to a bound output, while the lower corresponds to an input transition. Two repositories are present in the states, as well

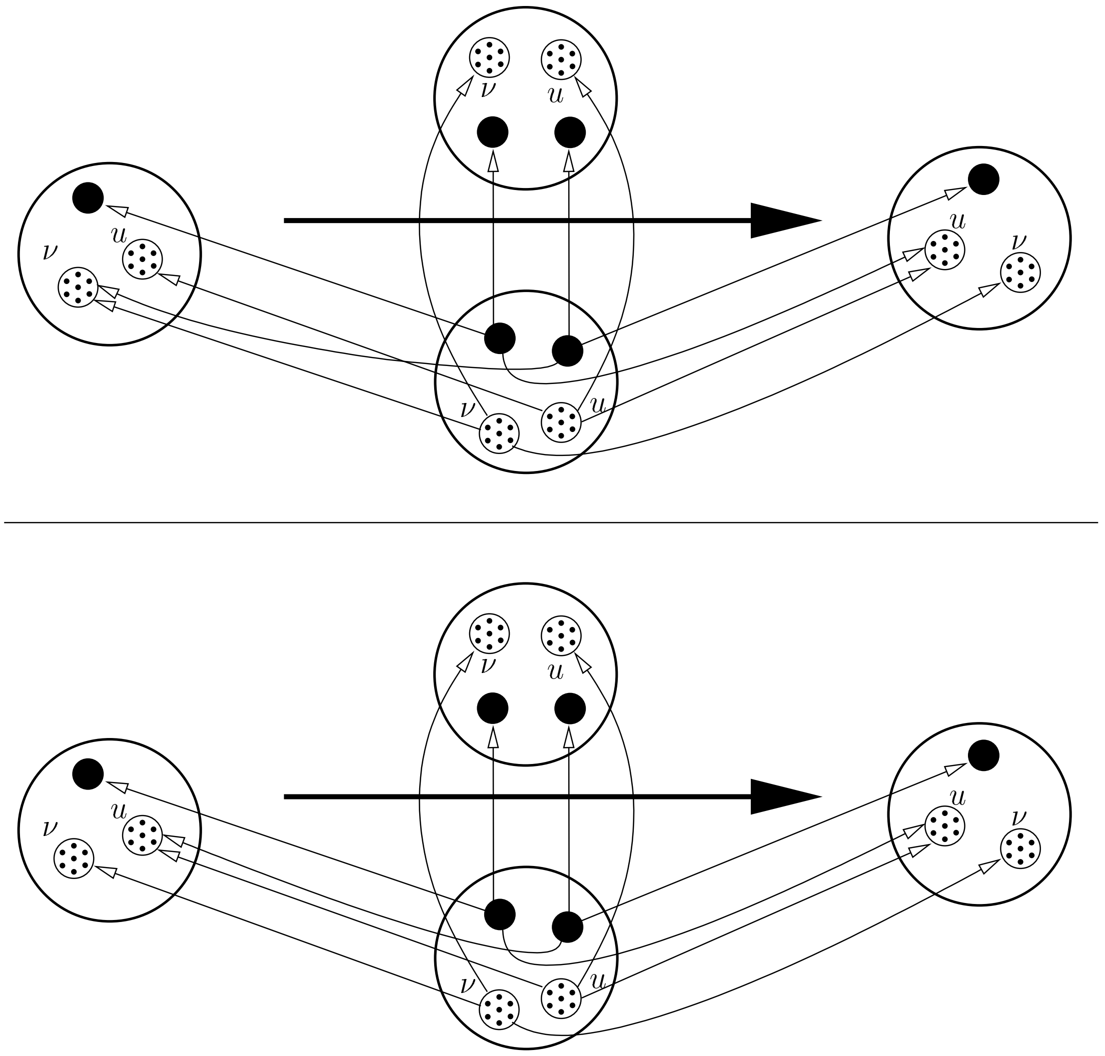

Figure 6: Two transitions with infinite repositories

as in the transitions and in the labels: these are named ν and u in figure. In the case of the bound output transitions, fresh names are taken from the infinite repository ν. In the case of the input transitions, instead, the fresh names are taken from repository u.

Repository u is used also as a "drain" for the names that are discarded in a transition. To understand why it is correct to use u as the drain, we have to think at the role that play the two repositories in the π-calculus. Names in ν are used to represent the fresh names that can be generated locally by an agent. Names in u, instead, are those names that can be sent to the agent by the environment: that is, these names represent channels that already exist in the environment, but that the agent does not know. If an agent forgets a name, that name is still available in the environment, and the agent can receive it back in a successive input action. Hence, it is correct to put this names among those that can be received in an input transition.

In the Basic HD-automata of Section 4, only a finite set of names is associated to a state of a HD-automaton. In the extended model that we are defining, the whole set of names is represented; however, many of these names are indistinguishable and are represented in a compact way by means of the infinite repositories. According to this interpretation, in the case of a π-calculus agent the names are divided in the parts:

- a finite set of names that appear syntactically in the agent — these appear explicitly in the corresponding state of the HD-automaton;
- an infinite set of names that exist in the environment, but that are not known by the agent — these are represented by repository u;
- an infinite set of names that can be generated locally, when the agent performs a bound output — these are

represented by repository ν.

In this chapter we implement this idea by means of *History Dependent Automata with Symmetries*, in brief *HDS-automata*. The names that appear in their states, transitions, and labels are defined up to a *symmetry* — that is, a set of bijective functions, or permutations, on these names. These symmetries are used to identify the infinite subsets of these names that correspond to the different repositories. The symmetry of a state can be thought as a declaration that the behavior from that state is not altered by applying to the names of the states any of the corresponding permutations.

In the case of infinite name repositories, all the names in the same repository are interchangeable, so, in this case the symmetry is used to define an equivalence on the names. Symmetries can be used to represent more complex behaviors, though; for instance, in the case of agent

$$p = A(w,x,y) + A(w,y,z) + A(w,z,x),$$

we can use them to declare that the behavior of agent p is left unchanged by a shift $\begin{tikzcd} x \arrow[r, bend left] & y \arrow[r, bend left] & z \arrow[ll, bend left] \end{tikzcd}$: in this way, it is sufficient to represent explicitly only the transition of, say, $A(w,x,y)$.

Another important aspect of symmetries it that they make it possible to define minimal HD-automata. As we will see, minimization is not possible in the case of Basic HD-automata precisely since it is not possible to represent explicitly the symmetries defined by HD-bisimulation on the names of the states of the automata.

6.2 Symmetries on Names

In this section we define *symmetries* on names and functions between them.

Definition 6.1 (symmetries) *Let $\mathcal{N}$ be a infinite, denumerable set of names. A* symmetry Σ *on $\mathcal{N}$is a set of bijections on $\mathcal{N}$ that is a group for composition; that is:*

- $\mathrm{id}_{\mathcal{N}} \in \Sigma$ *(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 $\sigma;\sigma' = \mathrm{id}_{\mathcal{N}}$ (i.e., $\mathcal{N}$ is closed for inversion).*

We denote the set of all the symmetries on $\mathcal{N}$ with $\mathcal{S}ym(\mathcal{N})$.

While generic symmetries may appear in the HDS-automata, in most of the examples we use infinite repositories of names. Here we introduce some notations for them.

Notation 6.2 *We denote with $\left\{\!\left| d_1, \ldots, d_m, \textcircled{s_1}, \ldots, \textcircled{s_n} \right|\!\right\}$ the symmetry where $d_1, \ldots, d_m$ are the* distinct names *and $s_1, \ldots, s_n$ denote the* infinite repositories. *More precisely, $D = \{d_1, \ldots, d_m\} \subseteq \mathcal{N}$ is the finite subset of the names that are neutral for the permutations in the symmetry, i.e., name d_h for $h = 1 \cdots m$ is left unchanged by all the symmetries. Names in $\mathcal{N} \setminus D$ are split into n infinite sets $S_1, \ldots, S_n$, that correspond to the infinite repositories $s_1, \ldots, s_n$; all the names in set S_i are considered indistinguishable by the symmetry. Clearly, there are infinite many different ways to split set $\mathcal{N} \setminus D$ into the n subsets $S_1, \ldots, S_n$: here we assume to fix one of these split as the canonical one.*[6] *Therefore, if $\{S_1, \ldots, S_n\}$ is the canonical split of $\mathcal{N} \setminus D$ in n infinite sets, then*

$$\left\{\!\left| d_1, \ldots, d_m, \textcircled{s_1}, \ldots, \textcircled{s_n} \right|\!\right\} = \{\sigma \mid \sigma(d_h) = d_h \text{ for } h = 1 \cdots m \text{ and } \\ \sigma(S_k) = S_k \text{ for } k = 1 \cdots n\}.$$

With some abuse of notation, in the following we denote with $\textcircled{s_k}$ the set S_k corresponding to the names of the infinite repository s_k. So, for instance, to represent the fact that n is a name of infinite repository s_k we will write $n \in \textcircled{s_k}$ rather than $n \in S_k$.

As we will see in the following section, each state, label, and transition of a HDS-automaton is enriched by a symmetry on $\mathcal{N}$. Moreover, the HDS-automaton defines the correspondences between the symmetry that enrich a transition and those that enrich its source state, target state, and label. This is similar to what happens in the case of Basic HD-automata: in that case the correspondences are defined by means of inverse injections; in the case of HDS-automata they are defined by *embeddings on symmetries*.

[6]For instance, by exploiting the fact that $\mathcal{N}$ is countable, and by denoting with n_i the i-th name in $\mathcal{N}$, we can define $S_k = \{n_i \mid n_i \notin D \text{ and } k = i \bmod n+1\}$. In any case, since all the splits give rise to isomorphic symmetries, it is not important which split is taken as the canonical one.

Let Σ and Σ' be two symmetries on $\mathcal{N}$ and let ρ be a bijective function on $\mathcal{N}$. Assume that all the permutations of Σ also appear in Σ' via the function ρ, i.e., that $\Sigma \subseteq \rho; \Sigma'; \rho^{-1}$, 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'. Notice, however, 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 6.3 (embeddings on symmetries) *Let Σ and Σ' be two symmetries on $\mathcal{N}$. An* embedding *f of Σ into Σ' (written $f : \Sigma \to \Sigma'$) is a set of bijections on $\mathcal{N}$ such that:*

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

If symmetry Σ contains an infinite repository s, an embedding f has to map the names of repository s into the names of a repository s' of Σ': this is necessary to guarantee that all the symmetries that exist among the names of s in Σ are reflected in Σ'. Moreover, for each distinct name d' of Σ' there must be exactly one distinct name d of Σ that is mapped to d'. However, it is possible that also some distinct names of Σ are mapped into s', or that two repositories s_1 and s_2 of Σ are mapped into the same repository s' of Σ': all these cases respect the rule that Σ' can have more permutations than Σ.

We introduce with an example some useful notation for the embeddings between symmetries. Let us consider $\Sigma = \{\!\!|\, d_1, d_2, d_3, \textcircled{s_1}, \textcircled{s_2} \,|\!\!\}$ and $\Sigma' = \{\!\!|\, d_1', d_2', \textcircled{s'} \,|\!\!\}$. The embedding from Σ to Σ' that maps distinct names d_1 and d_2 of Σ into distinct names d_1' and d_2' of Σ' respectively, and all the other names of Σ into repository s' of Σ' is represented as follows:

$$\left\{\!\!\left|\begin{array}{c} d_1 \mapsto d_1' \\ d_2 \mapsto d_2' \\ d_3 \mapsto \textcircled{s'} \\ \textcircled{s_1} \mapsto \textcircled{s'} \\ \textcircled{s_2} \mapsto \textcircled{s'} \end{array}\right|\!\!\right\} \stackrel{\text{def}}{=} \{\rho \mid \rho(d_1) = d_1',\ \rho(d_2) = d_2',\ \rho\big(\{d_3\} \cup \textcircled{s_1} \cup \textcircled{s_2}\big) = \textcircled{s'}\}.$$

6.3 HDS-automata

Now we define *named sets with symmetries*: they are similar to named sets (Definition 4.2), 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 HDS-automata.

Definition 6.4 (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 6.5 (HD-automata with Symmetries) *A* HD-automaton with Symmetries *(or* HDS-automaton*) $\mathcal{A}$ 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);*

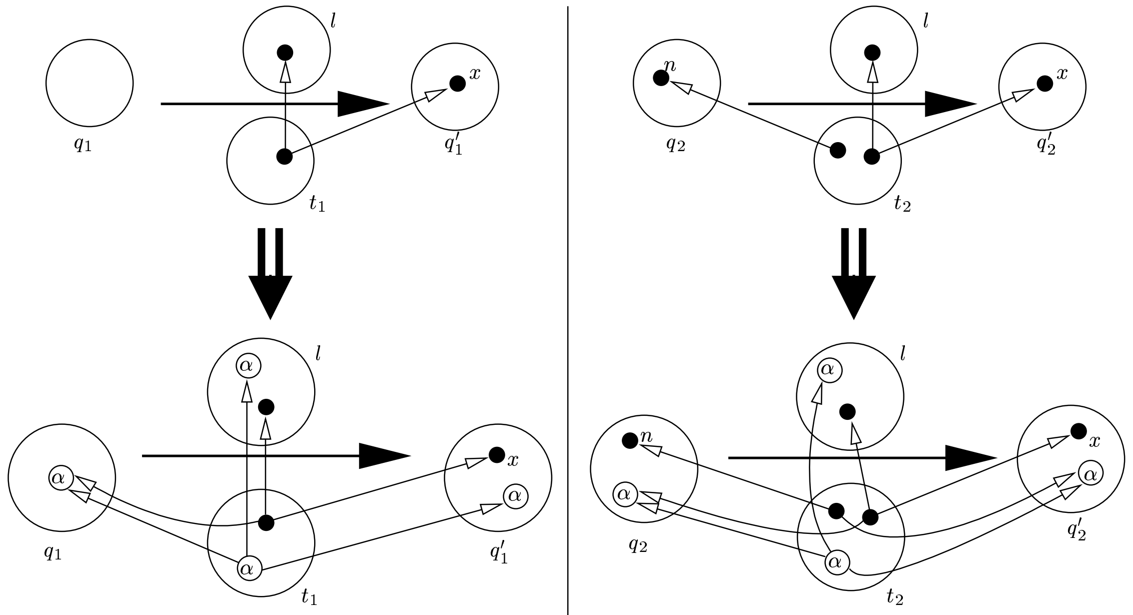

Figure 7: Wrong translation from HD-automata to HDS-automata

- *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 $\mathcal{N}$ 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.

6.3.1 From Basic HD-automata to HDS-automata

Now we show that Basic HD-automata are just a particular subclass of HDS-automata. To this purpose, we show how a HDS-automaton can be obtained from a HD-automaton.

The first idea that comes to mind is to add to all the states, the transitions and the labels of the HD-automaton one infinite repository of names, that represents all the names that do not appear explicitly in the HD-automaton: an example of such translation is represented in Figure 7, where the repository is called $\textcircled{\alpha}$. Unfortunately, this simple translation is not correct. In fact, at the level of HD-automata, the left and right transitions of Figure 7 perform the same action: both of them have label l and generate a new name x in the target state. At the level of the HDS-automata, however, the two transitions have a different meaning: in the left transition, name x can correspond to any name; in the right transition, it can correspond to any name *except* name n of the source state. A HDS-automaton with one infinite repository implements a mechanism for generating fresh names that resembles the fresh-input transitions of π-calculus, while a HD-automaton implements a mechanism that resembles the bound-output transitions, and, as we have discussed in Section 6.1, the two mechanisms are quite different. To simulate the form of name generation that is realized by the HD-automata, two infinite repositories of names are necessary in the HDS-automata: we denote them with $\textcircled{\alpha}$ and $\textcircled{\omega}$. When a fresh name is generated in a transition of the HD-automaton, then a name is taken from the repository $\textcircled{\alpha}$ in the corresponding transition of the HDS-automaton. Instead, when a name is forgotten in a transition of the HD-automaton, then that name is put into repository $\textcircled{\omega}$ in the HDS-automaton. This correct translation is illustrated in Figure 8: here the two transitions match also in the case of the HDS-automata, if we assume that name n of the source state of the right transition corresponds in the left transition to a name in repository $\textcircled{\omega}$, that is, if we assume that the left HDS-automaton has already forgotten name n.

Finally, we assume that the set of names $\mathcal{N}$ associated to the HDS-automata is split in two infinite subsets $\mathcal{N}^{\mathrm{HD}}$ and $\mathcal{N}^{\alpha}$. We assume that $\mathcal{N}^{\mathrm{HD}}$ contains all the global names used in the HD-automata (that is, if σ_0 is the initial embedding of a HD-automaton, then $\mathrm{cod}(\sigma_0) \subseteq \mathcal{N}^{\mathrm{HD}}$). This split in the names of HDS-automata is necessary to guarantee that all the HDS-automata associated to HD-automata agree on the names $\mathcal{N}^{\alpha}$ that are initially reserved for repository $\textcircled{\alpha}$.

Definition 6.6 (from HD-automata to HDS-automata) *The HDS-automaton $\mathcal{B}$ corresponding to HD-automaton $\mathcal{A}$ is defined as follows:*

- $L_{\mathcal{B}} = L_{\mathcal{A}}$ *and, for each* $l \in L_{\mathcal{B}}$, $\mathsf{L}_{\mathcal{B}}[l] = \{\!|n \in \mathsf{L}[l], \textcircled{\alpha}, \textcircled{\omega}|\!\}$;

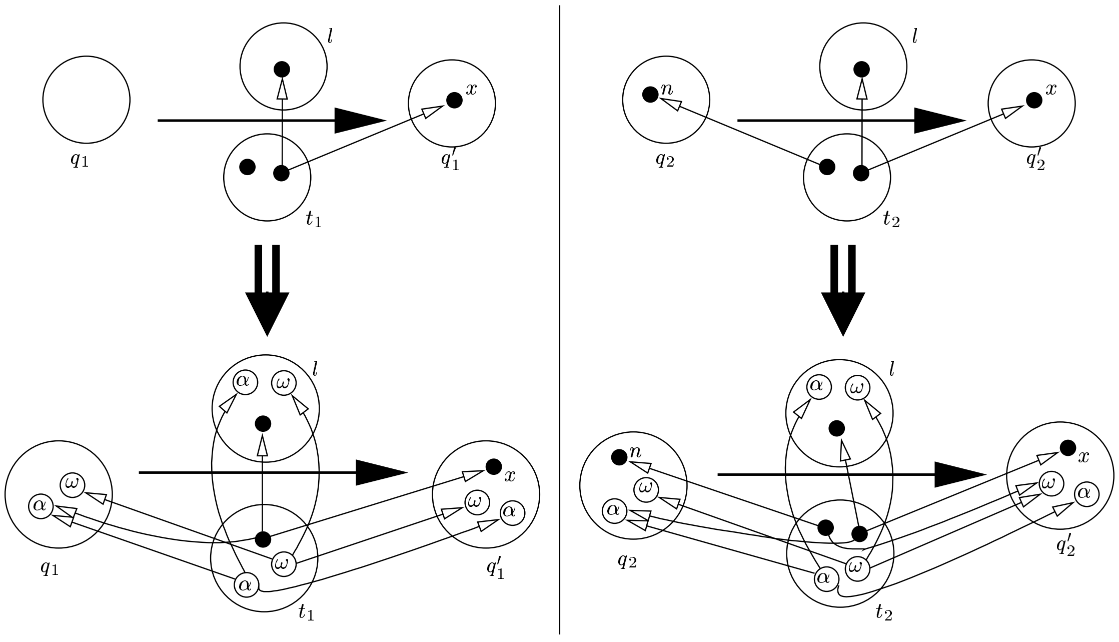

Figure 8: Correct translation from HD-automata to HDS-automata

- *$Q_{\mathcal{B}} = Q_{\mathcal{A}}$ and, for each $q \in Q_{\mathcal{B}}$, $\mathsf{Q}_{\mathcal{B}}[q] = \{\!|n \in \mathsf{Q}[q], \textcircled{\alpha}, \textcircled{\omega}|\!\}$;*
- *$T_{\mathcal{B}} = T_{\mathcal{A}}$ and, for each $t \in T_{\mathcal{B}}$, $\mathsf{T}_{\mathcal{B}}[t] = \{\!|n \in \mathsf{T}[t], \textcircled{\alpha}, \textcircled{\omega}|\!\}$;*
- *$s_{\mathcal{B}} = s_{\mathcal{A}}$, and, if $t \in T_{\mathcal{B}}$ then*

$$\mathsf{s}_{\mathcal{B}}[t] = \left\{\left|\begin{array}{ll} n \mapsto \mathsf{s}_{\mathcal{A}}[t](n) & \textit{if } n \in \mathrm{dom}(\mathsf{s}_{\mathcal{A}}[t]) \\ n \mapsto \textcircled{\alpha} & \textit{otherwise} \\ \textcircled{\alpha} \mapsto \textcircled{\alpha} & \\ \textcircled{\omega} \mapsto \textcircled{\omega} & \end{array}\right|\right\}$$

- *$d_{\mathcal{B}} = d_{\mathcal{A}}$, and, if $t \in T_{\mathcal{B}}$ then*

$$\mathsf{d}_{\mathcal{B}}[t] = \left\{\left|\begin{array}{ll} n \mapsto \mathsf{d}_{\mathcal{A}}[t](n) & \textit{if } n \in \mathrm{dom}(\mathsf{d}_{\mathcal{A}}[t]) \\ n \mapsto \textcircled{\omega} & \textit{otherwise} \\ \textcircled{\alpha} \mapsto \textcircled{\alpha} & \\ \textcircled{\omega} \mapsto \textcircled{\omega} & \end{array}\right|\right\}$$

- *$o_{\mathcal{B}} = o_{\mathcal{A}}$, and, if $t \in T_{\mathcal{B}}$ then*

$$\mathsf{o}_{\mathcal{B}}[t] = \left\{\left|\begin{array}{ll} n \mapsto \mathsf{o}_{\mathcal{A}}[t](n) & \textit{if } n \in \mathrm{dom}(\mathsf{o}_{\mathcal{A}}[t]) \\ n \mapsto \textcircled{\omega} & \textit{otherwise} \\ \textcircled{\alpha} \mapsto \textcircled{\alpha} & \\ \textcircled{\omega} \mapsto \textcircled{\omega} & \end{array}\right|\right\}$$

- *$q_{0\mathcal{B}} = q_{0\mathcal{A}}$, and*

$$f_{0\mathcal{B}} = \left\{\left|\begin{array}{ll} n \mapsto \sigma_{0\mathcal{A}}^{-1}(n) & \textit{if } n \in \mathrm{cod}(\sigma_{0\mathcal{A}}) \\ n \mapsto \textcircled{\omega} & \textit{if } n \in \mathcal{N}^{\mathrm{HD}} \setminus \mathrm{cod}(\sigma_{0\mathcal{A}}) \\ n \mapsto \textcircled{\alpha} & \textit{if } n \in \mathcal{N}^{\alpha} \end{array}\right|\right\}$$

By exploiting this encoding, all the formalisms that we have mapped into Basic HD-automata in Section 4 can be mapped as well into HDS-automata. In the case of the history-preserving semantics of Petri nets, moreover, the possibility of using symmetries in the label is particularly convenient; according to Definition 5.26, in fact, each transition on the net configurations is mapped into a set of transitions of the HD-automaton: these transitions differ for a permutation of the names that represent the past causes. By exploiting the symmetries, a single transition is required, where all the names that represent the past clauses are declared indistinguishable.

$l \in \mathrm{L}^{\pi}$	`tau`	`in`	`in`$_2$	`out`	`out`$_2$	`bout`
$\mathsf{L}^{\pi}[l]$	$\{\!\!\{\textcircled{\nu}, \textcircled{u}\}\!\!\}$	$\left\{\!\!\left\{ \begin{array}{c} n_{\mathrm{sub}}, n_{\mathrm{obj}}, \\ \textcircled{\nu}, \textcircled{u} \end{array} \right\}\!\!\right\}$	$\left\{\!\!\left\{ \begin{array}{c} n, \\ \textcircled{\nu}, \textcircled{u} \end{array} \right\}\!\!\right\}$	$\left\{\!\!\left\{ \begin{array}{c} n_{\mathrm{sub}}, n_{\mathrm{obj}}, \\ \textcircled{\nu}, \textcircled{u} \end{array} \right\}\!\!\right\}$	$\left\{\!\!\left\{ \begin{array}{c} n, \\ \textcircled{\nu}, \textcircled{u} \end{array} \right\}\!\!\right\}$	$\left\{\!\!\left\{ \begin{array}{c} n_{\mathrm{sub}}, n_{\mathrm{obj}}, \\ \textcircled{\nu}, \textcircled{u} \end{array} \right\}\!\!\right\}$

Table 7: The named set of labels for early π-calculus

6.3.2 From early π-calculus to HDS-automata

Now we map the *early* semantics of π-calculus agents (see Section 3.2) into HDS-automata. As we have already discussed in the introduction of this chapter, two infinite repositories of names are used in this case. The first one, that we denote with $\textcircled{\nu}$, is used for the names that are generated by the agent in the bound output transitions. The other one, denoted with $\textcircled{u}$, is used for the names that exist in the environment but that are not known by the agent; fresh names are taken from here in the case of input transitions, and names that are forgotten in the transitions of the agent are collected in this infinite repository.

Now we define the named set with symmetries L^{π_e} of the labels for the π-calculus (see also Table 7). Differently from the labels defined in Section 4.1 for the ground semantics, here we have free input actions `in` and `in`$_2$ instead of bound input action `bin`.

$$\mathrm{L}^{\pi_e} = \{\mathtt{tau}, \mathtt{in}, \mathtt{in}_2, \mathtt{out}, \mathtt{out}_2, \mathtt{bout}\}.$$

No distinct name is associated to `tau`, one distinct name (n) is associated to `in`$_2$ and `out`$_2$, and two distinct names (n_{sub} and n_{obj}) are associated to `in`, `out` and `bout`. In all these labels, then, there are the two infinite repositories $\textcircled{\nu}$ and $\textcircled{u}$. Labels `in`$_2$ and `out`$_2$ are used when subject and object names of inputs or free outputs coincide.

As we have done in Section 4 for the ground semantics of π-calculus, we assume to have a function norm that selects the representative of each class of agents differing for a renaming. Only *representative transitions* are used when building a HDS-automaton from a π-calculus agent. Representative transitions are defined as for the ground semantics (Definition 4.4). According to that definition, a single transition is taken from a bunch of bound outputs that differ only for the extruded name, and from a bunch of input transitions that differ in the fresh name that is received from the environment. In the case of the input transitions, all the names that appear in the source state have to be considered as other possible input values.

Finally, we assume that the set of names $\mathcal{N}$ is split in two infinite subsets $\mathcal{N}^{\nu}$ and $\mathcal{N}^{u}$. Names $\mathcal{N}^{\nu}$ are reserved: they are used when a name is created during the bound output transition. Hence, we assume that, initially, all the free names of π-calculus agents are in $\mathcal{N}^{u}$.

Definition 6.7 (from π-calculus agents to HDS-automata) *Let p be a π-calculus agent with $\mathrm{fn}(p) \subseteq \mathcal{N}^{u}$.*
The HDS-automaton $\mathcal{A}_p^{\pi_e}$ corresponding to p is the smallest HDS-automaton that respects the following rules:

- *if $\mathrm{norm}(p) = (q, \sigma)$ then:*
  - *$q \in \mathrm{Q}$ is the initial state and $\mathsf{Q}[q] = \{\!\!\{n \in \mathrm{fn}(q), \textcircled{\nu}, \textcircled{u}\}\!\!\}$;*
  - *the initial embedding is defined as follows:*

$$f_0 = \left\{\!\!\left\{ \begin{array}{ll} n \mapsto \sigma(n) & \text{if } n \in \mathrm{fn}(p) \\ n \mapsto \textcircled{u} & \text{if } n \in \mathcal{N}^{u} \setminus \mathrm{fn}(p) \\ n \mapsto \textcircled{\nu} & \text{if } n \in \mathcal{N}^{\nu} \end{array} \right\}\!\!\right\}$$

- *if $q \in \mathrm{Q}$, $q \xrightarrow{\mu} q'$ is a representative transition and $\mathrm{norm}(q') = (q'', \sigma)$, then:*
  - *$q'' \in \mathrm{Q}$ and $\mathsf{Q}[q''] = \{\!\!\{n \in \mathrm{fn}(q''), \textcircled{\nu}, \textcircled{u}\}\!\!\}$;*
  - *there is some $t \in \mathrm{T}$ such that $\mathsf{T}[t] = \left\{\!\!\left\{n \in \big(\mathrm{fn}(q) \cup \mathrm{n}(\mu)\big), \textcircled{\nu}, \textcircled{u}\right\}\!\!\right\}$; moreover*

μ	τ	xy		xx	$\bar{x}y$		$\bar{x}x$	$\bar{x}(y)$	
$l \in \mathrm{L}^{\pi}$	`tau`	`in`		`in`$_2$	`out`		`out`$_2$	`bout`	
$\square = \lambda(\diamond) \in \mathrm{n}(\mu)$	/	x	y	x	x	y	x	x	y
$\diamond = \kappa(\square) \in \mathsf{L}^{\pi}[l]$	/	n_{sub}	n_{obj}	n	n_{sub}	n_{obj}	n	n_{sub}	n_{obj}

Table 8: Relations between early π-calculus labels and labels of HDS-automata

– $\mathrm{s}(t) = q$, $\mathrm{d}(t) = q''$,

$$\mathsf{s}[t] = \left\{ \begin{array}{ll} n \mapsto n & \textit{if } n \in \mathrm{fn}(q) \\ n \mapsto \textcircled{ν} & \textit{if } \mu = \overline{x}(y) \textit{ and } n = y \\ n \mapsto \textcircled{u} & \textit{if } \mu = xy,\, y \notin \mathrm{fn}(q) \textit{ and } n = y \\ \textcircled{ν} \mapsto \textcircled{ν} & \\ \textcircled{u} \mapsto \textcircled{u} & \end{array} \right\}$$

$$\mathsf{d}[t] = \left\{ \begin{array}{ll} n \mapsto \sigma(n) & \textit{if } n \in \mathrm{fn}(q') \\ n \mapsto \textcircled{u} & \textit{if } n \in (\mathrm{fn}(q) \cup \mathrm{n}(\mu)) \setminus \mathrm{fn}(q') \\ \textcircled{ν} \mapsto \textcircled{ν} & \\ \textcircled{u} \mapsto \textcircled{u} & \end{array} \right\}$$

– $\mathrm{o}(t) = l$ *and*

$$\mathsf{o}[t] = \left\{ \begin{array}{ll} n \mapsto \kappa(n) & \textit{if } n \in \mathrm{n}(\mu) \\ n \mapsto \textcircled{u} & \textit{if } n \in \mathrm{fn}(q) \setminus \mathrm{n}(\mu) \\ \textcircled{ν} \mapsto \textcircled{ν} & \\ \textcircled{u} \mapsto \textcircled{u} & \end{array} \right\}$$

where l and κ are related to μ as defined in Table 8.

Definition 6.7 gives rise in general to infinite HDS-automata. However, there are classes of π-calculus agents that generate finite HDS-automata. This is case of *finitary* π-calculus agents (Definition 4.7).

Theorem 6.8 *Let p be a finitary π-calculus agent. Then the HD-automaton $\mathcal{A}_p^{\pi_e}$ is finite.*

Proof. The proof of Theorem 4.8 can be reused, with small changes, also in this case. □

We recall that it is not decidable whether an agent is finitary; however, there are subclass of finitary agents which can be characterized syntactically. The most well-known is the class of the agents with *finite control*.

6.4 Bisimulation on HDS-automata

Now we introduce bisimulation on HDS-automata and describe some of its basic properties. We also show that early bisimulation on π-calculus agents is captured by bisimulation on HDS-automata.

Similarly to what happens for HD-bisimulation (Section 4.2), also a HDS-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 correspondence between the names of the states; in this case, however, the correspondence between the names in total, i.e., $\delta : \mathcal{N} \longleftrightarrow \mathcal{N}$ is a total bijection between the names of q_1 and those of q_2.

Let us consider the HDS-automata in Figure 9. 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 HDS-automata, proviso the target states are bisimilar according to the correspondences 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 HDS-automaton. 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. This is obtained in the definition of HDS-bisimulation by requiring that, given transition $t_1 : q_1 \xrightarrow{l} q_2$, for each bijection[7] $\alpha_1 \in \mathsf{s}_1[t_1]$ there exist a transition t_2 from q_2 and a bijection $\alpha_2 \in \mathsf{s}_2[t_2]$ so

[7]It is worth remind that, according to Definition 6.3, all the permutations in the symmetry of q_1 are reflected in the bijections that form the embedding $\mathsf{s}_1[t_1]$.

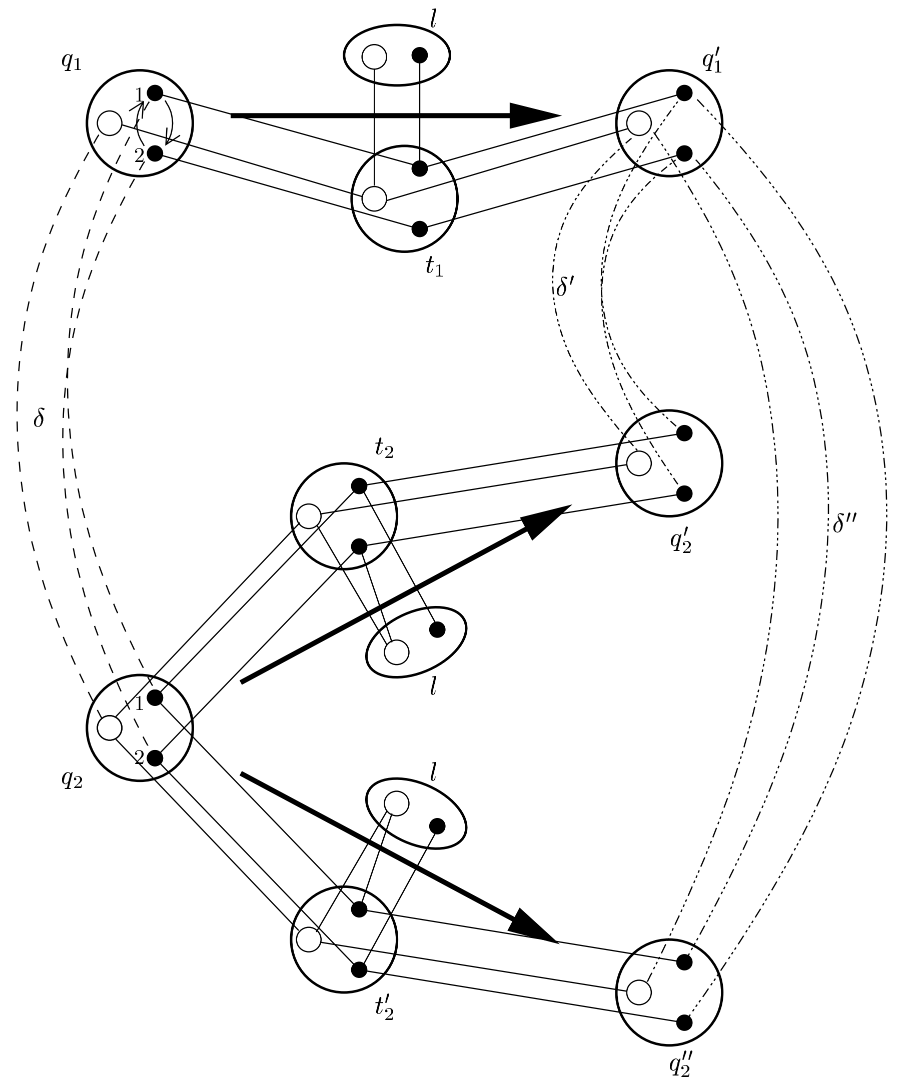

Figure 9: A step of bisimulation on HDS-automata

that t_1 and t_2 match w.r.t. bijections α_1 and α_2. This behavior works correctly also in the case of the input transitions in the HDS-automata obtained from π-calculus agents. In this case, in fact, if state q_1 has a distinct name n that does not appear in state q_2, this name is mapped by δ in repository $\textcircled{u}$ of state q_2. The specific transition of q_1 that corresponds to the reception of name n is correctly matched in q_2 by the "default" transition that corresponds to the reception of a fresh name taken from $\textcircled{u}$. 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 6.9 (HDS-bisimulation) *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two HDS-automata. A* HDS-simulation *for $\mathcal{A}_1$ and $\mathcal{A}_2$ is a set of triples $\mathcal{R} \subseteq \{\langle q_1, \delta, q_2\rangle \mid q_1 \in Q_1,\ q_2 \in Q_2,\ \delta : \mathcal{N} \longleftrightarrow \mathcal{N}\}$ such that, whenever $\langle q_1, \delta, q_2\rangle \in \mathcal{R}$ then:*

for each transition $t_1 : q_1 \xrightarrow{l} q_1'$ in $\mathcal{A}_1$ and bijections $\alpha_1 \in \mathsf{s}_1[t_1]$, $\gamma_1 \in \mathsf{o}_1[t_1]$ and $\beta_1 \in \mathsf{d}_1[t_1]$, there exist some transition $t_2 : q_2 \xrightarrow{l} q_2'$ in $\mathcal{A}_2$ and bijections $\alpha_2 \in \mathsf{s}_2[t_2]$, $\gamma_2 \in \mathsf{o}_2[t_2]$ and $\beta_2 \in \mathsf{d}_2[t_2]$, such that $\alpha_1;\delta;\alpha_2^{-1} = \gamma_1;\gamma_2^{-1} = \beta_1;\delta';\beta_2^{-1}$ and $\langle q_1'\delta', q_2'\rangle \in \mathcal{R}$. $(*)$

A HDS-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 HDS-bisimulation for $\mathcal{A}$ is a HDS-bisimulation for $\mathcal{A}$ and $\mathcal{A}$.
The HDS-automata $\mathcal{A}_1$ 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 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 $\rho = \alpha_1;\delta;\alpha_2^{-1}$.

6.4.1 Some basic properties of HDS-bisimulation

In this section we present some basic properties of HDS-bisimulations. First of all, similarly to HD-bisimulations (Subsection 4.2.1), also HDS-bisimulations are closes w.r.t. union and concatenation.

Proposition 6.10 *Let $\{\mathcal{R}_i \mid i \in I\}$ be a (finite or infinite) set of HDS-bisimulations for $\mathcal{A}_1$ and $\mathcal{A}_2$. Then $\bigcup_{i\in I} \mathcal{R}_i$ is a HDS-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$.*

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

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

As a consequence, greatest HDS-bisimulations exist: we denote with $\mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2}$ the greatest HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, and with $\mathcal{R}_{\mathcal{A}}$ the greatest HD-bisimulation for $\mathcal{A}$. Moreover, relation $\sim$ is an equivalence on HDS-automata.

In the case of HDS-bisimulation, operator $\widehat{-}$ has no meaning: the name correspondence in a triple of a HDS-bisimulation is a total bijection, and it is not possible to extend it to new pairs of names. In the case of HDS-bisimulations, however, a new operator can be defined, that closes a bisimulation w.r.t. all the symmetries that are present in the states of the HDS-automata.

Proposition 6.11 *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:*

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

Proof. We prove that $\widetilde{\mathcal{R}}$ is a HDS-simulation whenever $\mathcal{R}$ is a HDS-simulation.
Assume $\langle q_1, \delta', q_2\rangle \in \widetilde{\mathcal{R}}$. Then, by definition of $\widetilde{\mathcal{R}}$, $\langle q_1, \delta, q_2\rangle \in \mathcal{R}$ with

$$\delta' = \sigma_1;\delta;\sigma_2 \tag{25}$$

for some $\sigma_1 \in \mathsf{Q}_1[q_1]$ and $\sigma_2 \in \mathsf{Q}_2[q_2]$.
Let $t_1 : q_1 \xrightarrow{l} q_1'$ and $\alpha_1' \in \mathsf{s}_1[t_1]$, $\gamma_1 \in \mathsf{o}_1[t_1]$, $\beta_1 \in \mathsf{d}_1[t_1]$.
Now, let

$$\alpha_1 \stackrel{\text{def}}{=} \alpha_1';\sigma_1. \tag{26}$$

Then, according to definition of embedding on symmetries, $\alpha_1 \in \mathsf{s}_1[t_1]$.

Since $\langle q_1, \delta, q_2 \rangle \in \mathcal{R}$ and $\mathcal{R}$ is a HDS-bisimulation, then there exist $t_2 : q_2 \xrightarrow{l} q'_2$ and $\alpha_2 \in \mathsf{s}_2[t_2]$, $\gamma_2 \in \mathsf{o}_2[t_2]$, $\beta_2 \in \mathsf{d}_2[t_2]$ such that

$$\alpha_1; \delta; \alpha_2^{-1} = \gamma_1; \gamma_2^{-1} = \beta_1; \delta''; \beta_2^{-1}$$

and $\langle q'_1, \delta'', q'_2 \rangle \in \mathcal{R}$.
Moreover, let

$$\alpha'_2 \stackrel{\text{def}}{=} \alpha_2; \sigma_2. \tag{27}$$

Then, applying equations (26), (25) and (27) to $\alpha_1; \delta; \alpha_2^{-1} = \gamma_1; \gamma_2^{-1}$, we obtain

$$\alpha'_1; \delta'; {\alpha'_2}^{-1} = \gamma_1; \gamma_2^{-1}.$$

Moreover, $\mathcal{R} \subseteq \widetilde{\mathcal{R}}$, so $\langle q'_1, \delta'', q'_2 \rangle \in \mathcal{R}$ implies $\langle q'_1, \delta'', q'_2 \rangle \in \widetilde{\mathcal{R}}$. This proves that t_2 is a good matching transition for t_1 also for relation $\widetilde{\mathcal{R}}$.
This concludes the proof that $\widetilde{\mathcal{R}}$ is a HDS-simulation whenever $\mathcal{R}$ is a HDS-simulation. To prove that $\widetilde{\mathcal{R}}$ is a HDS-bisimulation whenever $\mathcal{R}$ is a HDS-bisimulation it is sufficient to notice that $\left(\widetilde{\mathcal{R}}\right)^{-1} = \widetilde{\mathcal{R}^{-1}}$. $\square$

6.4.2 Global states and global bisimulation

Now we define global bisimulation on HDS-automata; the intuition is similar to the one applied in Subsection 4.2.2 for HD-automata: we enrich states and transitions with a mapping from their local names into the global names.

Definition 6.12 (global states and global transitions) *A* global state *of a HDS-automaton $\mathcal{A}$ is a pair $g = (q, \sigma)$, where $q \in Q$ and $\sigma : \mathcal{N} \longleftrightarrow \mathcal{N}$. The set of all the global states of $\mathcal{A}$ is denoted by G.*
A global transition *is a pair $u = (t, \rho)$, where $t \in T$ and $\rho : \mathcal{N} \longleftrightarrow \mathcal{N}$. The set of all the global transitions of $\mathcal{A}$ is denoted by U.*
If $t : q \xrightarrow{l} q'$ then we write $(t, \rho) : (q, \sigma) \xrightarrow{(l,\lambda)} (q', \sigma')$, where $\sigma = \alpha^{-1}; \rho$, and $\lambda = \gamma^{-1}; \rho$, and $\sigma' = \beta^{-1}; \rho$ for some $\alpha \in \mathsf{s}[t]$, $\gamma \in \mathsf{o}[t]$ and $\beta \in \mathsf{d}[t]$.

We remark that this definition is different in many aspects from the definition of global states and global transitions on Basic HD-automata (Definition 4.17). First of all, global states and global transitions are not named sets with symmetries: it is meaningless to define symmetries on global names, since this is contrary to the intuition that global names, differently from local names, have a precise identity. Global states and global transitions can be seen better as a convenient notation: this interpretation is enforced by the fact that different repositories, labels and targets can be assigned to the same global transition (t, ρ): in fact, it holds that $(t, \rho) : (q, \sigma_1) \xrightarrow{(l,\lambda_1)} (q', \sigma'_1)$ and $(t, \rho) : (q, \sigma_2) \xrightarrow{(l,\lambda_2)} (q', \sigma'_2)$ whenever σ_1 and σ_2 differ for a permutation in the symmetry of q (and similarly for the labels and the targets).
Now we give the definition of global bisimulation on HDS-automata.

Definition 6.13 (global bisimulation) *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two HDS-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 there exist some $u_2 : g_2 \xrightarrow{k} g'_2$ in U_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 $\mathcal{R}$ 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 HDS-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 $(q_{01}, \sigma_{01}) \; \mathcal{R} \; (q_{02}, \sigma_{02})$ for each $\sigma_{01} \in f_{01}$ and $\sigma_{02} \in f_{02}$.*

In the definition of global bisimulation for Basic HD-automata, clause "$\mathsf{U}_1[u_1]_{\text{new}} \cap \mathsf{G}_2[g_2] = \emptyset$" was added to guarantee the freshness for of the new names introduced in a transition. This clause is not necessary in the case of HDS-automata, since the mechanism of infinite repositories is sufficient to guarantee the correspondence of the new names.

Proposition 6.14 *Two HDS-automata are HDS-bisimilar iff they are global bisimilar.*

Proof. We show that if $\mathcal{R}$ is a HDS-bisimulation then

$$\mathcal{R}' \stackrel{\text{def}}{=} \{\langle (q_1, \sigma_2), (q_2, \sigma_2) \rangle \mid \langle q_1; (\sigma_1; \sigma_2^{-1}), q_2 \rangle \in \mathcal{R}\}$$

is a global bisimulation, and, conversely, that if $\mathcal{R}'$ is a global bisimulation then

$$\mathcal{R} \stackrel{\text{def}}{=} \{\langle q_1; (\sigma_1; \sigma_2^{-1}), q_2\rangle \mid (q_1, \sigma_2)\ \mathcal{R}'\ (q_2, \sigma_2)\}$$

is a HDS-bisimulation.
We proof only the first implication: the second can be recovered easily by reverting the steps of the proof.
Assume that $(q_1, \sigma_1)\ \mathcal{R}'\ (q_2, \sigma_2)$. By definition of global bisimulation, we want to prove that for each $(t_1, \rho_1) : (q_1, \sigma_1) \xrightarrow{(l,\lambda)} (q_1', \sigma_1')$ there is some $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(l,\lambda)} (q_2', \sigma_2')$ such that $(q_1', \sigma_1')\ \mathcal{R}'\ (q_2', \sigma_2')$.
By definition of global transitions this corresponds to prove that for each $t_1 : q_1 \xrightarrow{l} q_1'$, $\alpha_1 \in \mathsf{s}_1[t_1]$, $\beta_1 \in \mathsf{d}_1[t_1]$, $\gamma_1 \in \mathsf{o}_1[t_1]$ there exist $t_2 : q_2 \xrightarrow{l} q_2'$, $\alpha_2 \in \mathsf{s}_2[t_2]$, $\beta_2 \in \mathsf{d}_2[t_2]$, $\gamma_2 \in \mathsf{o}_2[t_2]$ such that

$$\gamma_1^{-1}; \alpha_1; \sigma_1 = \lambda = \gamma_2^{-1}; \alpha_2; \sigma_2 \tag{28}$$

and $(q_1', \sigma_1')\ \mathcal{R}'\ (q_2', \sigma_2')$ with

$$\sigma_1' = \beta_1^{-1}; \alpha_1; \sigma_1 \qquad \text{and} \qquad \sigma_2' = \beta_2^{-1}; \alpha_2; \sigma_2. \tag{29}$$

Now, (28) and (29) are equivalent to require that

$$\alpha_1; \delta; \alpha_2^{-1} = \gamma_1; \gamma_2^{-1} = \beta_1; \delta'; \beta_2^{-1}$$

for $\delta \stackrel{\text{def}}{=} \sigma_1; \sigma_2^{-1}$ and $\delta' \stackrel{\text{def}}{=} \sigma_1'; {\sigma_2'}^{-1}$.
That is, it is equivalent to require that transitions t_1 and t_2 match according to HDS-bisimulation $\mathcal{R}$. □

6.4.3 Relating HD-bisimulation and HDS-bisimulation

In Subsection 6.3.1 we have seen a mapping from HD-automata to HDS-automata. Here we show that, according to this map, HDS-bisimilar HDS-automata are associated to, and only to, HD-bisimilar HD-automata.

Theorem 6.15 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two Basic HD-automata, and let $\mathcal{B}_1$ and $\mathcal{B}_2$ be the corresponding HDS-automata, according to Definition 6.6. Then $\mathcal{A}_1 \sim \mathcal{A}_2$ if and only if $\mathcal{B}_1 \sim \mathcal{B}_2$.*

Proof (Sketch). If $\mathcal{R}$ is the greatest HD-bisimulation between $\mathcal{A}_1$ and $\mathcal{A}_2$, then $\mathcal{R}'$ is a HDS-bisimulation between $\mathcal{B}_1$ and $\mathcal{B}_2$, where

$$\mathcal{R}' \stackrel{\text{def}}{=} \{\langle q_1, \delta', q_2\rangle \mid \langle q_1, \delta, q_2\rangle \in \mathcal{R} \wedge \delta' : \mathcal{N} \longleftrightarrow \mathcal{N} \wedge \delta' \text{ extends } \delta\}$$

and where δ' extends δ if the following conditions hold:

- $\delta \subseteq \delta'$ (i.e., all the name correspondences in δ are also in δ'),
- $n \in \textcircled{\alpha}$ if and only if $\delta'(n) \in \textcircled{\alpha}$ (i.e., the names in sinks $\textcircled{\alpha}$ in q_1 and q_2 correspond through δ'),
- if $n \in \mathsf{Q}_{\mathcal{A}_1}[q_1] \setminus \mathrm{dom}(\delta)$ then $\delta'(n) \in \textcircled{\omega} \cup (\mathsf{Q}_{\mathcal{A}_2}[q_2] \setminus \mathrm{cod}(\delta))$ and if $\delta'(n) \in \mathsf{Q}_{\mathcal{A}_2}[q_2] \setminus \mathrm{cod}(\delta)$ then $n \in \textcircled{\omega} \cup (\mathsf{Q}_{\mathcal{A}_1}[q_1] \setminus \mathrm{dom}(\delta))$ (i.e., all the distinct names of q_1 that are not in relation δ are mapped into names of sink $\textcircled{\omega}$, and vice-versa).

For the converse, if $\mathcal{R}'$ is the greatest HDS-bisimulation between $\mathcal{B}_1$ and $\mathcal{B}_2$, then $\mathcal{R}$ is a HD-bisimulation between $\mathcal{A}_1$ and $\mathcal{A}_2$, where:

$$\mathcal{R} \stackrel{\text{def}}{=} \{\langle q_1, \delta, q_2\rangle \mid \langle q_1, \delta', q_2\rangle \in \mathcal{R}' \ \wedge\ \delta\ =\ \delta' \cap (\mathsf{Q}_{\mathcal{A}_1}[q_1] \times \mathsf{Q}_{\mathcal{A}_2}[q_2])\}.$$

□

6.4.4 Relating π-calculus early bisimulation and HDS-bisimulation

Now we show that two π-calculus agents are bisimilar, according to the early semantics, if and only if the corresponding HDS-automata are bisimilar.

Theorem 6.16 *Let $\mathcal{A}_1^{\pi_e}$ and $\mathcal{A}_2^{\pi_e}$ be the HDS-automata that correspond to π-calculus agents p_1 and p_2, according the* early *semantics. Then $p_1 \sim_e p_2$ iff $\mathcal{A}_1^{\pi_e} \sim \mathcal{A}_2^{\pi_e}$.*

Proof (Sketch). For the "if" implication, we show that, if $\mathcal{R}$ is a HDS-bisimulation for $\mathcal{A}_1^{\pi_e}$ and $\mathcal{A}_2^{\pi_e}$, then

$$\mathcal{R}' = \{\langle p_1\sigma_1, p_2\sigma_2\rangle \mid \sigma_1, \sigma_2 : \mathcal{N} \longleftrightarrow \mathcal{N},\ \langle p_1, (\sigma_1; \sigma_2^{-1}), p_2\rangle \in \mathcal{R}\}$$

is a π-calculus early bisimulation.
Assume $q_1\ \mathcal{R}'\ q_2$. Then $\langle p_1, (\sigma_1; \sigma_2^{-1}), p_2\rangle \in \mathcal{R}$ and $q_1 = p_1\sigma_1$, $q_2 = p_2\sigma_2$.
Assume $q_1 \xrightarrow{\mu} q_1'$ with $\mathrm{bn}(\mu) \cap \mathrm{fn}(q_1|q_2) = \emptyset$. Then we want to prove that $q_2 \xrightarrow{\mu} q_2'$ and $q_1'\ \mathcal{R}'\ q_2'$.
The proof proceeds by cases, considering the different kinds of π-calculus transitions. Here we consider only the input transitions $q_1 \xrightarrow{xy} q_1'$.
We distinguish five sub-cases:

1. $x \neq y, y \in \mathrm{fn}(q_1)$ and $y \in \mathrm{fn}(q_2)$;
2. $x \neq y, y \in \mathrm{fn}(q_1)$ and $y \notin \mathrm{fn}(q_2)$;
3. $x \neq y, y \notin \mathrm{fn}(q_1)$ and $y \in \mathrm{fn}(q_2)$;
4. $x \neq y, y \notin \mathrm{fn}(q_1)$ and $y \notin \mathrm{fn}(q_2)$;
5. $x = y$ (hence $y \in \mathrm{fn}(q_1)$ and $y \in \mathrm{fn}(q_2)$).

Let us consider sub-case 1.
Since σ_1 is a bijection on $\mathcal{N}$, $q_1 \xrightarrow{xy} q_1'$ implies $p_1 \xrightarrow{ab} p_1'$ with $x = \sigma_1(a)$, $y = \sigma_1(b)$ and

$$q_1' = p_1'\sigma_1. \tag{30}$$

Clearly, $b \in \mathrm{fn}(p_1)$, so $p_1 \xrightarrow{ab} p_1'$ is a representative transition. By Definition 6.7, $t_1 : p_1 \xrightarrow{\mathtt{in}} p_1''$ is a transition of $\mathcal{A}_1^{\pi_e}$, where $\mathrm{norm}(p_1') = (p_1'', \sigma_1'')$.
Let us fix any $\alpha_1 \in \mathsf{s}_1[t_1]$, $\gamma_1 \in \mathsf{o}_1[t_1]$ and $\beta_1 \in \mathsf{d}_1[t_1]$.
Then there are some $t_2 : p_2 \xrightarrow{\mathtt{in}} p_2''$, $\alpha_2 \in \mathsf{s}_2[t_2]$, $\gamma_2 \in \mathsf{o}_2[t_2]$, and $\beta_2 \in \mathsf{d}_2[t_2]$, such that

$$\alpha_1; (\sigma_1; \sigma_2^{-1}); \alpha_2^{-1} = \gamma_1; \gamma_2^{-1} = \beta_1; \delta'; \beta_2^{-1} \tag{31}$$

and $\langle p_1'', \delta', p_2'' \rangle \in \mathcal{R}$ for some δ'.
Now, since $t_2 : p_2 \xrightarrow{\mathtt{in}} p_2''$, there must be some representative transition $p_2 \xrightarrow{cd} p_2'$ such that $(p_2'', \sigma_2'') = \mathrm{norm}(p_2')$.
Then $q_2 \xrightarrow{wz} q_2'$, where $w = \sigma_2(c)$, $z = \sigma_2(d)$ and

$$q_2' = p_2'\sigma_2. \tag{32}$$

Now we show that $w = x$. In fact, $\gamma_1(a) = n_{\mathrm{sub}} = \gamma_2(c)$. By applying (31) this implies $\sigma_1(\alpha_1(a)) = \sigma_2(\alpha_2(c))$. However, since $a \in \mathrm{fn}(p_1)$, by constriction $\alpha_1(a) = a$. Similarly, $\alpha_2(c) = c$. So, $x = \sigma_1(a) = \sigma_2(c) = w$.
Analogously, $y = z$ holds.
It remains to show that $q_1' \ \mathcal{R}' \ q_2'$. To this purpose, we show that

$$q_1' = p_1''\sigma_1' \qquad \text{and} \qquad q_2' = p_2''\sigma_2' \tag{33}$$

for some σ_1' and σ_2' such that $\sigma_1'; \sigma_2'^{-1} = \delta'$. Then, $q_1' \ \mathcal{R}' \ q_2'$ derives. by definition of $\mathcal{R}'$, from $\langle p_1'', \delta', p_2'' \rangle \in \mathcal{R}$.
Let $\sigma_1' \stackrel{\mathrm{def}}{=} \sigma_1''^{-1}; \sigma_1$ and $\sigma_2' \stackrel{\mathrm{def}}{=} \delta'^{-1}; \sigma_1'$. Then, clearly, $\sigma_1'; \sigma_2'^{-1} = \delta'$.
Now we show that (33) holds. First of all,

$$\begin{aligned} q_1' &= p_1' && \text{by (30)} \\ &= p_1''\sigma_1''^{-1}\sigma_1 && \text{since } \mathrm{norm}(p_1') = (p_1'', \sigma_1''), \end{aligned}$$

which proves the first equivalence of (30). For the second equivalence, we observe that

$$\begin{aligned} q_2' &= p_2'\sigma_2 && \text{by (32)} \\ &= p_2''\sigma_2''^{-1}\sigma_2 && \text{since } \mathrm{norm}(p_2') = (p_2'', \sigma_2''). \end{aligned}$$

It is hence sufficient to prove that $\sigma_2'(n) = \sigma_2(\sigma_2''^{-1}(n))$ for each $n \in \mathrm{fn}(p_2'')$.
By definition of σ_1' and of σ_2', this amounts to require that

$$\sigma_1(\sigma_1''^{-1}(\delta'^{-1}(n))) = \sigma_2(\sigma_2''^{-1}(n)). \tag{34}$$

Let $m \stackrel{\mathrm{def}}{=} \sigma_1''^{-1}(\delta'^{-1}(n))$. Then we can rewrite (34) as follows:

$$\sigma_2^{-1}(\sigma_1(m)) = \sigma_2''^{-1}(\delta'(\sigma_1''(m))). \tag{35}$$

This is a consequence of $\alpha_1; (\sigma_1; \sigma_2^{-1}); \alpha_2^{-1} = \beta_1; \delta'; \beta_2^{-1}$, since by construction $\mathrm{id}_{\mathrm{fn}(p_1')} \subseteq \alpha_1$, $\sigma_1'' \subseteq \beta_1$, and similarly for α_2 and β_2.
This completes the proof of sub-case 1. The other cases of the input transitions, as well as the other kinds of transitions, follow similar patterns.
The proof of the "only if" implication is similar. □

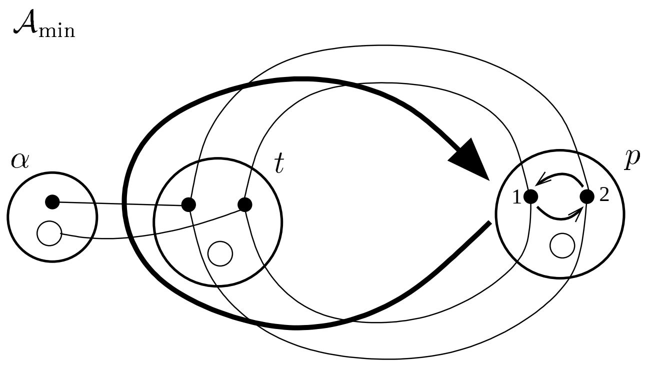

Figure 10: A minimal HDS-automaton for the HD-automata of Figure 4.

6.5 Minimizing HDS-automata

In this section we show that, given a HDS-automaton $\mathcal{A}$, it is possible to minimize it, i.e., to define a HDS-automaton $\mathcal{A}_{\min}$ that is bisimilar to $\mathcal{A}$ and that is "minimal" in the class of HDS-automata bisimilar to $\mathcal{A}$ — we define below what is the meaning of "minimal".

We start by showing that the counter-example presented in Section 4.3 on the existence of minimal HD-automata does not apply to the case of HD-automata with Symmetries. The minimal HDS-automaton corresponding to the two HD-automata in Figure 4 is represented in Figure 10. HDS-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. HDS-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 HDS-automaton. 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:

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

The following proposition shows that: *1.* $\Delta_{\mathcal{A}}(q,q)$ is a symmetry on $\mathcal{N}$ and $\mathsf{Q}[q]$ is a subset of $\Delta_{\mathcal{A}}(q,q)$; that *2.* $\Delta_{\mathcal{A}}(p,q)$ can be recovered, starting from any of its elements, say δ, by composing δ with all the elements of $\Delta_{\mathcal{A}}(q,q)$; and that *3.* if two state p and q are equivalent, then $\Delta_{\mathcal{A}}(p,p)$ are $\Delta_{\mathcal{A}}(q,q)$ are isomorphic.

Proposition 6.17 *Let $\mathcal{A}$ be a HD-automaton. Then:*

1. *$\Delta_{\mathcal{A}}(q,q)$ is symmetry on $\mathcal{N}$ such that $\mathsf{Q}[q] \subseteq \Delta_{\mathcal{A}}(q,q)$;*
2. *if $\delta \in \Delta_{\mathcal{A}}(p,q)$ then $\Delta_{\mathcal{A}}(p,q) = \{\delta;\delta' \mid \delta' \in \Delta_{\mathcal{A}}(q,q)\} = \{\delta';\delta \mid \delta' \in \Delta_{\mathcal{A}}(p,p)\}$;*
3. *if $p \equiv_{\mathcal{A}} q$ and $\delta \in \Delta_{\mathcal{A}}(p,q)$, then $\Delta_{\mathcal{A}}(q,q) = \{\delta^{-1};\rho;\delta \mid \rho \in \Delta_{\mathcal{A}}(p,p)\}$.*

Proof.

1. The proof that $\Delta_{\mathcal{A}}(q,q)$ is a symmetry is easy: clearly $\mathrm{id}_{\mathcal{N}} \in \Delta_{\mathcal{A}}(q,q)$; moreover, if $\delta \in \Delta_{\mathcal{A}}(q,q)$ and $\delta' \in \Delta_{\mathcal{A}}(q,q)$ then $\delta;\delta' \in \Delta_{\mathcal{A}}(q,q)$ by Proposition 6.10. Finally, if $\delta \in \Delta_{\mathcal{A}}(q,q)$ then $\delta^{-1} \in \Delta_{\mathcal{A}}(q,q)$ since $\mathcal{R}_{\mathcal{A}}$ is symmetric.
   Assume $\delta \in \mathsf{Q}[q]$. Since $\mathrm{id}_{\mathcal{N}} \in \Delta_{\mathcal{A}}(q,q)$ and $\mathrm{id}_{\mathcal{N}} \in \mathsf{Q}[q]$, then by Proposition 6.11 $\delta = \delta;\mathrm{id}_{\mathcal{N}};\mathrm{id}_{\mathcal{N}} \in \Delta_{\mathcal{A}}(q,q)$.

2. Let $\delta \in \Delta_{\mathcal{A}}(p,q)$.
   By Proposition 6.10, if $\delta' \in \Delta_{\mathcal{A}}(q,q)$ then $\delta;\delta' \in \Delta_{\mathcal{A}}(p,q)$. This proves that $\Delta_{\mathcal{A}}(p,q) \supseteq \{\delta';\delta \mid \delta' \in \Delta_{\mathcal{A}}(p,p)\}$.
   Conversely, assume $\delta'' \in \Delta_{\mathcal{A}}(p,q)$. By symmetry of $\mathcal{R}_{\mathcal{A}}$, $\delta^{-1} \in \Delta_{\mathcal{A}}(q,p)$. Hence, by setting $\delta' \stackrel{\text{def}}{=} \delta^{-1};\delta''$ we obtain $\delta' \in \Delta_{\mathcal{A}}(q,q)$. Clearly, $\delta'' = \delta;\delta'$, and this proves that $\Delta_{\mathcal{A}}(p,q) \subseteq \{\delta';\delta \mid \delta' \in \Delta_{\mathcal{A}}(p,p)\}$.
3. Let $\delta \in \Delta_{\mathcal{A}}(p,q)$ (set $\Delta_{\mathcal{A}}(p,q)$ is not empty since $p \equiv q$). Then $\delta^{-1} \in \Delta_{\mathcal{A}}(q,p)$ by symmetry of $\mathcal{R}_{\mathcal{A}}$.
   If $\rho' \in \Delta_{\mathcal{A}}(q,q)$ then $\rho \stackrel{\text{def}}{=} \delta;\rho';\delta^{-1} \in \Delta_{\mathcal{A}}(p,p)$ by Proposition 6.10.
   Conversely, if $\rho \in \Delta_{\mathcal{A}}(p,p)$ then $\rho' \stackrel{\text{def}}{=} \delta^{-1};\rho;\delta \in \Delta_{\mathcal{A}}(p,p)$.

□

Also the transitions can be partitioned in a similar fashion. So, on the transitions we 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 6.17 also hold for transitions.

Now we are ready to define the minimal HDS-automaton corresponding to a given HDS-automaton $\mathcal{A}$. It is obtained by replacing each class of equivalent states and transitions of $\mathcal{A}$ with a single state or transition. The symmetries associated to states and transitions of the minimal HDS-automaton are those defined by $\Delta_{\mathcal{A}}$: these, in fact, express all the symmetries that exist between the names, not only those "declared" in HDS-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 HDS-automata.

In the definition of the minimal HDS-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.

A HDS-automaton may contain states and transitions that are not reachable from the initial state. These states and transitions should not appear in the minimal HDS-automaton, since they do not contribute to the definition of the behavior of the automaton. We start by defining formally reachable states and transitions.

Definition 6.18 (reachable states) *Reachable states and reachable transitions of HDS-automaton $\mathcal{A}$ are the smallest sets of states and transitions that are closed for the following rules:*

- *q_0 is a reachable state;*
- *for all $t \in T$, if $s(t)$ is a reachable state, then t is a reachable transition;*
- *for all $t \in T$, if t is a reachable transition, then $d(t)$ is a reachable state.*

The definition of minimal HDS-automaton follows.

Definition 6.19 (minimal HDS-automaton) *The minimal HDS-automaton $\mathcal{A}_{\min}$ 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* $\mathsf{Q}_{\min}[q] = \Delta_{\mathcal{A}}(q,q)$ *for each* $q \in \mathrm{Q}_{\min}$*;*
- $\mathrm{T}_{\min} = \{\lfloor t \rfloor_{\equiv_{\mathcal{A}}} \mid t \in \mathrm{T},\ t$ *reachable transition*$\}$ *and* $\mathsf{T}_{\min}[t] = \Delta_{\mathcal{A}}(t,t)$ *for each* $t \in \mathrm{T}_{\min}$*;*
- $\mathrm{o}_{\min}(t) = \mathrm{o}(t)$ *and* $\mathsf{o}_{\min}[t] = \mathsf{o}[t]$ *for each* $t \in \mathrm{T}_{\min}$*;*
- $\mathrm{s}_{\min}(t) = \lfloor \mathrm{s}(t) \rfloor_{\equiv_{\mathcal{A}}}$ *and* $\mathsf{s}_{\min}[t] = \{\sigma \mid \sigma = \sigma';\sigma''$ *for* $\sigma' \in \mathrm{s}(t)$ *and* $\sigma'' \in \Delta_{\mathcal{A}}(s(t),\lfloor s(t) \rfloor_{\equiv_{\mathcal{A}}})\}$ *for each* $t \in \mathrm{T}_{\min}$*;*
- $\mathrm{d}_{\min}(t) = \lfloor \mathrm{d}(t) \rfloor_{\equiv_{\mathcal{A}}}$ *and* $\mathsf{d}_{\min}[t] = \{\sigma \mid \sigma = \sigma';\sigma''$ *for* $\sigma' \in \mathrm{d}(t)$ *and* $\sigma'' \in \Delta_{\mathcal{A}}(d(t),\lfloor d(t) \rfloor_{\equiv_{\mathcal{A}}})\}$ *for each* $t \in \mathrm{T}_{\min}$*;*
- $q_{0\,\min} = \lfloor q_0 \rfloor_{\equiv_{\mathcal{A}}}$ *and* $f_{0\,\min} = \{\sigma \mid \sigma = \sigma';\sigma''$ *for* $\sigma' \in f_0$ *and* $\sigma'' \in \Delta_{\mathcal{A}}(q_0, q_{\min 0})\}$.

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

Proposition 6.20 *Let $\mathcal{A}$ be a HDS-automaton. Then $\mathcal{A} \sim \mathcal{A}_{\min}$.*

Proof (Sketch). It is easy to prove that

$$\mathcal{R} \overset{\text{def}}{=} \{\langle q, \sigma, \lfloor q \rfloor_{\equiv_{\mathcal{A}}} \rangle \mid q \in Q_{\mathcal{A}} \text{ reachable state} \wedge \sigma \in \Delta_{\mathcal{A}}(q, \lfloor q \rfloor_{\equiv_{\mathcal{A}}})\}$$

is a HDS-bisimulation. □

Now we show that minimal HDS-automata are unique, up to isomorphism, for each class of bisimilar HDS-automata. We start by defining isomorphic HDS-automata.

Definition 6.21 (isomorphic HDS-automata) *An* isomorphism *between two HDS-automata $\mathcal{A}$ and $\mathcal{B}$ is a pair of named functions with symmetries $(\mathsf{i}_Q, \mathsf{i}_T)$ with $\mathsf{i}_Q : \mathsf{Q}_{\mathcal{A}} \to \mathsf{Q}_{\mathcal{B}}$ and $\mathsf{i}_T : \mathsf{T}_{\mathcal{A}} \to \mathsf{T}_{\mathcal{B}}$ such that:*

- *$i_Q : Q_{\mathcal{A}} \longleftrightarrow Q_{\mathcal{B}}$ and, for each $q \in Q_{\mathcal{A}}$, holds $\mathsf{Q}_{\mathcal{B}}[i_Q(q)] = \mathsf{i}_\mathsf{Q}[q]^{-1}; \mathsf{Q}_{\mathcal{A}}[q]; \mathsf{i}_\mathsf{Q}[q]$;*
- *$i_T : T_{\mathcal{A}} \longleftrightarrow T_{\mathcal{B}}$ and, for each $t \in T_{\mathcal{A}}$, holds $\mathsf{T}_{\mathcal{B}}[i_T(t)] = \mathsf{i}_\mathsf{T}[t]^{-1}; \mathsf{T}_{\mathcal{A}}[t]; \mathsf{i}_\mathsf{T}[t]$;*
- *for all $t \in T_{\mathcal{A}}$ hold $o_{\mathcal{A}}(t) = o_{\mathcal{B}}(i_T(t))$ and $\mathsf{o}_{\mathcal{A}}[t] = \mathsf{i}_\mathsf{T}[t]; \mathsf{o}_{\mathcal{B}}[i_T(t)]$;*
- *for all $t \in T_{\mathcal{A}}$ hold $i_Q(s_{\mathcal{A}}(t)) = s_{\mathcal{B}}(i_T(t))$ and $\mathsf{s}_{\mathcal{A}}[t]; \mathsf{i}_\mathsf{Q}[s_{\mathcal{A}}(t)] = \mathsf{i}_\mathsf{T}[t]; \mathsf{s}_{\mathcal{B}}[i_T(t)]$;*
- *for all $t \in T_{\mathcal{A}}$ hold $i_Q(d_{\mathcal{A}}(t)) = d_{\mathcal{B}}(i_T(t))$ and $\mathsf{d}_{\mathcal{A}}[t]; \mathsf{i}_\mathsf{Q}[d_{\mathcal{A}}(t)] = \mathsf{i}_\mathsf{T}[t]; \mathsf{d}_{\mathcal{B}}[i_T(t)]$.*

It is easy to see that, by changing the canonical representatives $\lfloor q \rfloor_{\equiv_{\mathcal{A}}}$ of the states of HDS-automaton $\mathcal{A}$, isomorphic minimal HDS-automata are obtained. Moreover, isomorphic HDS-automata are HDS-bisimilar, as shown by the following proposition.

Proposition 6.22 *Let $\mathcal{A}$ and $\mathcal{B}$ be two isomorphic HDS-automata. Then $\mathcal{A}$ and $\mathcal{B}$ are HDS-bisimilar.*

Proof (Sketch). Let $(\mathsf{i}_\mathsf{Q}, \mathsf{i}_\mathsf{T})$ be an isomorphism between $\mathcal{A}$ and $\mathcal{B}$. It is easy to prove that

$$\mathcal{R} \overset{\text{def}}{=} \{\langle q, \mathsf{i}_\mathsf{Q}[q], i_Q(q) \rangle \mid q \in Q_{\mathcal{A}}\}$$

is a HDS-bisimulation. □

The next lemma shows that two equivalent states q_1 and q_2 in two bisimilar HDS-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ have isomorphic symmetries $\Delta_{\mathcal{A}_1}(q_1, q_1)$ and $\Delta_{\mathcal{A}_1}(q_2, q_2)$. This lemma will be used in the proof that two bisimilar HDS-automata have isomorphic minimal realizations.

Lemma 6.23 *Let $q_1 \in Q_1$ and $q_2 \in Q_2$ be two states of two HDS-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ and let $\Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2) \overset{\text{def}}{=} \{\delta \mid \langle q_1, \delta, q_2 \rangle \in \mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2}\}$. Then, for any $\delta \in \Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2)$:*

$$\Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2) = \Delta_{\mathcal{A}_1}(q_1, q_1); \delta = \delta; \Delta_{\mathcal{A}_2}(q_2, q_2).$$

Moreover, $\Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2)$ is an embedding of symmetry $\Delta_{\mathcal{A}_1}(q_1)$ into symmetry $\Delta_{\mathcal{A}_2}(q_2)$.

Proof. Let $\delta \in \Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2)$. We prove that $\Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2) = \Delta_{\mathcal{A}_1}(q_1, q_1); \delta$. By Proposition 6.10, if $\delta_1 \in \Delta_{\mathcal{A}_1}(q_1, q_1)$ then $\delta_1; \delta \in \Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2)$. This proves that $\Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2) \supseteq \{\delta_1; \delta \mid \delta_1 \in \Delta_{\mathcal{A}_1}(q_1, q_1)\}$. Conversely, assume $\delta' \in \Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2)$. Let $\delta_1 \overset{\text{def}}{=} \delta'; \delta^{-1}$. By Proposition 6.10 we have $\delta_1 \in \Delta_{\mathcal{A}_1}(q_1, q_1)$. Clearly, $\delta' = \delta_1; \delta$, and this proves that $\Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2) \subseteq \{\delta_1; \delta \mid \delta_1 \in \Delta_{\mathcal{A}_1}(q_1, q_1)\}$.
The proof that $\Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2) = \delta; \Delta_{\mathcal{A}_2}(q_2)$ for any $\delta \in \Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2)$ is similar.
Finally, from $\Delta_{\mathcal{A}_1}(q_1, q_1); \delta = \delta; \Delta_{\mathcal{A}_2}(q_2, q_2)$ we obtain $\delta^{-1}; \Delta_{\mathcal{A}_1}(q_1, q_1); \delta = \Delta_{\mathcal{A}_2}(q_2, q_2)$. This concludes the proof that $\Delta_{\mathcal{A}_1;\mathcal{A}_2}(q_1, q_2)$ is a symmetry embedding. □

Proposition 6.24 *Let $\mathcal{A}$ and $\mathcal{B}$ be two HDS-automata such that $\mathcal{A} \sim \mathcal{B}$. Then $\mathcal{A}_{\min}$ and $\mathcal{B}_{\min}$ are isomorphic.*

Proof. Let $i_Q \subseteq Q_{\mathcal{A}_{\min}} \times Q_{\mathcal{B}_{\min}}$ be defined as follows:

$$(q_a, q_b) \in i_Q \quad \text{iff} \quad \langle q_a, \delta, q_b \rangle \in \mathcal{R}_{\mathcal{A},\mathcal{B}} \text{ for some } \delta.$$

Now we prove that $i_Q : Q_{\mathcal{A}_{\min}} \to Q_{\mathcal{B}_{\min}}$, namely that q_a in $Q_{\mathcal{A}_{\min}}$ there is one, and only one, q_b in $Q_{\mathcal{B}_{\min}}$ such that $\langle q_a, \delta, q_b \rangle \in \mathcal{R}_{\mathcal{A},\mathcal{B}}$ for some δ. The fact that there is (at least) one q_b depends on the fact that q_a is reachable in $\mathcal{A}$, and hence a matching state q_b should exist in $\mathcal{B}$ according to the definition of HDS-bisimulation. On the other hand, if q_b and q'_b are two states of $Q_{\mathcal{B}_{\min}}$ that can match q_a, then $q_b \equiv_{\mathcal{B}} q'_b$, and hence $q_b = q'_b$ by definition of minimal HDS-automaton. By the symmetry in the definition of i_Q we deduce that $i_Q^{-1} : Q_{\mathcal{B}_{\min}} \to Q_{\mathcal{A}_{\min}}$, and hence $i_Q : Q_{\mathcal{A}_{\min}} \longleftrightarrow Q_{\mathcal{B}_{\min}}$.

Let us extend function i_Q to a named function with symmetries Let us define $\mathsf{i}_\mathsf{Q} : \mathsf{Q}_{\mathcal{A}_{\min}} \to \mathsf{Q}_{\mathcal{B}_{\min}}$ as follows:

$$\mathsf{i}_\mathsf{Q}[q_a] = \{\delta \mid \langle q_a, \delta, q_b \rangle \in \mathcal{R}_{\mathcal{A},\mathcal{B}}\}.$$

The fact that $\mathsf{i}_\mathsf{Q}[q_a]$ is indeed an embedding of $\mathsf{Q}_\mathcal{A}[q_a]$ into $\mathsf{Q}_\mathcal{B}[i_Q(q_a)]$ is a consequence of Lemma 6.23.
Finally, again by Lemma 6.23 applied to $\mathsf{i}_\mathsf{Q}[q_a]^{-1}$, we have $\mathsf{Q}_\mathcal{B}[i_Q(q)] = \mathsf{i}_\mathsf{Q}[q]^{-1}; \mathsf{Q}_\mathcal{A}[q]; \mathsf{i}_\mathsf{Q}[q]$.
We define $\mathsf{i}_\mathsf{T} : \mathsf{T}_{\mathcal{A}_{\min}} \to \mathsf{T}_{\mathcal{B}_{\min}}$ similarly to i_Q, namely:

$$(t_a, t_b) \in i_T \quad \text{iff} \quad \langle t_a, \delta, t_b \rangle \in \mathcal{R}_{\mathcal{A},\mathcal{B}} \text{ for some } \delta.$$

and

$$\mathsf{i}_\mathsf{T}[t_a] = \{\delta \mid \langle t_a, \delta, t_b \rangle \in \mathcal{R}_{\mathcal{A},\mathcal{B}}\}.$$

Also in this case, i_T is well defined, and $\mathsf{T}_\mathcal{B}[i_T(t)] = \mathsf{i}_\mathsf{T}[t]^{-1}; \mathsf{T}_\mathcal{A}[t]; \mathsf{i}_\mathsf{T}[t]$.
To conclude the proof we have to show that $(\mathsf{i}_\mathsf{Q}, \mathsf{i}_\mathsf{T})$ is an isomorphism between $\mathcal{A}_{\min}$ and $\mathcal{B}_{\min}$. We have already proved the first two items in the definition of HDS-isomorphism. The other items follow easily by the definition of HDS-bisimulation. ☐

Theorem 6.25 *Let $\mathcal{A}$ and $\mathcal{B}$ be two HDS-automata. Then $\mathcal{A} \sim \mathcal{B}$ if and only if $\mathcal{A}_{\min}$ and $\mathcal{B}_{\min}$ are isomorphic.*

Proof. The "only if" implication consists of Proposition 6.24.
For the "if" implication, assume that $\mathcal{A}_{\min}$ and $\mathcal{B}_{\min}$ are isomorphic. Then, by Proposition 6.22, we have $\mathcal{A}_{\min} \sim \mathcal{B}_{\min}$. By Proposition 6.20 we also have that $\mathcal{A} \sim \mathcal{A}_{\min}$ and that $\mathcal{B}_{\min} \sim \mathcal{B}$. By transitivity of $\sim$ we conclude that $\mathcal{A} \sim \mathcal{B}$. ☐

The obtained HDS-automaton $\mathcal{A}_{\min}$ is *minimal* since it has the minimum number of states and of transitions among the HDS-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 HDS-automaton is obtained: this is a consequence of Proposition 6.24.

7 A categorical approach to history dependent automata

In this section we give an alternative characterization of HD-automata and of HD-bisimulation in a categorical setting. This approach works both for Basic HD-automata and for HD-automata with Symmetries.

We start by defining HD-automata by extending a classical categorical definitions of ordinary automata. (Section 7.1). Then, we exploit open maps [JNW96] to define HD-bisimulation on these categories of HD-automata (Section 7.2); we also show that, in the case of HD-automata with Symmetries, the minimal HDS-automaton, that we have built explicitly in Section 6.5, is reobtained in the categorical setting.

7.1 The categories of HD-automata

In a categorical setting, an ordinary automaton is classically defined as a diagram

$$\begin{tikzcd} L & T \arrow[l, "o"'] \arrow[r, bend left, "s"] \arrow[r, bend right, "d"'] & Q & I \arrow[l, "i"'] \end{tikzcd}$$

in the category $\mathbf{Set}$ of (small) sets. Usually, set I is a singleton $\{*\}$ and the initial state q_0 of the automaton is designated as $i(*)$; we only consider such single-pointed automata in this chapter.

Given two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ on the same set L of labels, a *morphism* $m : \mathcal{A}_1 \to \mathcal{A}_2$ is a pair of arrows $m_Q : Q_1 \to Q_2$ and $m_T : T_1 \to T_2$ that respect sources, destinations, labels, and the initial state.

Automata and morphisms between automata form a category $\mathbf{Aut}$. It is often useful to consider the full subcategory of $\mathbf{Aut}$ whose automata have the same set L of labels: this category is denoted by $\mathbf{Aut}_L$.

7.1.1 Categories of enriched sets

Basic HD-automata and HD-automata with Symmetries can be defined in a similar way; we have just to replace the category $\mathbf{Set}$ with a category of "sets enriched with names" and of "sets enriched with symmetries". Now we show that also these "enriched sets" can be defined by exploiting a standard categorical construction, namely the $\coprod$-construction.

Definition 7.1 ($\coprod$-construction) *Given a category $\mathbf{C}$, the* free coproduct completion *of $\mathbf{C}$ is the category $\coprod \mathbf{C}$ defined as follows:*

- *its objects are families (i.e., indexed sets) of objects of* $\mathbf{C}$*; more precisely, each object* A *of* $\coprod \mathbf{C}$ *is a pair* $(\mathrm{A}, (\mathsf{A}[a])_{a \in \mathrm{A}})$*, where* A *is a set and, for each* $a \in \mathrm{A}$*,* $\mathsf{A}[a]$ *is an object of* $\mathbf{C}$*;*

- *its arrows* f *from* A *to* B *are pairs* $(\mathrm{f}, (\mathsf{f}[a])_{a \in \mathrm{A}})$*, where* $\mathrm{f} : \mathrm{A} \to \mathrm{B}$ *and, for each* $a \in \mathrm{A}$*,* $\mathsf{f}[a] : \mathsf{A}[a] \to \mathsf{B}[f(a)]$*is an arrow in* $\mathbf{C}$*;*

- *identity arrows and composition are defined as usual.*

Notice that $\coprod \mathbf{C} = \left(\prod(\mathbf{C}^{\mathrm{op}})\right)^{\mathrm{op}}$, where $\prod(\mathbf{D})$ is the free product completion of category $\mathbf{D}$.

Now we revise the definitions of named sets (Definition 4.2) and of named sets with symmetries (Definition 6.4) in a categorical setting, by using the $\coprod$-construction.

Definition 7.2 (category $\mathbf{NSet}$ of named sets) *Let* $\mathcal{N}$ *be an infinite, denumerable set of* names.
Category $\mathbf{J}$ *of* injective functions on names *is defined as follows:*

- *objects of* $\mathbf{J}$ *are the subsets of* $\mathcal{N}$*;*

- *arrows of* $\mathbf{J}$ *are the injective functions between the subsets of* $\mathcal{N}$*;*

- *identity and composition are defined as usual.*

Category $\mathbf{NSet}$ *of* named sets *is defined as follows:* $\mathbf{NSet} = \coprod(\mathbf{J}^{\mathrm{op}})$.

We would like to remark that category $\mathbf{J}$ is similar to category $\mathbf{I}$ of injective functions on finite sets that has been used in [Sta96], [FMS96], and [CSW97] as the category of names in semantic models of π-calculus. The difference is that sets of cardinality ω are allowed in $\mathbf{J}$, while they are not in $\mathbf{I}$.

Notice that named sets are built on the top of category $\mathbf{J}^{\mathrm{op}}$, rather than of $\mathbf{J}$: in fact, according to Definition 4.2, in a function between named sets, inverse injective functions are used on the names.

Definition 7.3 (category $\mathbf{Sym}$ of named sets with symmetries) *Category* $\mathbf{Sym}$ *of the* symmetries on $\mathcal{N}$ *is defined as follows:*

- *objects of* $\mathbf{Sym}$ *are the symmetries on* $\mathcal{N}$ *(Definition 6.1);*

- *arrows of* $\mathbf{Sym}$ *are the embeddings on symmetries on* $\mathcal{N}$ *(Definition 6.3);*

- *if* Σ *is an object of* $\mathbf{Sym}$*, then the identity arrow for* Σ *is defined as* $\mathrm{id}_\Sigma = \Sigma$*;*

- *if* $f : \Sigma \to \Sigma'$ *and* $f' : \Sigma' \to \Sigma''$ *are two arrows of* $\mathbf{Sym}$*, then their composition* $f; f' : \Sigma \to \Sigma''$ *is defined as* $f; f'' = \{\sigma; \sigma' \mid \sigma \in f, \sigma' \in f'\}$.

Category $\mathbf{SymSet}$ *of* named sets with symmetries *is defined as follows:* $\mathbf{SymSet} = \coprod \mathbf{Sym}$.

7.1.2 Defining the enriched automata

Enriched automata on a base category $\mathbf{C}$ are defined by taking states, transitions, labels (and the morphisms between them) from category $\coprod \mathbf{C}$.

Definition 7.4 (category of enriched automata) *Let* $\mathbf{C}$ *be a base category. An* enriched automaton *on* $\mathbf{C}$ *is a diagram*

$$\begin{tikzcd} L & T \arrow[l, "o"'] \arrow[r, bend left, "s"] \arrow[r, bend right, "d"'] & Q & I \arrow[l, "i"'] \end{tikzcd}$$

in category $\coprod \mathbf{C}$.
Let $\mathcal{A}_1$ *and* $\mathcal{A}_2$ *be two enriched automata on* $\mathbf{C}$ *with the same labels* L *and initial points* I*. A* morphism $m : \mathcal{A}_1 \to \mathcal{A}_2$ *is a pair of arrows* $m_Q : Q_1 \to Q_2$ *and* $m_T : T_1 \to T_2$ *that respect sources, destinations, labels, and the initial states, i.e., such that the two overlapped diagrams*

$$\begin{tikzcd} & T_1 \arrow[dl, "o_1"'] \arrow[r, bend left, "s_1"] \arrow[r, bend right, "d_1"'] \arrow[dd, "m_T"] & Q_1 \arrow[dd, "m_Q"] & \\ L & & & I \arrow[ul, "i_1"'] \arrow[dl, "i_2"] \\ & T_2 \arrow[ul, "o_2"] \arrow[r, bend left, "s_2"] \arrow[r, bend right, "d_2"'] & Q_2 & \end{tikzcd}$$

in category $\coprod \mathbf{C}$ commute in the obvious way. The identity morphism and the composition of two morphisms are defined component-wise.
We denote with $\mathbf{Aut}(\mathbf{C})$ the category of enriched automata on category $\mathbf{C}$. Moreover, we denote with $\mathbf{Aut}(\mathbf{C})_{L,I}$ the full subcategory of the enriched automata on category $\mathbf{C}$ with labels L and initial points I. Subcategories $\mathbf{Aut}(\mathbf{C})_I$ and $\mathbf{Aut}(\mathbf{C})_L$ are defined similarly.

The general definition of enriched automata can be specialized to obtain Basic HD-automata and HD-automata with Symmetries.

Definition 7.5 (Basic HD-automata) *Let I be the named set with $*$ as singleton element and with $\mathsf{I}[*] = \mathcal{N}$.*
Category $\mathbf{HD}$ of the (Basic) HD-automata *is the category of $\mathbf{Aut}(\mathbf{J}^{\mathrm{op}})_{\mathsf{I}}$.*
Moreover, let L be a named set. Category $\mathbf{HD}_{\mathsf{L}}$ is the full subcategory of $\mathbf{Aut}(\mathbf{J}^{\mathrm{op}})_{\mathsf{L},\mathsf{I}}$ whose objects respect the observation condition.

Definition 7.6 (HDS-automata) *Let I be the named set with symmetries that has $*$ as singleton element and such that $\mathsf{I}[*] = \{\mathrm{id}_{\mathcal{N}}\}$.*
Category $\mathbf{HDS}$ of the HD-automata with Symmetries *is the category of $\mathbf{Aut}(\mathbf{Sym})_{\mathsf{I}}$.*
Moreover, let L be a named set with symmetries. Category $\mathbf{HDS}_{\mathsf{L}}$ of the HD-automata with Symmetries on labels L is the category $\mathbf{Aut}(\mathbf{Sym})_{\mathsf{L},\mathsf{I}}$.

We would like to remark that the approach followed above to define Basic HD-automata and HDS-automata in a categorical framework is quite general. To define a category of enriched automata, it is sufficient to give the base category $\mathbf{C}$ that defines the information that enriches the automata. Then, $\coprod(\mathbf{C})$ defines the sets enriched with the information in $\mathbf{C}$, and $\mathbf{Aut}(\mathbf{C})$ defines the category of enriched automata on $\mathbf{C}$.

A comment is in order on the choice of the enriched set I of initial points. We remark that, for the case of HD-automata, set $\mathcal{N}$ is the only *weak initial* object[8] in category $\mathbf{J}^{\mathrm{op}}$. Similarly, symmetry $\{\mathrm{id}_{\mathcal{N}}\}$ is the only *weak initial* object in category $\mathbf{Sym}$. In general, we propose to take as I the singleton set $\{*\}$, and to let $\mathsf{I}[*]$ be the weak initial object c_0 of the base category $\mathbf{C}$. In this way, object c_0 represents the case of maximal information on the state of the system (e.g., all names known, no symmetry on the names), and arrow $\mathsf{i} : \mathsf{I} \to \mathsf{Q}$ models the correspondence $\mathsf{i}[*]$ that exists between this maximal, global information and the "local" information associated to the initial state $q_0 = i(*)$.

It is interesting to observe that ordinary automata can be obtained by applying definition 7.4 to the category $\mathbf{1}$ (we recall that category $\mathbf{1}$ has a single object, and the identity arrow for it).

Fact 7.7 *Category $\mathbf{Set}$ is isomorphic to category $\coprod \mathbf{1}$. As a consequence, category $\mathbf{Aut}$ of the automata is isomorphic to category $\mathbf{Aut}(\mathbf{1})$ of the enriched automata on category $\mathbf{1}$.*

7.2 Open maps and bisimulations

Consider a morphism $m : \mathcal{A}_1 \to \mathcal{A}_2$ in the category of automata. Relation

$$\mathcal{R} = \{\langle q_1, q_2 \rangle \in Q_1 \times Q_2 \mid q_2 = m_Q(q_1)\}$$

is a simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$. In fact, assume $q_1 \;\mathcal{R}\; q_2$ and $t_1 : q_1 \xrightarrow{l} q_1'$; then we have $t_2 : q_2 \xrightarrow{l} q_2'$ and $q_1' \;\mathcal{R}\; q_2'$ by taking $t_2 = m_T(t_1)$ and $q_2' = m_Q(q_1')$. Moreover $q_{01} \;\mathcal{R}\; q_{02}$, since $q_{02} = m_Q(q_{01})$.

Therefore, a morphism $m : \mathcal{A}_1 \to \mathcal{A}_2$ expresses the fact that all the transitions of $\mathcal{A}_1$ can be simulated in $\mathcal{A}_2$, starting from the initial states. In general, however, it is not true that all the transitions of $\mathcal{A}_2$ can be simulated in $\mathcal{A}_1$.

Nevertheless, it is possible to define a particular class of "bisimulation" morphisms, such that the existence of such a morphism from $\mathcal{A}_1$ to $\mathcal{A}_2$ guarantees not only that the transitions of $\mathcal{A}_1$ can be adequately simulated in $\mathcal{A}_2$ but also the converse; i.e., the existence of a "bisimulation" morphism guarantees that $\mathcal{A}_1$ and $\mathcal{A}_2$ are bisimilar.

There exist bisimilar automata $\mathcal{A}_1$ and $\mathcal{A}_2$ such that no "bisimulation" morphism (nor generic morphisms) can be found between them. However, whenever two automata $\mathcal{A}_1$ and $\mathcal{A}_2$ are bisimilar, it is possible to find a common predecessor $\mathcal{A}$ and a span of "bisimulation" morphisms $m_1 : \mathcal{A} \to \mathcal{A}_1$ and $m_2 : \mathcal{A} \to \mathcal{A}_2$ between them:

$$\begin{tikzcd} & \mathcal{A} \arrow[dl, "m_1"'] \arrow[dr, "m_2"] & \\ \mathcal{A}_1 & & \mathcal{A}_2 \end{tikzcd}$$

The span of morphisms is necessary for being able to define generic relations between the states of the two automata.

[8]We recall that object c_0 is weak initial in category $\mathbf{C}$ if for any other object x of $\mathbf{C}$ there is some (not necessarily unique) arrow from c_0 to x.

These “bisimulation” morphisms have been defined in various manner in the literature, and different names have been given to them. They are called *abstraction homomorphisms* in [Cas87] and in [MS89], *zig-zag morphisms* in [vB84], *transition preserving homomorphisms* in [FM90] and in [FMM97], and *open maps* in [JNW96]. Here we consider the approach based on open maps; this approach has the advantage of being general enough to be applied not only to automata, but also to other models of concurrency, like Petri nets and event structures.

Assume a category $\mathbf{M}$ of *models*. Let $\mathbf{E}$ be the subcategory of $\mathbf{M}$ whose objects are the *experiments* that can be executed on $\mathbf{M}$ and whose arrows express how the experiments can be extended. If X is an object of $\mathbf{E}$ and M is an object of $\mathbf{M}$, an arrow $x : X \to M$ of $\mathbf{M}$ represents the execution of the experiment X in the model M.

Consider an arrow $m : M \to N$ in $\mathbf{M}$. We can see this arrow as a simulation of model M in model N. Hence, correctly, if an experiment X can be executed in M (there exists an arrow $x : X \to M$) and N can simulate M (there exists an arrow $m : M \to N$) then the experiment X can be executed in N (via the arrow $x; m : X \to N$).

Suppose now to extend the experiment X to an experiment Y (via an arrow $f : X \to Y$ in $\mathbf{E}$) and that an arrow $y : Y \to N$ exists such that the following diagram commutes in $\mathbf{M}$:

$$\begin{tikzcd} X \arrow[r, "x"] \arrow[d, "f"'] & M \arrow[d, "m"] \\ Y \arrow[r, "y"] & N \end{tikzcd} \qquad (*)$$

This means that the execution of the experiment X in N (via $x; m$) can be extended to an execution of the experiment Y in N (via y).

This does not imply in general that also the execution of X in M can be extended to an execution of Y in M. We can make this sure by requiring that there is an arrow y' such that the diagram

$$\begin{tikzcd} X \arrow[r, "x"] \arrow[d, "f"'] & M \arrow[d, "m"] \\ Y \arrow[r, "y"] \arrow[ur, "y'"] & N \end{tikzcd} \qquad (**)$$

commutes. Given $m : M \to N$, if for each commuting diagram $(*)$ there is an arrow y' such that also $(**)$ commutes, we say that m is an $\mathbf{E}$-*open map*.

It is easy to check that the open maps form a subcategory of $\mathbf{M}$ (i.e., identities are open and open maps are closed for composition).

Definition 7.8 (open bisimulation) *We say that two objects M_1 and M_2 of $\mathbf{M}$ are* open-bisimilar *with respect to $\mathbf{E}$ if and only if there is a span of $\mathbf{E}$-open maps m_1, m_2.*

$$\begin{tikzcd} & M \arrow[dl, "m_1"'] \arrow[dr, "m_2"] & \\ M_1 & & M_2 \end{tikzcd}$$

In [JNW96] it is shown that, if the category $\mathbf{Aut}_L$ is used as the category of the models and the full subcategory $\mathbf{Bran}_L$ of the *branches* (i.e., of those finite automata which consist of a linear sequence of transitions) is used as the category of experiments, then two automata are open-bisimilar if and only if they are bisimilar according to Definition 2.3.

Now we apply open maps to Basic HD-automata and to HD-automata with Symmetries.

7.2.1 Application to basic HD-automata

In the case of Basic HD-automata, an experiment is a finite sequences of transitions and an extended experiment can be obtained by adding new transitions. Moreover, we require that no name is forgotten during an experiment, since this models the idea that the observer can remember all the names previously used in the experiment.

Definition 7.9 (category of HD-experiments) *A HD-automaton $\mathcal{X}$ is a* HD-experiment *if:*

- *$Q = \{q_0, q_1, \ldots, q_n\}$ are the states and $T = \{t_1, \ldots, t_n\}$ are the transitions, and $s(t_i) = q_{i-1}$ and $d(t_i) = q_i$;*
- *for all $t \in T$, $\mathsf{d}[t] : \mathsf{T}[t] \longleftrightarrow \mathsf{Q}[q]$ is a bijection.*

A morphism $\langle \mathsf{m_Q}, \mathsf{m_T} \rangle : \mathcal{X} \to \mathcal{X}'$ *is* name preserving *if* $\mathsf{m_Q}$ *and* $\mathsf{m_T}$ *are bijections on the names, i.e.,* $\mathsf{m_Q}[q]$ *is a bijection between* $\mathsf{Q}'[m_Q(q)]$ *and* $\mathsf{Q}[q]$ *for all* $q \in Q$*, and similarly for* $\mathsf{m_T}$*.*
The category $\mathbf{HD\text{-}Exp}$ of HD-experiments *is the subcategory of* $\mathbf{HD}$ *with HD-experiments as objects and name preserving morphisms as arrows.*
Category $\mathbf{HD\text{-}Exp_L}$ *is the full subcategory of* $\mathbf{HD\text{-}Exp}$ *whose objects are* $\mathbf{HD}_\mathsf{L}$*-automata.*

Now we show that the notion of HD-bisimulation given in Definition 4.10 is also obtained by the uniform technique of the open maps.

Theorem 7.10 *Two Basic HD-automata on the same labels* L *are HD-bisimilar iff they are open-bisimilar w.r.t. experiments* $\mathbf{HD\text{-}Exp_L}$*.*

Proof. We prove the two implications separately.
Proof of the "if" implication. We show that, if $m : \mathcal{A} \to \mathcal{B}$ in a $\mathbf{HD\text{-}Exp}$-open map, then $\mathcal{A}$ and $\mathcal{B}$ are HD-bisimilar. From this, by transitivity of HD-bisimilarity, it is easy to conclude that two HD-automata are HD-bisimilar if there is a span of open maps for them. We show that $\mathcal{R}$ is a HD-bisimulation for $\mathcal{A}$ and $\mathcal{B}$, where

$$\mathcal{R} \overset{\text{def}}{=} \{\langle q_1, \delta, q_2 \rangle \mid q_1 \text{ reachable}, q_2 = m_Q(q_1), \delta = \mathsf{m_Q}[q_1]\}$$

Suppose that $\langle q_1, \delta, q_2 \rangle \in \mathcal{R}$ and that $t_1 : q_1 \xrightarrow{l} q_1'$. It is easy to check that t_1 is matched by $m_T(t_1) = t_2 : q_2 \xrightarrow{l} q_2'$, choosing $\zeta = \mathsf{m_T}[t_1]$ and $\xi = \zeta|_{\mathsf{T}_\mathcal{A}[t_1]_{\text{new}}}$.
Suppose, conversely, that $t_2 : q_2 \xrightarrow{l} q_2'$. Since q_1 is reachable, there is an experiment $\mathcal{X}$ and a morphism $x : \mathcal{X} \to \mathcal{A}$ so that experiment $\mathcal{X}$ terminates on q_1. Via $x; m$, experiment $\mathcal{X}$ can be executed in $\mathcal{B}$, ending in state q_2.
Now we extend experiment $\mathcal{X}$ to an experiment $\mathcal{Y}$, obtained by adding a transition corresponding to t_2. Let the last state of $\mathcal{X}$ be q_n. Then $\mathcal{Y}$ is obtained from $\mathcal{X}$ by adding a state q_{n+1} and a transition $t_{n+1} : q_n \xrightarrow{l} q_{n+1}$. Let us define $\mathsf{T}_\mathcal{Y}[t_{n+1}] = \mathsf{Q}_\mathcal{Y}[q_{n+1}] = \mathsf{Q}_\mathcal{X}[q_n] \uplus \mathsf{T}_2[t_2]_{\text{new}}$ (we assume, without loss of generality, that $\mathsf{Q}_\mathcal{Y}[q_n]$ and $\mathsf{T}_2[t_2]_{\text{new}}$ are disjoint). Maps $\mathsf{s}_\mathcal{Y}[t_{n+1}]$ and $\mathsf{d}_\mathcal{Y}[t_{n+1}]$ are defined in the obvious way, whereas map $\mathsf{o}_\mathcal{Y}[t_{n+1}] = \alpha; \mathsf{o}_2[t_2]$, with $\alpha : \mathsf{T}_\mathcal{Y}[t_{n+1}] \longleftrightarrow \mathsf{T}_2[t_2]$ defined as follows:

- if $n \in \mathsf{T}_2[t_2]_{\text{new}}$ then $\alpha(n) = n$;
- if $n \in \mathsf{Q}_\mathcal{X}[q_n]$ then $\alpha(n) = \mathsf{s}_2[t_2]^{-1}(\mathsf{m_Q}[q_1](\mathsf{x_Q}[q_n](n)))$.

The definition of the name-preserving morphism $f : \mathcal{X} \to \mathcal{Y}$ is obvious, since $\mathcal{X}$ is a prefix of $\mathcal{Y}$.
Let us define morphism $y : \mathcal{Y} \to \mathcal{B}$ as follows: on $\mathcal{X}$, morphism y is defined by $x; m$. Moreover transition t_{n+1} is mapped to t_2 so that $\mathsf{y_T}[t_{n+1}] = \alpha$. Finally, state q_{n+1} is mapped on q_2' so that $\mathsf{y_Q}[q_{n+1}] = \mathsf{s}_2[t_2]; \mathsf{y_T}[t_{n+1}]$.
It is easy to check that diagram

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "x"] \arrow[d, "f"'] & \mathcal{A} \arrow[d, "m"] \\
\mathcal{Y} \arrow[r, "y"'] & \mathcal{B}
\end{tikzcd}$$

commutes. Since m is an open map, a morphism $y' : \mathcal{Y} \to \mathcal{A}$ exists so that diagram

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "x"] \arrow[d, "f"'] & \mathcal{A} \arrow[d, "m"] \\
\mathcal{Y} \arrow[r, "y"'] \arrow[ur, "y'"] & \mathcal{B}
\end{tikzcd}$$

commutes. Let us define $q_1' = y_Q'(q_{n+1})$ and $t_1 = y_T'(t_{n+1})$; then $t_1 : q_1 \xrightarrow{l} q_1'$. Let us define $\zeta = \mathsf{m_T}[t_1]$ and $\xi = \zeta|_{\mathsf{T}_1[t_1]_{\text{new}}}$.
Then it is easy to check that:

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

This concludes the proof of the first implication.
Proof of the "only if" implication. We show that, if $\mathcal{A}$ and $\mathcal{B}$ are HD-bisimilar automata, then here exist a HD-automaton $\mathcal{C}$ and a span of open maps

$$\begin{tikzcd}
& \mathcal{C} \arrow[dl, "m"'] \arrow[dr, "m'"] & \\
\mathcal{A} & & \mathcal{B}
\end{tikzcd}$$

Suppose that $\mathcal{A}$ and $\mathcal{B}$ are HD-bisimilar via the HD-bisimulation $\mathcal{R}$; by Proposition 4.15 we can assume that $\mathcal{R} = \widehat{\mathcal{R}}$.
The HD-automaton $\mathcal{C}$ and m and m' are defined as follows:

- $Q_{\mathcal{C}} = \mathcal{R}$; for each $r = \langle q_a, \delta, q_b\rangle \in \mathcal{R}$ we have $m_Q(r) = q_a$ and $m'_Q(r) = q_b$. Moreover
$$\mathsf{Q}_{\mathsf{c}}[r] = \big(\mathsf{Q}_{\mathcal{A}}[q_a] \setminus \mathrm{dom}(\delta)\big) \uplus \delta \uplus \big(\mathsf{Q}_{\mathcal{B}}[q_b] \setminus \mathrm{cod}(\delta)\big),$$
i.e., the names of state r in $\mathcal{C}$ are the union of the names of q_a and of q_b, where the names related by δ are identified. Finally, $\mathsf{m}_{\mathsf{Q}}[r]$ and $\mathsf{m}'_{\mathsf{Q}}[r]$ are defined as follows:
  - if $n \in \mathsf{Q}_{\mathcal{A}}[q_a] \setminus \mathrm{dom}(\delta)$ then $\mathsf{m}_{\mathsf{Q}}[r](n) = n$ and $\mathsf{m}'_{\mathsf{Q}}[r](n)$ is undefined;
  - if $(n, n') \in \delta$, then $\mathsf{m}_{\mathsf{Q}}[r](n, n') = n$ and $\mathsf{m}'_{\mathsf{Q}}[r](n, n') = n'$;
  - if $n' \in \mathsf{Q}_{\mathcal{B}}[q_b] \setminus \mathrm{cod}(\delta)$ then $\mathsf{m}_{\mathsf{Q}}[r](n)$ is undefined and $\mathsf{m}'_{\mathsf{Q}}[r](n) = n$.
- $T_{\mathcal{C}} = \mathcal{S}$, where $\langle t_a, \zeta, t_b\rangle \in \mathcal{S}$ if t_a and t_b are matching transitions according to $\zeta : \mathsf{T}_{\mathcal{A}}[t_a] \longrightarrow \mathsf{T}_{\mathcal{B}}[t_b]$. For each $t_c \in T_{\mathcal{C}}$, the set of names $\mathsf{T}_{\mathcal{C}}[t_c]$ and the embeddings $\mathsf{m}_{\mathsf{T}}[t_c]$ and $\mathsf{m}'_{\mathsf{T}}[t_c]$ are defined as the corresponding components for the states.
- The starting state of $\mathcal{C}$ is $\langle q_{0\mathcal{A}}, (\sigma_{0\mathcal{A}}; \sigma_{0\mathcal{B}}^{-1}), q_{0\mathcal{B}}, \rangle \in \mathcal{R}$.

It is easy to show that m and m' are morphisms. Now we show that m is an open map (the proof for m' is similar). Suppose then that diagram

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "x"] \arrow[d, "f"'] & \mathcal{C} \arrow[d, "m"] \\
\mathcal{Y} \arrow[r, "y"'] & \mathcal{A}
\end{tikzcd}$$

commutes, with $f : \mathcal{X} \to \mathcal{Y}$ is a morphism between experiments. We have to show that there exists some $y' : \mathcal{Y} \to \mathcal{A}$ such that

$$\begin{tikzcd}
\mathcal{X} \arrow[r, "x"] \arrow[d, "f"'] & \mathcal{C} \arrow[d, "m"] \\
\mathcal{Y} \arrow[r, "y"'] \arrow[ur, "y'"] & \mathcal{A}
\end{tikzcd}$$

commutes. It is sufficient to consider the cases in which $\mathcal{Y}$ is obtained from $\mathcal{X}$ by adding just a state q_{n+1} and a transition $t_{n+1} : q_n \xrightarrow{l} q_{n+1}$. On the prefix $\mathcal{X}$ of $\mathcal{Y}$, morphism y' is defined as morphism x (this assures that the upper triangle commutes). Now we define how transition t_{n+1} and state q_{n+1} are mapped into $\mathcal{C}$.
Suppose that $y_T(t_{n+1}) = t_a : q_a \xrightarrow{l} q'_a$ with $q_a = y_Q(q_n)$ and $q'_a = y_Q(q_{n+1})$. Then $x_Q(q_n) = \langle q_a, \delta, q_b\rangle \in \mathcal{R}$. Since $\mathcal{R}$ is a HD-bisimulation, there must be some transition t_b of $\mathcal{B}$ that matches t_a according to some ζ, i.e., there is some $t = \langle t_a, \zeta, t_b\rangle : \langle q_a, \delta, q_b\rangle \xrightarrow{l} q' = \langle q'_a, \delta', q'_b\rangle$ of $\mathcal{C}$. By definition of $\mathcal{C}$, we know that $m_T(t) = t_a$ and $m_Q(q') = q'_a$. Let us define $y'_T(t_{n+1}) = t$ and $y'_Q(q_{n+1}) = q'$. Then it is easy to show that also the lower triangle commutes. ☐

7.2.2 Application to HDS-automata

In the case of HD-automata with Symmetries, an experiment is a finite sequences of transitions and an extended experiment can be obtained by adding new transitions. Moreover, we require that no symmetries are defined on the names of states and transitions of the experiments: the intuition is that the observer can distinguish all the names.

Definition 7.11 (category of HDS-experiments) *A HDS-automaton $\mathcal{X}$ is a* HDS-experiment *if:*

- *$Q = \{q_0, q_1, \ldots, q_n\}$ are the states and $T = \{t_1, \ldots, t_n\}$ are the transitions, and $s(t_i) = q_{i-1}$ and $d(t_i) = q_i$;*
- *for each $q \in Q$, $\mathsf{Q}[q] = \{\mathrm{id}_{\mathcal{N}}\}$; similarly, for each $t \in T$, $\mathsf{T}[t] = \{\mathrm{id}_{\mathcal{N}}\}$.*

The category $\mathbf{HDS\text{-}Exp}$ of HD-experiments *is the full subcategory of* $\mathbf{HDS}$ *with HDS-experiments as objects.*
Category $\mathbf{HDS\text{-}Exp}_{\mathsf{L}}$ *is the full subcategory of* $\mathbf{HDS\text{-}Exp}$ *whose objects are* $\mathbf{HDS}_{\mathsf{L}}$*-automata.*

Theorem 7.12 *Two HDS-automata on the same labels* L *are HDS-bisimilar iff they are open-bisimilar w.r.t. experiments* $\mathbf{HDS\text{-}Exp}_{\mathsf{L}}$.

Proof (Sketch). We prove the two implications separately.
Proof of the "if" implication. This proof is similar to the proof of the "if" implication of Theorem 7.10.
It consists of showing that, if $m : \mathcal{A} \to \mathcal{B}$ in a $\mathbf{HD\text{-}Exp}$-open map, then $\mathcal{A}$ and $\mathcal{B}$ are HDS-bisimilar via HDS-bisimulation

$$\mathcal{R} \overset{\mathrm{def}}{=} \{\langle q_1, \delta, q_2\rangle \mid q_1 \text{ reachable}, q_2 = m_Q(q_1), \delta \in \mathsf{m}_{\mathsf{Q}}[q_1]\}.$$

We omit the details.
Proof of the "only if" implication. This proof is similar to the proof of the "only if" implication of Theorem 7.10.
It consists of showing that, if $\mathcal{A}$ and $\mathcal{B}$ are HD-bisimilar automata, then here exist a HDS-automaton $\mathcal{C}$ and a span of open maps $m : \mathcal{C} \to \mathcal{A}$ and $m' : \mathcal{C} \to \mathcal{B}$.
Suppose that $\mathcal{A}$ and $\mathcal{B}$ are HDS-bisimilar via the HDS-bisimulation $\mathcal{R}$. The HDS-automaton $\mathcal{C}$ and m and m' are defined as follows:

- $Q_{\mathcal{C}} = \mathcal{R}$; for each $q = \langle q_a, \delta, q_b\rangle \in \mathcal{R}$ we have $m_Q(q) = q_a$ and $m'_Q(q) = q_b$. Moreover $\mathsf{Q_c}[q] = \{\mathrm{id}_{\mathcal{N}}\}$, i.e., all the names of state q in $\mathcal{C}$ are distinguished. Finally, $\mathsf{m_Q}[q] = \{\rho \mid \rho \in \mathsf{Q}_{\mathcal{A}}[q_a]\}$ and $\mathsf{m}'_{\mathsf{Q}}[q] = \{\delta;\rho \mid \rho \in \mathsf{Q}_{\mathcal{A}}[q_a]\}$.
- $T_{\mathcal{C}} = \mathcal{S}$, where $\langle t_a, \zeta, t_b\rangle \in \mathcal{S}$ if t_a and t_b are matching transitions according to $\zeta : \mathcal{N} \longleftrightarrow \mathcal{N}$. For each $t \in T_{\mathcal{C}}$, we define $\mathsf{T}_{\mathcal{C}}[t_c] = \{\mathrm{id}_{\mathcal{N}}\}$. Embeddings $\mathsf{m_T}[t]$ and $\mathsf{m}'_{\mathsf{T}}[t]$ are defined as for the states.
- The starting state of $\mathcal{C}$ is $\langle q_{0\mathcal{A}}, (\sigma_{0\mathcal{A}}; \sigma_{0\mathcal{B}}^{-1}), q_{0\mathcal{B}}, \rangle \in \mathcal{R}$ for some $\sigma_{0\mathcal{A}} \in f_{0\mathcal{A}}$ and $\sigma_{0\mathcal{B}} \in f_{0\mathcal{B}}$.

We omit the proofs that m and m' are open maps. $\square$

7.2.3 Minimal HDS-automata

We have seen in Section 2.1 that minimal automata exist in each class of bisimilar automata. In Section 6.5 a similar result has been given for HDS-automata. Now we show that minimal automata and minimal HDS-automata can be defined also in the categorical setting.

Let $\mathbf{Aut}^r_\sim$ be the subcategory of $\mathbf{Aut}$ that is defined as follows:

- the objects of $\mathbf{Aut}^r_\sim$ are the *reachable automata*, i.e., all the state are reachable from the starting state via some sequence of transitions;
- the arrows of $\mathbf{Aut}^r_\sim$ are the open maps of $\mathbf{Aut}$.

Clearly, category $\mathbf{Aut}^r_\sim$ consists of different connected components, that correspond to the different classes of bisimilar automata.

Theorem 7.13 *Every connected component of category $\mathbf{Aut}^r_\sim$ has a terminal object.*[9]

Therefore, in each class of bisimilar automata there is a particular automaton that is terminal with respect to bisimulation morphisms. Standard categorical results guarantee that the terminal object is unique up to isomorphisms. So, Theorem 7.13 defines an unique canonical representative for each class of bisimilar automata. It is easy to prove that this canonical automaton coincides with the minimal automaton that we have defined explicitly in Section 2.1.

The same result also hold for HDS-automata. In fact, let $\mathbf{HDS}^r_\sim$ be the category of reachable HDS-automata and of the $\mathbf{HDS\text{-}Exp}$-open morphisms on them. Then the following theorem holds:

Theorem 7.14 *Every connected component of category $\mathbf{HDS}^r_\sim$ has a terminal object.*

Proof. Let $\mathcal{A}$ be a reachable HDS-automaton and let $\mathcal{A}_{\min}$ be the minimal HDS-automaton corresponding to $\mathcal{A}$ according to Definition 6.19.
Now we show that there exists one and only one open map $m : \mathcal{A} \to \mathcal{A}_{\min}$. This is sufficient to conclude the proof, since by Proposition 6.24 if $\mathcal{B} \sim \mathcal{A}$ then $\mathcal{A}_{\min}$ and $\mathcal{B}_{\min}$ are isomorphic, and hence there is one and only one open map also from $\mathcal{B}$ to $\mathcal{A}_{\min}$.
Let us define morphism $m : \mathcal{A} \to \mathcal{A}_{\min}$ as follows:

- if $q \in Q$ then $m_Q(q) = \lfloor q \rfloor \in Q_{\min}$; moreover $\mathsf{m_Q}[q] = \Delta_{\mathcal{A}}(q, \lfloor q \rfloor)$;
- if $t \in T$ then $m_T(t) = \lfloor t \rfloor \in t_{\min}$; moreover $\mathsf{m_T}[t] = \Delta_{\mathcal{A}}(t, \lfloor t \rfloor)$.

It is easy to check that m is a morphism between HDS-automata, and that it is an open maps.
Conversely, let $m' : \mathcal{A} \to \mathcal{A}_{\min}$ be an open map; we show that $m' = m$.
Let $q \in Q$ and let $q' = m'_Q(q)$; we remark that, by definition of $\mathcal{A}_{\min}$, $q' \in Q$.
It is easy to prove that, for each $\delta \in \mathsf{m}'_{\mathsf{Q}}[q]$, $\langle q, \delta, q'\rangle \in \mathcal{R}_{\mathcal{A};\mathcal{A}_{\min}}$. By definition of $\mathcal{A}_{\min}$, this implies that $\langle q, \delta, q'\rangle \in \mathcal{R}_{\mathcal{A}}$. Hence, $q \equiv q'$, which implies $q' \equiv \lfloor q \rfloor$ by definition of $\lfloor q \rfloor$, and this in turn implies $q' = \lfloor q \rfloor$ by definition of $\mathcal{A}_{\min}$. So, $m'_Q(q) = m_Q(q)$.
Let $\delta \in \mathsf{m}'_{\mathsf{Q}}[q]$. We have already seen that it holds $\langle q, \delta, q'\rangle \in \mathcal{R}_{\mathcal{A}}$, i.e., $\delta \in \Delta(q, q') = \mathsf{m_Q}[q]$. Let $\delta' \in \Delta(q, q') = \mathsf{m_Q}[q]$. Then $\delta^{-1}; \delta' \in \Delta(q', q')$ and, by definition of $\mathcal{A}_{\min}$, $\delta^{-1}; \delta' \in \mathsf{Q}_{\min}[q']$. Hence, $\delta' = \delta; (\delta^{-1}; \delta') \in \mathsf{m}'_{\mathsf{Q}}[q]$ by definition of embedding on symmetries. So, $\mathsf{m}'_{\mathsf{Q}}[q] = \mathsf{m_Q}[q]$ for any $q \in Q$.
This concludes the proof that $m_Q = m'_Q$. We omit the proof that $m_T = m'_T$, which is similar. $\square$

8 Possible extensions and other work

In this section we discuss some possible extensions of the approach described in the paper. In particular, we present some other examples of formalisms that could be mapped into HD-automata. We also discuss a verification environment that exploits HD-automata to verify history dependent formalisms.

[9] An object t of a (sub)category $\mathbf{C}$ is terminal if, for any other object x of $\mathbf{C}$ there exists exactly one arrow from x to t.

8.1 CCS with causality

In Section 5.1 we have presented the location semantics of CCS as a way to define a truly concurrent behavioral equivalence for this language. The *causal semantics* represents another possible approach to obtain this result: in this case, the causality relations between the actions of an agent are taken into account, as we did in the history-preserving semantics of Petri nets.

In agent $\alpha|\beta$ the two actions α and β are independent: in fact, the meaning of the parallel composition is that the two subagents are in two distinct computation threads. In agent $\alpha.\beta + \beta.\alpha$, instead, the two actions can occur in any order, however they are not independent: the second action is sequential to the first one and is enabled only by the occurrence of the first.

There are many ways to formalize this idea. For instance, in [DDNM90] CCS agents are mapped into Petri nets, so that the history preserving semantics of the nets can be used to give a causal semantics to the agents. In [DD89, Kie94] the causal dependencies between the actions of an agent are observed directly in the labels of the transition system. Here we follow the approach of [Kie94], which is similar to the location approach of Section 5.1. Also in this case we extend the language with prefixes of the form $c :: p$; here, however, c is a *cause* rather that a location. Transitions are of the form

$$p \xrightarrow[C,c]{\alpha} p'$$

where C is the set — not a sequence as in the location approach — of the causes that enabled action α and c represents the new name that will be used in the following transitions to refer to this action.

The main difference between the location and the causal approach is in the synchronization: consider the agent

$$p = (\nu\beta)\,(\alpha.\beta|\bar{\beta}.\gamma).$$

In the case of the location semantics we observe the computation

$$p \xrightarrow[l_1]{\alpha} (\nu\beta)\,(l_1 :: \beta|\bar{\beta}.\gamma) \xrightarrow{\tau} (\nu\beta)\,(l_1 :: \mathbf{0}|\gamma) \xrightarrow[l_2]{\gamma} (\nu\beta)\,(l_1 :: \mathbf{0}|l_2 :: \mathbf{0})$$

and in the case of the causal approach we observe:

$$p \xrightarrow[\emptyset,c_1]{\alpha} (\nu\beta)\,(c_1 :: \beta|\bar{\beta}.\gamma) \xrightarrow{\tau} (\nu\beta)\,(c_1 :: \mathbf{0}|c_1 :: \gamma) \xrightarrow[\{c_1\},c_2]{\gamma} (\nu\beta)\,(c_1 :: \mathbf{0}|c_1 :: c_2 :: \mathbf{0}).$$

Action γ depends on action α (there is no way to execute γ without first executing α), and this is represented by requiring that the synchronization transition on β extends also to the subagent γ the causes of β.

In general, a synchronization "mixes" the causes of the two complementary actions. As a consequence, the structure of the causes is not a tree, as it was for the structure of location paths; rather, it is a partial order, as it was in the case of the Petri nets. Hence, to map CCS with causality on Basic HD-automata we can use techniques similar to the ones for nets.

First of all, we define an alternative semantics where only the maximal causes that enable an action are observed in the transition. In this way, causes can be removed from an agent when there are no more processes which depend directly on them[10]. This "incremental" causal semantics for CCS is equivalent to the classical one, but allows for discarding unused causes. A mapping to Basic HD-automata can now be defined so that the obtained HD-automata are finite whenever the agents are finitary. Moreover, HD-bisimulation exactly matches causal bisimulation:

Theorem 8.1 *Let p_1 and p_2 be two CCS agents and let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the corresponding HD-automata according to the causal semantics. Then $p_1 \sim_{\mathrm{cau}} p_2$ iff $\mathcal{A}_1 \sim \mathcal{A}_2$.*

8.2 The late π-calculus semantics

In Section 3 we have presented the *ground* and the *early* semantics of the π-calculus. However, these are not the only semantics that has been proposed for this language. For instance, the semantics that was proposed from the very beginning for the π-calculus was *late* [MPW92]. The difference with respect to the early semantics is in the input transitions: in the early approach, the label of the transition carries the actual channel name which is received in the communication; in the late context, instead, the label carries just a "placeholder" for the received name and the instantiation takes place in a successive step[11]. Agent $p = a(x).p'$ performs the bound input transition

$$p \xrightarrow{a(x)} p'$$

[10]Notice that to remember the partial order of the causes it is necessary to add a new structure to the agents. In fact, it was possible to represent the tree structure of the location directly in the syntax of the agents; however, it is quite more complex to represent a partial order in this way.

[11]The names *early* and *late* refer to the time of the instantiation of the received name.

where name x is the placeholder. If the received name will be b, then the effective obtained agent will be $p'\{b/x\}$. This instantiation is performed in the definition of bisimulation, where a specific clause for input actions is present: if $\mathcal{R}$ is a bisimulation and $p \mathrel{\mathcal{R}} q$, then

whenever $p \xrightarrow{a(x)} p'$ and $x \notin \text{fn}(p|q)$, then $p \xrightarrow{a(x)} q'$ and $p'\{b/x\} \mathrel{\mathcal{R}} q'\{b/x\}$ for each name b.

We denote with $\sim_l$ the late bisimulation equivalence.

In [FMQ96] a different approach has been proposed for the late semantics. The input transition is split in two steps, the first which announces an input on a given channel, and the second which corresponds to the reception of the input value: so, for instance, the input of the agent $p = a(x).p'$ is modeled as follows:

$$p \xrightarrow{a} \lambda x.p' \xrightarrow{[b]} p'\{b/x\}$$

where b is the effectively received name. The two steps have to be considered atomic, i.e., when an agent perform the first step, the second step has to follow immediately.

Since the instantiation of the received name is performed in the transition system, no special clause for the input transition has to appear in the definition of bisimulation. In [FMQ96] it is proved that this approach coincides with the original late semantics of [MPW92].

We can exploit this idea also to capture the late semantics of the π-calculus within the framework of the HDS-automata. We have to replace the two labels $\mathtt{in}$ and $\mathtt{in}_2$, used for the early input actions, with the labels $\mathtt{in}_{\mathtt{start}}$ and $\mathtt{in}_{\mathtt{end}}$, corresponding to the two steps $\xrightarrow{a}$ and $\xrightarrow{[b]}$ respectively; one distinct name n_{sub} is associated to label $\mathtt{in}_{\mathtt{start}}$, whereas one distinct name n_{obj} is associated to label $\mathtt{in}_{\mathtt{end}}$. Also the definition of the mapping from π-calculus agents to HDS-automata has to be changed accordingly.

All the results we have presented for the early semantics can be restated for the late approach. Also in this case the HDS-automata corresponding to agent p is finite whenever agent p is finitary. Finally:

Theorem 8.2 *Let p and q be two π-calculus agents and let $\mathcal{A}_p^{\pi_l}$ and $\mathcal{A}_q^{\pi_l}$ be the corresponding HDS-automata, according to the late translation. Then $p \sim_l q$ iff $\mathcal{A}_p^{\pi_l} \sim \mathcal{A}_q^{\pi_l}$.*

8.3 Causality/localities and the π-calculus

In [San96a] a localities semantics is proposed for the π-calculus: the approach is very similar to the one described in Section 5.1 for CCS. [San96a] also shows that it is possible to “codify” the location semantics into the π-calculus: there exists a fully abstract mapping from the π-calculus with localities to the bare π-calculus. In [BS98] similar results are obtained for the π-calculus with causality. The usage of names in the π-calculus is, hence, general enough to represent the tree-like structure of localities and the partial order structure of causes. However, the encodings defined in [San96a] and in [BS98] do not preserve finitary agents: the obtained agents are not finitary whenever the starting agents can perform infinite computations.

HDS-automata can be exploited also to capture the localities and the causal semantics of π-calculus. In the case of the localities semantics, for instance, it is sufficient to combine the techniques described in Section 6.3.2 and in Section 5.2. The obtained HDS-automata have four infinite repositories; two of them are the repositories $\textcircled{\nu}$ and $\textcircled{u}$ that are necessary to deal with π-calculus names; the other two repositories are used as source and drain for the localities: they play the same role of repositories $\textcircled{\alpha}$ and $\textcircled{\omega}$ in Section 6.3.1.

The approaches of [San96a] and [BS98] and the approach based on HDS-automata that we have just sketched have different aims and use different techniques. The translations of [San96a] and [BS98] are syntactic (and hence they always terminate), while our translations generate an operational model for the agents (and hence they terminate only for certain classes of agents). On the other hand, the fact that the obtained HDS-automata are finite whenever the starting π-calculus agents are finitary shows that HDS-automata offer more freedom in handling names than π-calculus.

8.4 The weak semantics

In this paper we have presented the *strong* semantics of CCS, π-calculus and CCS with localities. These process calculi are equipped also with a *weak* semantics.

Differently from the strong approach, where τ actions are dealt with as other actions, in the weak approach they are considered internal actions which cannot be observed from the outside. Weak transitions are defined to this purpose:

- $\overset{\epsilon}{\Longrightarrow} \overset{\text{def}}{=} (\xrightarrow{\tau})^*$, and
- $\overset{\alpha}{\Longrightarrow} \overset{\text{def}}{=} \overset{\epsilon}{\Longrightarrow}\xrightarrow{\alpha}\overset{\epsilon}{\Longrightarrow}$ for $\alpha \neq \tau$.

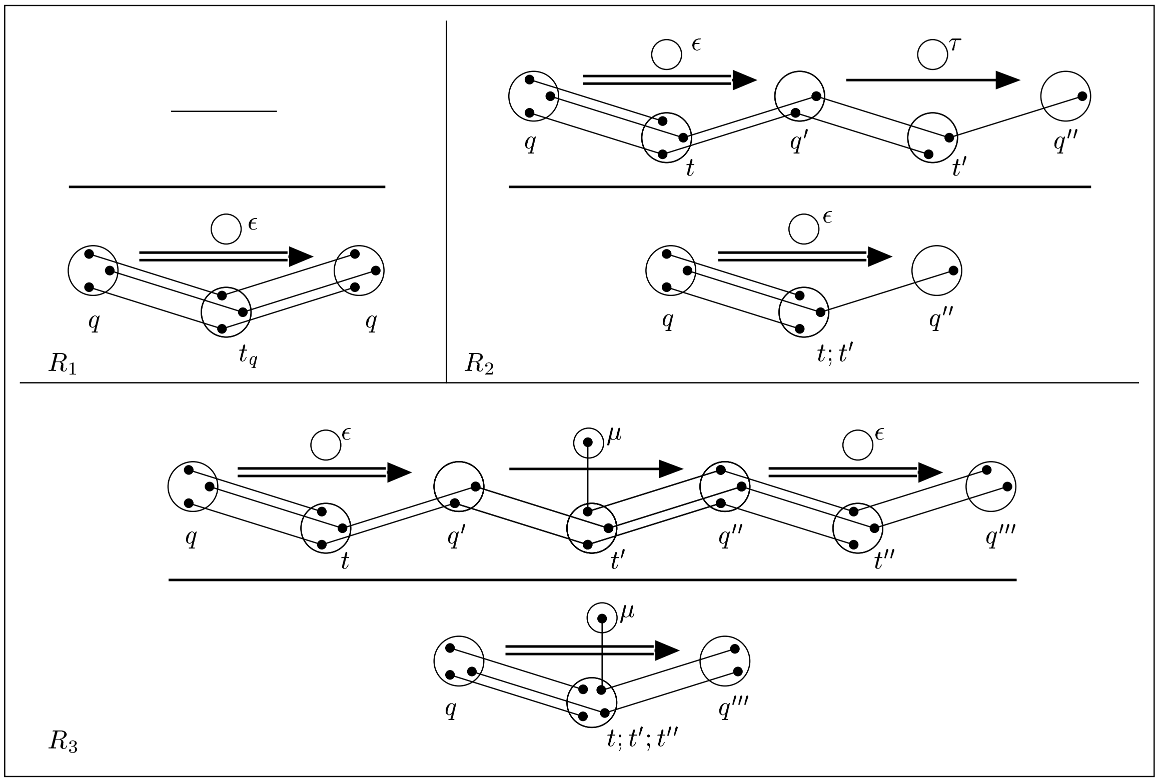

Figure 11: Weak transitions of HD-automata

Notice that $p \overset{\epsilon}{\Longrightarrow} p'$ means that p can evolve in p' by making just internal actions, i.e., by not communicating with the outside; in particular, we can have $p = p'$. Also, visible transitions can be preceded and followed by any unobserved internal computation.

The "weak" automaton can be built by first building the automaton according to the strong operational semantics and then by adding to it the weak transitions according to the rules above. A weak behavioral equivalence is obtained by exploiting the ordinary bisimulation on these weak automata. Whenever the strong automaton is finite, also the corresponding weak automaton is finite — but possibly much larger [PT87, KS90] — so the standard algorithms can be used also in this case.

Two different approaches are possible for building a "weak" HD-automata for the calculi considered in the paper. The first approach is simply to use the weak transitions of these calculi in the construction of the automaton. The second approach consists of building the strong HD-automaton and then of generating the weak transitions directly on the HD-automaton. A graphical representation of the rules for generating the weak transitions in the case of Basic HD-automata is given in Figure 11. The first two rules are for the ϵ transitions, that correspond to a sequence of τ transitions: rule R_1 is for the empty sequence of τ transitions, while rule R_2 shows how this sequence can be extended by adding a new τ transition. Rule R_3 is for the visible transitions: it shows that these can absorb sequences of τ transitions that precede and follow them. Similar rules exist for HD-automata with Symmetries.

It is possible to prove that equivalent "weak" HD-automata are obtained by using the weak transitions of the calculus, and by first building the "strong" HD-automaton and then generating the weak transitions with the rules in Figure 11.

8.5 A verification environment based on HD-automata

The *HD-Automata Laboratory* (HAL) is an integrated tool-set for the specification, verification and analysis of concurrent and distributed systems. The core of HAL are the HD-automata: here they are used as a common format for the various history-dependent languages. The HAL environment includes modules which implement decision procedures to calculate behavioral equivalences, and modules which support verification of behavioral properties expressed as formulae of suitable temporal logics. The environment has been successfully applied to the specification and verification of mobile processes defined as π-calculus agents. In particular, the verification of the handover protocol for mobile telephones [OP92] has been carried out within the HAL environment. Here we provide a short overview of the current implementation of the HAL environment. A fuller account of this and of other case studies may be found in [FFG$^+$97] and [GR97].

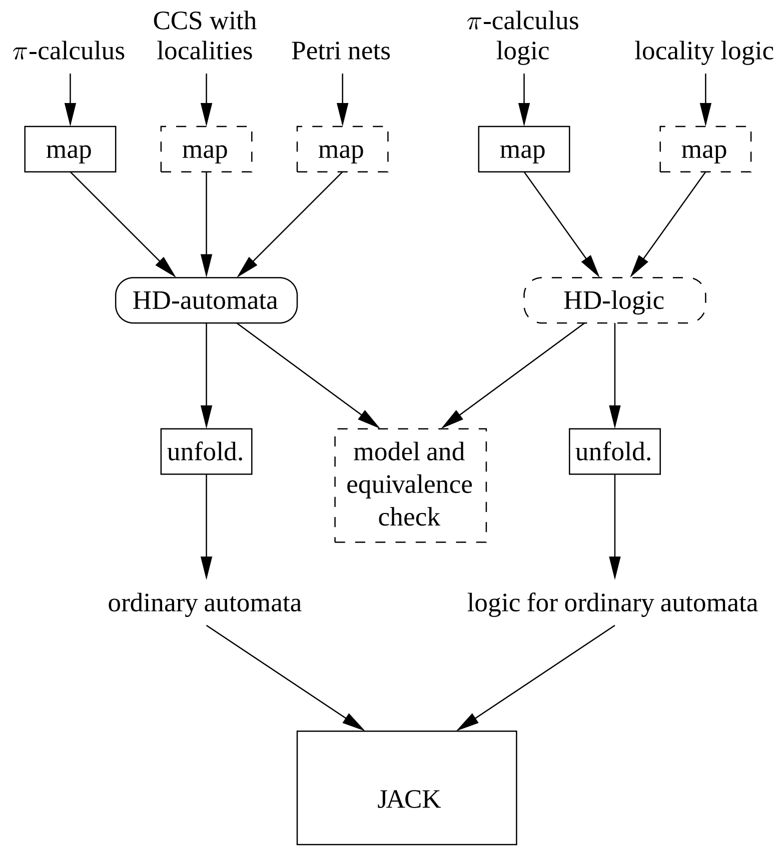

Figure 12: The HAL environment: an overview

The HAL environment allows π-calculus agents to be translated into ordinary automata, so that existing equivalence checkers can be used to calculate whether the π-calculus agents are bisimilar. The environment also supports verification of logical formulae expressing desired properties of the behavior of π-calculus agents. To this purpose, we found convenient to exploit a logic with modalities indexed by π-calculus actions, and to implement a translation of this π-logic into a logic for ordinary automata. Hence, existing model checkers can be used to verify whether or not a formula holds for a given π-calculus agent.

Figure 12 presents an overview of the HAL environment. The dashed boxes indicate work-in-progress, i.e., modules which are under development. In the current implementation the HAL environment consists essentially of five modules: three modules perform the translations from π-calculus agents to HD-automata, from HD-automata to ordinary automata, and from π-logic formulae to ordinary ACTL formulae. The fourth module provides routines that manipulate the HD-automata in order to reduce their size. The fifth module is basically the JACK system [BGL94] which works at the level of ordinary automata and performs the standard operations on them like behavioral verification and model checking. The JACK environment combines different specification and verification tools, independently developed, around a common format for representing ordinary automata: the FC2 file format [BRRdS96]. The HAL environment supports also a textual user interface to invoke the commands in the modules of the system and graphical user interface, that allows for a user-friendly invocation of the basic functionalities of HAL.

8.6 HD-automata with negative transitions

Basic HD-automata and HD-automata with Symmetries are not suitable for the open semantics [San96b] and the asynchronous semantics [HT91, ACS98] of π-calculus. These definitions of bisimulation are quite different from the classical one. When two agents p and q are bi-simulated, not all the transitions of p are matched by corresponding transitions of q (and vice-versa). Some of the transitions of p are *redundant*, i.e., they are covered by more general transition of p itself, and q is required to match only the *irredundant* transitions of p. (See [PS01] and [MP99] for a detailed description of the open and asynchronous bisimulation, respectively.)

In [Pis99, MP99] HD-automata with Negative transitions (HDN-automata) have been defined. They extend Basic HD-automata by allowing for defining *irredundancy* of transitions in a quite general way. In particular they are able to capture the open and asynchronous semantics of π-calculus. An interesting direction of further investigation is the integration of negative transitions and symmetries. This combination will open the possibility of having minimal realizations also for HDN-automata.

8.7 A coalgebraic definition of HD-automata

In [MP00] a variant of HD-automata with Symmetries is defined in a coalgebraic setting. In that paper, the transition system of π-calculus is modeled as a coalgebra on a category of name permutation algebras. The role of permutations in that context is very similar to that of permutations in HDS-automata, namely, they allow for an explicit representation of the distinguished names of the agents, and of the generation of fresh names during the evolutions of the agents. The classical results of the coalgebraic theory for ordinary transition systems [Rut00], and notably the existence of a final coalgebra, apply also to this extended case.

In [MP00] the coalgebraic semantics is also linked to HD-automata. It is shown that (a variant of) HD-automata with Symmetries are in bijective correspondence, up to isomorphism, with the coalgebraic transition systems built on the top of the permutation algebra, and that minimal HD-automata correspond to minimal transition systems. Therefore, HDS-automata can be seen as a concrete and compact representation of coalgebras built on the top of permutation algebras, precisely in the same way as ordinary automata can be seen as the concrete representation of standard coalgebras [Rut00]. This result is particularly interesting, as it shows that the definitions of HDS-automata and of HDS-bisimulation given in this paper are not arbitrary: rather, they derive naturally from the more primitive concept of coalgebras on permutation algebras.

9 Concluding remarks

We have presented History-Dependent Automata and we have shown that they are an operational model particularly adequate for history-dependent calculi. We have considered different kinds of history-dependent calculi, that implement different phenomena of concurrent systems: these phenomena include mobility (π-calculus), locality (CCS with localities), and causality (Petri nets).

We have defined two variants of HD-automata. Basic HD-automata are, as their name suggests, the simplified version: they are automata whose states, transitions, and labels are enriched with sets of local names. Basic HD-automata are well suited to model the creation of fresh names in the evolution of a system. CCS with localities, Petri nets and the ground π-calculus are mapped into this family of HD-automata. In HDS-automata, symmetries among the names are added to the model. These symmetries have been particularly useful for the representation of history-dependent calculi with more sources of fresh names: the most important example is the (early and late) π-calculus. The two families of HD-automata come equipped with a bisimulation-based observational semantics. HD-bisimulation on HD-automata corresponds to the ordinary bisimulation semantics for history-dependent calculi.

An important property that holds only for HDS-automata is the existence, in each class of equivalent HD-automata, of a minimal representative. As it happens for ordinary automata, this minimal HDS-automaton can be considered the semantic object corresponding to the class of equivalent HDS-automata.

We have also revisited the definitions of HD-automata and of HD-bisimulation in a categorical framework. Classical categorical definitions of automata and of bisimulations are extended to deal with Basic HD-automata and with HDS-automata. The possibility of using uniform techniques for defining HD-automata and HD-bisimulation enforces our confidence in the approach.

HD-automata provide the core of HAL [FFG$^+$97, GR97], a verification environment for concurrent systems described in history-dependent formalism: 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 the history-dependent calculi.

The approach proposed in this paper can be extended in different directions. Clearly, new history-dependent formalisms can be represented by means of HD-automata. Particularly interesting would be the application of HD-automata to the join calculus [FGL$^+$96, BFL98], the fusion calculus [PV98, Vic98], and the spi calculus [AG99, AG98]; the existing definitions of bisimulations for these calculi are very specific, and HD-automata could allow for describing then in a clean and uniform way.

More interesting is the extension of the approach from HD-automata to different classes of enriched automata. Enriched automata and enriched bisimulation can be defined in a parametric way with respect to the information that enriches the automata. It would be useful to see if the approach can be applied, for instance, to symbolic transition graphs [HL95], or to concurrent constraint programming [Sar93].

Another interesting research direction consists of extending to HD-automata the meta-theory of process calculi based on formats [DS85, GV92, BIM95]. This meta-theory is advantageous from a theoretical point of view — it allows the development of theories that can be applied to different languages — and for a practical point of view — it would be possible to implement tools that work for all the languages defined in these formats. While some extensions of the theory of formats to richer families of calculi have been already appeared in the literature [WB96, Ber98], we feel that the idea of multiple repositories of names introduced in Section 6 could be particularly interesting for the development of formats for π-calculus-like calculi.

References

[Ace94] L. Aceto. A static view of localities. *Formal Aspects of Computing*, 6(2):201–222, 1994.

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

[AG98] M. Abadi and A. D. Gordon. A bisimulation method for cryptographic protocols in the spi calculus. In *Proc. ESOP'98*, volume 1381 of *LNCS*. Springer Verlag, 1998.

[AG99] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 148:1–70, 1999.

[BB92] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, 1992.

[BCHK93] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. *Theoretical Computer Science*, 114(1):31–61, 1993.

[BDKP91] E. Best, R. Devillers, A. Kiehn, and L. Pomello. Concurrent bisimulations in Petri nets. *Acta Informatica*, 28(3):231–264, 1991.

[Ber98] K. Bernstein. A congruence theorem for structured operational semantics of higher-order languages. In *Proc. LICS'98*, 1998.

[BFL98] M. Boreale, C. Fournet, and C. Laneve. Bisimulations for the join-calculus. In *Proc. PROCOMET'98*. Chapman & Hall, 1998.

[BGL94] A. Bouali, S. Gnesi, and S. Larosa. The integration project for the JACK environment. *Bullettin of the EATCS*, 54, 1994.

[BIM95] B. Bloom, S. Istrail, and A. R. Meyer. Bisimulation can't be traced. *Journal of ACM*, 42:232–268, 1995.

[BRRdS96] A. Bouali, A. Ressouche, V. Roy, and R. de Simone. The FC2Tools set. In *Proc. CAV'96*, volume 1102 of *LNCS*. Springer Verlag, 1996.

[BS98] M. Boreale and D. Sangiorgi. A fully abstract semantics for causality in the π-calculus. *Acta Informatica*, 35, 1998.

[Cas87] I. Castellani. Bisimulation and abstraction homomorphisms. *Journal of Computer and System Sciences*, 34(2/3):210–235, 1987.

[Cas93] I. Castellani. Observing distribution in processes. In *Proc. MFCS'93*, volume 711 of *LNCS*. Springer Verlag, 1993.

[CN94] F. Corradinio and R. De Nicola. Distribution and locality of concurrent systems. In *Proc. ICALP'94*, volume 920 of *LNCS*. Springer Verlag, 1994.

[CSW97] G. L. Cattani, I. Stark, and G. Winskel. Presheaf models for the pi-calculus. In *Proc. CTCS'97*, volume 1290 of *LNCS*. Springer Verlag, 1997.

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

[DD89] Ph. Darondeau and P. Degano. Causal trees. In *Proc. ICALP'89*, volume 372 of *LNCS*. Springer Verlag, 1989.

[DDNM90] P. Degano, R. De Nicola, and U. Montanari. A partial ordering sematics for CCS. *Theoretical Computer Science*, 75:223–262, 1990.

[DS85] R. De Simone. Higher level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science*, 37(3):245–267, 1985.

[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.

[FGL$^+$96] C. Fournet, G. Gonthier, J.-J. Lévy, L. Maranget, and D. Rémy. A calculus of mobile agents. In *Proc. CONCUR'96*, volume 1119 of *LNCS*. Springer Verlag, 1996.

[FM90] G. Ferrari and U. Montanari. Towards the unification of models for concurrency. In *Proc. CAAP'90*, volume 431 of *LNCS*. Springer Verlag, 1990.

[FMM97] G. Ferrari, U. Montanari, and M. Mowbray. Structured transition systems with parametric observations: observational congruences and minimal realizations. *Mathematical Structures in Computer Science*, 7:1–42, 1997.

[FMQ96] G. Ferrari, U. Montanari, and P. Quaglia. A π-calculus with explicit substitutions. *Theoretical Computer Science*, 168(1):53–103, 1996.

[FMS96] M. Fiore, E. Moggi, and D. Sangiorgi. A fully-abstract model for the π-calculus. In *Proc. LICS'96*. IEEE, Computer Society Press, 1996.

[GR83] U. Goltz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57(2/3):125–147, 1983.

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

[GV92] J. F. Groote and F. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[HL95] M. Hennessy and H. Lin. Symbolic bisimulation. *Theoretical Computer Science*, 138:353–389, 1995.

[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.

[JNW96] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

[Kie94] A. Kiehn. Comparing locality and causality based equivalences. *Acta Informatica*, 31(8):697–718, 1994.

[KS90] P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

[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.

[MP95] U. Montanari and M. Pistore. Checking bisimilarity for finitary π-calculus. In *Proc. CONCUR'95*, volume 962 of *LNCS*. Springer Verlag, 1995.

[MP97a] U. Montanari and M. Pistore. History dependent verification for partial order systems. In *Partial Order Methods in Verification*, volume 29 of *DIMACS Series*. American Mathematical Society, 1997.

[MP97b] U. Montanari and M. Pistore. Minimal transition systems for history-preserving bisimulation. In *Proc. STACS'97*, volume 1200 of *LNCS*. Springer Verlag, 1997.

[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.

[MPY96] U. Montanari, M. Pistore, and D. Yankelevich. Efficient minimization up to location equivalence. In *Proc. ESOP'96*, volume 1058 of *LNCS*. Springer Verlag, 1996.

[MS89] U. Montanari and M. Sgamma. Canonical representatives for observational equivalence classes. In *Resolution Of Equations In Algebraic Structures*, volume 1: Algebraic Techniques. Academic Press, 1989.

[OP92] F. Orava and J. Parrow. An algebraic verification of a mobile network. *Formal Aspects of Computing*, 4(5):497–543, 1992.

[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. Available at `http://www.di.unipi.it/phd/tesi/tesi_1999/TD-5-99.ps.gz`.

[PS01] M. Pistore and D. Sangiorgi. A partition refinement algorithm for the π-calculus. *Information and Computation*, 164(2):264–321, 2001.

[PT87] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, 1987.

[PV98] J. Parrow and B. Victor. The fusion calculus: Expressiveness abd symmetry in mobile processes. In *Proc. LICS'98*, 1998. To appear.

[RT88] A. Rabinovich and B. A. Trakhtenbrot. Behaviour structures and nets. *Fundamenta Informaticae*, 11(4):357–404, 1988.

[Rut00] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[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.

[San96a] D. Sangiorgi. Locality and interleaving semantics in calculi for mobile processes. *Theoretical Computer Science*, 155(1):39–83, 1996.

[San96b] D. Sangiorgi. A theory of bisimulation for π-calculus. *Acta Informatica*, 33:69–97, 1996.

[Sar93] V. A. Saraswat. *Concurrent Constraint Programming*. MIT Press, 1993.

[Sta96] I. Stark. A fully abstract domain model for the pi-calculus. In *Proc. LICS'96*. IEEE, Computer Society Press, 1996.

[vB84] J. van Bentham. Correspondence theory. In *Handbook of Philosophical Logic*, volume II. Reidel, 1984.

[Vic98] B. Victor. *The Fusion Calculus: Expressiveness and Symmetry in Mobile Processes*. PhD thesis, Department of Computer Systems, Uppsala University, 1998.

[VJ85] R. Valk and M. Jantzen. The residue vector sets with applications to decidability problems in Petri nets. *Acta Informatica*, 21:643–674, 1985.

[Vog95] W. Vogler. Generalized OM-bisimulation. *Information and Computation*, 118:38–47, 1995.

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

[WB96] S. Weber and B. Bloom. Metatheory of the π-calculus. Technical Report TR96-1564, Cornell University, 1996.