Università di Pisa

Dipartimento di Informatica

Technical Report : TR-98-11

History-Dependent Automata

Ugo Montanari, Marco Pistore

October 5, 1998

ADDR: Corso Italia 40,56125 Pisa,Italy. TEL: +39-50-887111. FAX: +39-50-887226

History-Dependent Automata

Ugo Montanari and Marco Pistore

Computer Science Department, University of Pisa
Corso Italia 40, 56100 Pisa, Italy
{ugo,pistore}@di.unipi.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, both in a set-theoretical way (that is suitable for automatic verification in the case of finite HD-automata) and in a categorical way (by exploiting open maps). HD-bisimulation captures the classical definitions of bisimulation on the considered history dependent formalisms.

Keywords: process calculi, automata, bisimulation, verification, π-calculus, location semantics, history preserving bisimulation, open maps.

Contents

1 Introduction

In the context of process calculi (e.g., Milner's CCS [28]), *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 [28, 38]: in fact they take into account just 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 [23, 27] 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 observations occurring in previous transitions of the same agent, expressing in this manner a dependence from them. For instance, in the case of CCS with localities [7], 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 [14, 13, 26].

Another quite interesting example is π-calculus [30, 29]. 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 [45]) and name sending thus models higher order communication [41]. 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. This is not completely adequate to deal with the peculiar features of the calculus and complications occur in the representation of 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 hence needed in the definition of bisimulation, which is not the standard one for transition systems, 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 permits to adequately represent 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: we present a similar mapping for CCS with localities [7] and for CCS with causality [14, 13, 26]. This shows that our definition of HD-automata — and of HD-bisimulation — is adequate to describe interesting cases of history-dependent calculi. 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 [8].

It is also possible to associate ordinary automata to the HD-automata, in such a way that bisimilar HD-automata are mapped into bisimilar automata, and finite HD-automata are mapped into finite automata. The ordinary automata obtained in this way are hence much more convenient than those corresponding to the ordinary operational semantics of the history-dependent calculi, since they are finite for all the finitary agents and since the classical bisimulation can be applied on them. The possibility to apply the classical algorithms on ordinary automata also to the HD-automata has allowed to develop a verification environment [17] which provides equivalence and model-checking for the history-dependent formalisms described above. The classical algorithms of [37, 25] can be exploited here to check bisimilarity. At the moment, this approach works only for a sub-class of HD-automata which is sufficient to represent all the finitary CCS agents with locations and with causality and all the finitary Petri nets, but which can be only used for the finitary π-calculus without the matching operator.

Finally, we present a categorical approach to HD-automata. A classical categorical definitions 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 [24] is an uniform approach to define equivalences for concurrent models presented in a categorical framework. We show that this approach can be applied also to the HD-automata, thus obtaining a categorical definition of HD-bisimulation. It is important to remark that in [24] open maps are applied to Petri nets; the captured bisimulation, however, is not the standard history-preserving bisimulation, but a variant called *strong* history preserving bisimulation. As far as we know, the approach of HD-automata is the first that captures standard history-preserving bisimulation via open maps.

The possibility of defining HD-automata and HD-bisimulation via uniform techniques enforces our confidence in the approach. Furthermore the categorical definition of HD-bisimulation is more elegant than the ordinary one, since it avoids the case analysis on the rôle of names that appears in the ordinary definition.

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 Section 4 HD-automata are introduced and the translation from π-calculus to HD-automata is defined. In Section 5 bisimulation on history-dependent automata is introduced. In Sections 6 we briefly describe the other history-dependent formalisms we have considered — namely CCS with localities and Petri nets with history-preserving semantics — and their translations on HD-automata. In Section 7 the categorical characterizations of HD-automata and of HD-bisimulations are presented. In Section 8 it is sketched how ordinary automata can be associated to HD-automata. In Section 9 we describe in short some other formalisms that can be captured by HD-automata and some possible extensions, and 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 [32] 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 [35], by Daniel Yankelevich and the authors, for the CCS with localities, in [34] for Petri nets, and in [33] for a class of partial-order systems, that includes CCS with localities and Petri nets. HD-automata and HD-bisimulations defined in [35, 34, 33] are much simpler than those needed for π-calculus, since there is no input of names.

Also, the categorical presentation of HD-automata and HD-bisimulation is new, as well as the study of general properties of HD-automata described in Section 5.

2 Results on ordinary automata

Automata are a very convenient operational model for process calculi like CCS and CSP. 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* A is defined by:

- a set L of *labels*;
- a set Q of *states* and 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 $l : T \to L$ which associates a label to each transition;
- an *initial state* $q_0 \in Q$.

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

Notation To represent the components of an automaton we will use the name of the automaton as subscript; so, for instance, Q_B are the states of automaton B and d_B is its destination function. In the case of automaton A_x, we will simply write Q_x and d_x rather than Q_{A_x} and d_{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 instead of automata as operational models in concurrency. The difference is that in a labelled transition system there cannot be two different transitions which have the same sources, labels and destinations, i.e., the function $\langle s, l, d \rangle : T \to Q \times L \times Q$ is injective. In this case the transitions can be equivalently defined as a subset of $Q \times L \times Q$, with functions s, l and d defined in the obvious way. Here, we use automata; however, all the results could be reformulated for labelled transition systems.

2.2 CCS

The version of CCS we present here is slightly different from the classical one [28] and follows some suggestion 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 which appear in this paper, namely CCS, CCS with localities and π-calculus.

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

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

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

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

We use $a, b, \ldots$ to range over *visible actions*, i.e., over actions different from τ. For each agent identifier A there is a definition $A \stackrel{\text{def}}{=} p_A$ 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 process 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 $p\setminus\alpha$. Finally, infinite behaviors are obtained by means of agent identifiers and of their definitions; also in this case, we prefer this solution to the **rec** $x.p$ construct for analogy with the π-calculus. Of course, the set of definitions has to be finite, to avoid agents with an "infinite program".

We now introduce a *structural congruence* in the style of the Chemical Abstract Machine [4] and of the π-calculus [29]. This structural congruence allows us to identify all the agents which represent essentially the same system and which differ just for syntactical details; moreover it simplifies the presentation of the operational semantics. 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) $p{+}\mathbf{0} \equiv p \quad p{+}q \equiv q{+}p \quad p{+}(q{+}r) \equiv (p{+}q){+}r$
(par) $p|\mathbf{0} \equiv p \quad p|q \equiv q|p \quad p|(q|r) \equiv (p|q)|r$
(res) $(\nu\alpha)\,\mathbf{0} \equiv \mathbf{0} \quad (\nu\alpha)\,(\nu\beta)\,p \equiv (\nu\beta)\,(\nu\alpha)\,p \quad (\nu\alpha)\,(p|q) \equiv p|(\nu\alpha)\,q$ if α does not appear in p

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

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) whereas 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. These sequential processes are then connected by means of the operators of parallel composition and restriction, that allow to describe the structure of the system in which the processes act.

The transitions that CCS agents can perform are defined by the axiom schemata and inference rules of Table 1.

2.3 Representing CCS agents as automata

It is easy to associate an ordinary automaton to a CCS agent:

Definition 2.2 (from CCS agents to ordinary automata) The automaton A_p corresponding to the CCS agent p is defined as follows:

- the set of the labels is given by all CCS actions;

[PREF] $\mu.p \xrightarrow{\mu} p$	[SUM] $\dfrac{p_1 \xrightarrow{\mu} p'}{p_1+p_2 \xrightarrow{\mu} p'}$
[PAR] $\dfrac{p_1 \xrightarrow{\mu} p_1'}{p_1 \vert p_2 \xrightarrow{\mu} p_1' \vert p_2}$	[COMM] $\dfrac{p_1 \xrightarrow{\bar{\alpha}} p_1' \quad p_2 \xrightarrow{\alpha} p_2'}{p_1 \vert p_2 \xrightarrow{\tau} p_1' \vert p_2'}$
[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 $A \stackrel{\text{def}}{=} p_A$

Table 1: Operational semantics for CCS.

- $p \in Q$ 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 $l(t) = \mu$.

An important result of this translation is that finite-state automata correspond to all the agents that cannot activate an unbounded number of sequential components — and hence of computation threads — during their evolution.

Definition 2.3 (finitary agents) The *degree of parallelism* $\mathtt{deg}(p)$ of an agent p is defined as

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

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

Proposition 2.4 *Let p be a finitary CCS agent. Then the automaton A_p is finite*[1]*.*

A syntactical condition which implies that an agent is finitary is the absence of parallel compositions in the bodies of recursive definitions. However, there are many finitary agents, like $B \stackrel{\text{def}}{=} (\nu\delta)\, (a.(b.\delta.d.B|c.\bar{\delta}.\mathbf{0}))$, which do not satisfy this condition.

The results of Proposition 2.4 is important, since in many cases, like most of the protocols, the involved agents are in fact finitary. Hence, automata are interesting for representing the operational behavior of concurrent languages like CCS also for an algorithmic point of view.

2.4 Bisimulation on automata

Various notions of behavioral preorders and equivalences have been defined on automata. The most important equivalence is the *bisimulation equivalence* [38, 28].

Definition 2.5 (bisimulation on automata) Let A_1 and 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 A_1 and A_2 if $q_1 \; \mathcal{R} \; q_2$ implies:

for all transitions $t_1 : q_1 \xrightarrow{\lambda} q_1'$ of A_1 there is some transition $t_2 : q_2 \xrightarrow{\lambda} q_2'$ of A_2 such that $q_1' \; \mathcal{R} \; q_2'$.

A relation $\mathcal{R} \subseteq Q_1 \times Q_2$ is a *bisimulation* for A_1 and A_2 if both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations.
Two automata A_1 and A_2 on the same set of labels are *bisimilar*, written $A_1 \sim A_2$, if there is some bisimulation $\mathcal{R}$ for A_1 and A_2 such that $q_{01} \; \mathcal{R} \; q_{02}$.

By applying this definition to CCS agents, the classical notion of bisimulation for CCS is obtained. In this case, it is straightforward that two CCS agents are bisimilar if and only if the corresponding automata are bisimilar.

[1]To obtain this result, the structural axioms introduced in Section 2.2 are important, since they allow for garbage collecting of terminated components and unused restrictions.

2.5 Minimal automaton

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 case of finite-state automata, there are also efficient algorithms that minimize automata [37, 25]. These results directly 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. Moreover, in the case of finitary agents, the minimal automaton can be effectively built, with evident advantages in the case the agent represents, for instance, the control part of an hardware component. Minimal automata are also convenient for model checking CCS agents: usually, a large set of properties has to be checked for the same agent; it is hence convenient to minimize the agent first, and then to verify the properties on the minimal automaton.

3 The π-calculus

The π-calculus [30, 29] is 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 permits 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 [45], and also higher-order communications can be easily encoded in the π-calculus [41], thus allowing for code migration.

Many versions of π-calculus have appeared in the literature. The π-calculus we present here is *early* and *monadic*; it was first introduced in [31], but we present a slightly simplified version, following in part the style proposed in [41, 29] for the polyadic π-calculus.

Let $\mathfrak{N}$ be an infinite, denumerable set of *names*, ranged over by $a, \ldots, z$, and let *Var* be a finite set of *agent identifiers*, denoted by $A, B, \ldots$; the π-calculus *agents*, ranged over by $p, q, \ldots$, are defined by the syntax:

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

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

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

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

If $\sigma : \mathfrak{N} \to \mathfrak{N}$, we denote with $p\sigma$ the agent p whose free names have been replaced according to substitution σ (possibly with changes in the bound names); we denote with $\{y_1/x_1 \cdots y_n/x_n\}$ the substitution that maps x_i into y_i for $i = 1, \ldots, n$ and which is the identity on the other names.

Notice that, with some abuse of notation, we can see substitution σ in $p\sigma$ as a function on $\mathtt{fn}(p)$ rather than on $\mathfrak{N}$; in fact, $p\sigma$ and $p\sigma'$ coincide whenever σ and σ' coincide on $\mathtt{fn}(p)$. Therefore, we say that substitution σ is injective for p if $\sigma : \mathtt{fn}(p) \to \mathfrak{N}$ is an injective function. We also say that agents p and q differ for a bijective substitution if there exists some bijective function $\sigma : \mathtt{fn}(p) \to \mathtt{fn}(q)$ such that $q = p\sigma$.

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

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

$$[\textsc{tau}]\ \tau.p \xrightarrow{\tau} p \qquad [\textsc{out}]\ \bar{x}y.p \xrightarrow{\bar{x}y} p \qquad [\textsc{in}]\ x(y).p \xrightarrow{xz} p\{z/y\}$$

$$[\textsc{sum}]\ \frac{p_1 \xrightarrow{\alpha} p'}{p_1{+}p_2 \xrightarrow{\alpha} p'} \qquad [\textsc{par}]\ \frac{p_1 \xrightarrow{\alpha} p_1'}{p_1|p_2 \xrightarrow{\alpha} p_1'|p_2} \text{ if } \mathtt{bn}(\alpha) \cap \mathtt{fn}(p_2) = \emptyset$$

$$[\textsc{comm}]\ \frac{p_1 \xrightarrow{\bar{x}y} p_1' \quad p_2 \xrightarrow{xy} p_2'}{p_1|p_2 \xrightarrow{\tau} p_1'|p_2'} \qquad [\textsc{close}]\ \frac{p_1 \xrightarrow{\bar{x}(y)} p_1' \quad p_2 \xrightarrow{xy} p_2'}{p_1|p_2 \xrightarrow{\tau} (\nu y)\,(p_1'|p_2')} \text{ if } y \notin \mathtt{fn}(p_2)$$

$$[\textsc{res}]\ \frac{p \xrightarrow{\alpha} p'}{(\nu x)\,p \xrightarrow{\alpha} (\nu x)\,p'} \text{ if } x \notin \mathtt{n}(\alpha) \qquad [\textsc{open}]\ \frac{p \xrightarrow{\bar{x}y} p'}{(\nu y)\,p \xrightarrow{\bar{x}(z)} p'\{z/y\}} \text{ if } x \neq y, z \notin \mathtt{fn}((\nu y)\,p)$$

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

Table 2: Early operational semantics.

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

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

and are called respectively *synchronization*, *input*, *free output* and *bound output* actions; x and y are free names of α ($\mathtt{fn}(\alpha)$), whereas z is a bound name ($\mathtt{bn}(\alpha)$); moreover $\mathtt{n}(\alpha) = \mathtt{fn}(\alpha) \cup \mathtt{bn}(\alpha)$. Name x is called the *subject* and y or z the *object* of the action.

The transitions for the *early operational semantics* are defined by the axiom schemata and the inference rules of Table 2.

Some comments on the syntax and on the operational semantics of π-calculus are now in order. The syntax of π-calculus is similar to that of CCS: the most important difference is in the prefixes. The *output* prefix $\bar{x}y.p$ specifies not just the channel x for the communication, but also the value y that is sent on x; in the input prefixes $x(y).p$, name x represents still the channel, whereas y represents a formal variable in p, that is instantiated by the effectively received value when the input transition takes place (this behaviour is represented in rule [IN], that holds for each name z).

The matching $[x{=}y]p$ represents a guard for agent p: agent p can act only if x and y coincide; this behavior is obtained by exploiting the structural congruence: in fact $[x{=}x]p \equiv p$; no transition can be derived from $[x{=}y]p$ if $x \neq y$.

Notice that, in the case of the π-calculus, the actions a process can perform are different from the prefixes. This happens due to the input and to the bound output. In the case of the input, the prefix has the form $x(y)$, while the action has the form xz; in fact, y represent a formal variable, whereas z is the effectively received value[2]. The bound output transitions 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.

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 $p \mathrel{\mathcal{R}} q$ then:

for each $p \xrightarrow{\alpha} p'$ with $\mathtt{bn}(\alpha) \cap \mathtt{fn}(p, q) = \emptyset$ there is some $q \xrightarrow{\alpha} q'$ such that $p' \mathrel{\mathcal{R}} q'$.

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 $p \sim_\pi q$, if $p \mathrel{\mathcal{R}} q$ for some early bisimulation $\mathcal{R}$.

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

[2]This is not true in all the versions of the π-calculus; in the case of the *late* and *open* versions, for instance, also the input actions have formal variables rather than values. Further comments on these alternative semantics appear in Section 9.1.

can perform an infinite number of different input transitions $p \stackrel{xw}{\rightarrow} \bar{w}z.\mathbf{0}$, corresponding to all the possible choices of $w \in \mathfrak{N}$. 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 modelled by using all the names which are not already in use to represent it. As a consequence, the definition of bisimulation is not the ordinary one: in general two bisimilar process can have different sets free names, and the clause "$\mathtt{bn}(\alpha) \cap \mathtt{fn}(p,q) = \emptyset$" has to be added in Definition 3.1 to deal with the bound output transitions that correspond to a name which is used only in one of the two processes. This clause makes it difficult to reuse standard theory and algorithms for bisimulation on the π-calculus — see for instance [12].

4 History-dependent automata

Ordinary automata are insufficient to deal with history-dependent calculi. To address this problem, in this section we describe a richer structure, the *history-dependent automata* (*HD-automata* in brief), which are obtained by allowing names to appear explicitly in states, transitions and labels. As we will see, it is convenient to assume that the names which appear in a state, a transition or a label of a HD-automaton are *local* names and do not have a global identity. In this way, for instance, a single state of the HD-automaton can be used to represent all the states of a system that differ just for a bijective renaming. In this way, however, each transition is required to represent explicitly the correspondences between the names of source, target and label.

In an ordinary automaton states, transitions and labels are sets. In a HD-automaton they are *named sets*. 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 is an injective map from the names of f to the names of e; that is, the names of the target element f are embedded into the names of the source element e. In a HD-automaton a source state, a target state and label are associated to each transition via three functions on named sets; these functions embed the names of the source, of the target and of the label into the names of the transition, thus describing their correspondences (see also Figure 1).

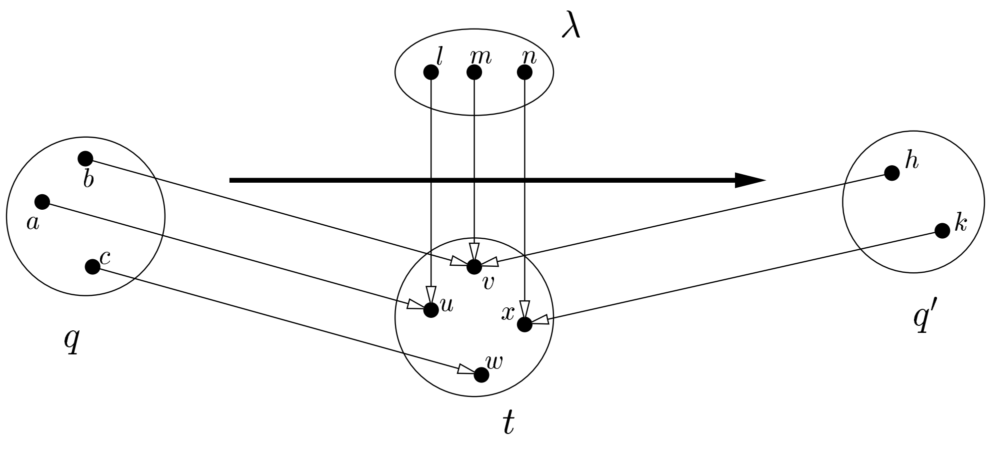

Figure 1: Example of a transition $t : q \stackrel{\lambda}{\rightarrow} q'$ of a HD-automaton.

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 We represent with $f : A \leftharpoondown\!\!\rightharpoonup B$ a *partial bijection* from set A to set B, i.e., $f \subseteq A \times B$ such that if $\langle a, b\rangle, \langle a', b'\rangle \in f$ then $a = a'$ iff $b = b'$.
We represent with $f : A \hookrightarrow B$ an *injection* from set A to set B, i.e., $f \subseteq A \times B$ such that for each $a \in A$ there exists exactly one $b \in B$ such that $\langle a, b\rangle \in f$.
We represent with $f : A \longleftrightarrow B$ a *total bijection* from set A to set B, i.e., $f \subseteq A \times B$ such that for each $a \in A$ there exists exactly one $b \in B$ such that $\langle a, b\rangle \in f$ and, conversely, for each $b \in B$ there exists exactly one $a \in A$ such that $\langle a, b\rangle \in f$.
If $f : A \leftharpoondown\!\!\rightharpoonup B$ and $\langle a, b\rangle \in f$ then we write $f(a)$ for b; we represent with $f^{-1} : B \leftharpoondown\!\!\rightharpoonup A$ the *inverse function* of f, i.e., $f^{-1} = \{\langle b, a\rangle \mid \langle a, b\rangle \in f\}$; we write $\mathrm{dom}(f)$ for the *domain* of f, i.e., $\mathrm{dom}(f) = \{a \mid \langle a, b\rangle \in f \text{ for some } b \in B\}$; we write $\mathrm{cod}(f)$ for the *codomain* of f, i.e., $\mathrm{cod}(f) = \{b \mid \langle a, b\rangle \in f \text{ for some } a \in A\}$.
If $f : A \leftharpoondown\!\!\rightharpoonup B$ and $g : B \leftharpoondown\!\!\rightharpoonup C$ are partial bijections, then we represent with $f; g : A \leftharpoondown\!\!\rightharpoonup C$ the bijection obtained by the *concatenation* of f and g, i.e., $\langle a, c\rangle \in f; g$ if $\langle a, b\rangle \in f$ and $\langle b, c\rangle \in g$ for some $b \in B$.

Definition 4.1 (named set) A *named set* E is a set denoted by E, and a family of name sets indexed by E, namely $\{\mathsf{E}[e] \in \mathit{Set}\}_{e\in E}$ (i.e., $\mathsf{E}[\_]$ is a map form E to Set).
Given two named sets E and E', a *named function* $\mathsf{m} : \mathsf{E} \to \mathsf{E}'$ is a function on the sets $m : E \to E'$ and a family of name embeddings indexed by m, namely $\{\mathsf{m}[e, e'] : \mathsf{E}'[e'] \hookrightarrow \mathsf{E}[e]\}_{\langle e,e'\rangle\in m}$:

$$\begin{array}{ccccc} \mathsf{E} & \ni & e & \quad & \mathsf{E}[e] \\ \downarrow{\scriptstyle \mathsf{m}} & & \downarrow{\scriptstyle m} & & \uparrow{\scriptstyle \mathsf{m}[e,e']} \\ \mathsf{E}' & \ni & e' & & \mathsf{E}'[e'] \end{array}$$

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

Definition 4.2 (history-dependent automaton) A *history-dependent automaton* (or *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 in the names of the transition);
- a named function $\mathsf{l} : \mathsf{T} \to \mathsf{L}$, which associates a label to each transition (and embeds the names of the label in the names of the transition);
- an *initial state* $q_0 \in Q$ and an *initial embedding* $\sigma_0 : \mathsf{Q}[q_0] \hookrightarrow \mathfrak{N}$ of the local names of q_0 into the infinite, denumerable set $\mathfrak{N}$ of *global names*.

We require that $\mathsf{T}[t] = \mathrm{cod}(\mathsf{s}[t, s(t)]) \cup \mathrm{cod}(\mathsf{l}[t, l(t)])$ (*observation condition*).
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 $l(t) = \lambda$ (in brief $t : q \xrightarrow{\lambda} q'$); one of such transition is represented in Figure 1. Then $\mathsf{s}[t, q]$ embeds the names of q into the names of t, whereas $\mathsf{d}[t, q']$ 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; 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 (i.e., names a and c in figure) are discarded, or forgotten, during the transition, whereas the names that appear in the target but not in the source (i.e., name k in figure) are created during the transition. The observation condition "$\mathsf{T}[t] = \mathrm{cod}(\mathsf{s}[t, s(t)]) \cup \mathrm{cod}(\mathsf{l}[t, l(t)])$" corresponds to require that all the names that are created in the transition must appear explicitly in the label (name discarding, instead, can appear silently: this is the case of name c if figure).

4.1 Well-sorted HD-automata

The HD-automata we have defined above are satisfactory for representing the operational semantics of many history dependent formalisms, like CCS with localities (Section 6.2) and Petri nets with history-preserving bisimulation (Section 6.4). They are not adequate for the π-calculus.

In fact, let $p(a,b,c)$ and $q(a,b)$ be equivalent π-calculus agents with different sets of free names[3] and suppose the two agents perform a bound output. According to Definition 3.1, in checking bisimilarity we require that the object of the bound output is a new name for *both* agents. On the HD-automata this can be achieved by representing the bound output with a transition that introduces a new name.

If the two agents perform an input, however, all the names must be considered as possible input values. To represent the input on a HD-automaton, we have to consider a transition for each of the names which are present in the source state, and a transition corresponding to the input of a fresh name. In both the HD-automata corresponding to p and q, hence, there are transitions corresponding to the input of names a and b and to the input of a fresh name. The transition for name c appears only in the HD-automaton of p, since c is not free in q: this transition of p is matched in q by the transition for the fresh name.

This shows that the objects of bound outputs and of inputs have different meanings: in the case of bound outputs they are *new* names, whereas the objects of inputs are either already present in the source state, or they are *universal* names (i.e., they represent *all the other* names, including the names which are free only in the other agent). In the HD-automata, however, there is only one way to introduce fresh names in a transition; so we need to add a new component to the HD-automata to distinguish between bound-output-like transitions and fresh-input-like transitions. This new component, called *sorting*, allows us to distinguish the names of a label that *must* appear in the source (the `old` names), those that *cannot* appear in the source (the `new` names) and those that *may* appear in the source (the `both` names).

Definition 4.3 (well-sorted HD-automata) Let L be a named set of labels. A *sorting* Γ for L associates to each label $\lambda \in L$ a function $\Gamma_\lambda : \mathsf{L}[\lambda] \to \{\mathtt{new}, \mathtt{old}, \mathtt{both}\}$.

A HD-automaton $\mathcal{A}$ with labels L is *well sorted* for Γ if following condition holds:

> let $t \in \mathsf{T}$ be a transition of the automaton, $l(t) = \lambda$ be its label, $n \in \mathsf{L}[\lambda]$ be a name of the label and $n' = \mathsf{l}[t,\lambda](n) \in \mathsf{T}[t]$ be the corresponding name in the transition; then:
>
> - if $\Gamma_\lambda(n) = \mathtt{old}$ then $n' \in \mathrm{cod}(\mathsf{s}[t, s(t)])$, and
> - if $\Gamma_\lambda(n) = \mathtt{new}$ then $n' \notin \mathrm{cod}(\mathsf{s}[t, s(t)])$.

Notation According to the previous considerations, the names of a transition $t : q \xrightarrow{\lambda} q'$ are classified as follows:

- $\mathsf{T}[t]_{\mathtt{new}} = \{n \mid n' \in \mathsf{L}[\lambda],\ \Gamma_\lambda(n') = \mathtt{new},\ \mathsf{l}[t,\lambda](n') = n\}$ are the new names of transition t, i.e., the names which correspond to names of the label of sort `new`;
- $\mathsf{T}[t]_{\mathtt{old}} = \{n \mid n' \in \mathsf{L}[\lambda],\ \Gamma_\lambda(n') = \mathtt{old},\ \mathsf{l}[t,\lambda](n') = n\}$ are the old names of transition t, i.e., the names which correspond to names of the label of sort `old`;
- $\mathsf{T}[t]_{\mathtt{both}} = \{n \mid n' \in \mathsf{L}[\lambda],\ \Gamma_\lambda(n') = \mathtt{both},\ \mathsf{l}[t,\lambda](n') = n\}$ are the names of transition t that can be both new or old, i.e., the names which correspond to names of the label of sort `both`;
- $\mathsf{T}[t]_{\mathtt{src}} = \mathrm{cod}(\mathsf{s}[t,q])$ are the names of transition t that are already present in the source state;
- $\mathsf{T}[t]_{\mathtt{univ}} = \mathsf{T}[t]_{\mathtt{both}} \setminus \mathsf{T}[t]_{\mathtt{src}}$ are the *universal names* of transition t, i.e., the names which correspond to names of the label of sort `both` and which are not present in the source state.

Notice that $\mathsf{T}[t]_{\mathtt{src}}$, $\mathsf{T}[t]_{\mathtt{new}}$ and $\mathsf{T}[t]_{\mathtt{univ}}$ are a partition of $\mathsf{T}[t]$, i.e., they are disjoint and their union contains all the names of t.

[3]This can be easily obtained: for instance, let $p(a,b,c) = q(a,b) + (\nu x)\,\bar{x}c.\mathbf{0}$, where the component $(\nu x)\,\bar{x}c.\mathbf{0}$ is deadlocked.

4.2 Representing π-calculus agents as HD-automata

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

$$L_\pi = \{\mathtt{tau}, \mathtt{in}, \mathtt{in}_2, \mathtt{out}, \mathtt{out}_2, \mathtt{bout}\}$$

where $\mathtt{in}_2$ and $\mathtt{out}_2$ are used when subject and object names of inputs or 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 $\mathtt{in}_2$ and $\mathtt{out}_2$ and two names (n_{sub} and n_{obj}) are associated to $\mathtt{in}$, $\mathtt{out}$ and $\mathtt{bout}$.

The sorting Γ_π on L_π is defined as follows:

λ	tau	in		$\mathtt{in}_2$	out		$\mathtt{out}_2$	bout	
$n \in L(\lambda)$	–	n_{sub}	n_{obj}	n	n_{sub}	n_{obj}	n	n_{sub}	n_{obj}
$\Gamma_\lambda(n)$	–	old	both	old	old	old	old	old	new

As we discussed in the previous section, the object names are new in the case of bound output, can be either old or new in the case of the input, and are old in the case of free output; the subject names are always old names.

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 the new names (if any) appearing in the label of the transition. A label of L_π is associated to each transition in the obvious way.

This naive construction can be improved to obtain more compact HD-automata. Consider the agent $p = x(z).B(x,y,z)$; it can perform an infinite number of input transitions, depending on the different received names. In the context of HD-automata, however, due to the local nature of names, the transitions of p corresponding to the input of all the names different from x and y are indistinguishable; so it is sufficient to consider just three input transitions for p, i.e., the inputs of names x and y, and the input of one representative of the fresh names.

Similarly, it is sufficient to consider just one bound output, whose extruded name is the representative of the names not appearing in the agent; finally, all the τ and the free output transitions have to be considered.

According to the following definition we choose to use the first name which does not appear free in p — namely $\min(\mathfrak{N} \setminus \mathtt{fn}(p))$ — as representative for the input and bound output transitions of p.

Definition 4.4 (representative transitions) A π-calculus transition $t : p \xrightarrow{\alpha} q$ is a *representative transition* if $\mathtt{n}(\alpha) \subseteq \mathtt{fn}(p) \cup \{\min(\mathfrak{N} \setminus \mathtt{fn}(p))\}$.

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 $t : p \xrightarrow{\alpha} q$, with $\alpha = ax$ (resp. $\alpha = \bar{a}(x)$), be a non-representative π-calculus transition. Then there is some representative transition $t' : p \xrightarrow{\alpha'} q'$, with $\alpha' = ay$ (resp. $\alpha' = \bar{a}(y)$), such that $q' = q\{y/x\ x/y\}$.*

If only representative transitions are used when building a HD-automaton from a π-calculus agent, the obtained HD-automaton is finite-branching (i.e., with a finite set of transitions from each state of the automaton).

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 $\mathtt{norm}$ that, given an agent p, returns a pair $(q, \sigma) = \mathtt{norm}(p)$, where q is the representative of the class of agents differing from p for bijective substitutions and $\sigma : \mathtt{fn}(q) \to \mathtt{fn}(p)$ is the bijective substitution such that $p = q\sigma$.

α	τ	xy		xx	$\bar{x}y$		$\bar{x}x$	$\bar{x}(y)$	
λ	`tau`	`in`		$\texttt{in}_2$	`out`		$\texttt{out}_2$	`bout`	
$n \in \mathsf{L}[\lambda]$	$-$	n_{sub}	n_{obj}	n	n_{sub}	n_{obj}	n	n_{sub}	n_{obj}
$\kappa(n)$	$-$	x	y	x	x	y	x	x	y

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

Definition 4.6 (from π-calculus agents to HD-automata) The HD-automaton $\mathcal{A}_p$ corresponding to a π-calculus agent p is defined as follows:

- if $\texttt{norm}(p) = (p', \sigma')$ then:
  - $p' \in Q$ is the initial state and $\mathsf{Q}[p'] = \texttt{fn}(p')$;
  - σ' is the initial embedding;
- if $q \in Q$, $t : q \xrightarrow{\alpha} q'$ is a representative transition and $\texttt{norm}(q') = (q'', \sigma)$, then:
  - $q'' \in Q$ and $\mathsf{Q}[q''] = \texttt{fn}(q'')$;
  - $t \in T$ and $\mathsf{T}[t] = \texttt{fn}(q) \cup \texttt{bn}(\alpha)$;
  - $s(t) = q$, $d(t) = q''$, $\mathsf{s}[t, q] = \mathrm{id}_{\texttt{fn}(q)}$ and $\mathsf{d}[t, q''] = \sigma$;
  - $l(t) = \lambda$ and $\mathsf{l}[t, \lambda] = \kappa$ are defined as in Table 3.

Lemma 4.7 *For every π-calculus agent p, the HD-automaton $\mathcal{A}_p$ is well-sorted.*

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

Definition 4.8 (finitary agents) The *degree of parallelism* $\texttt{deg}(p)$ of a π-calculus agent p is defined by the clauses of Definition 2.3 plus the following clause for matching:

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

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

Theorem 4.9 *Let p be a finitary π-calculus agent. Then the HD-automaton $\mathcal{A}_p$ is finite.*

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

$$q \equiv (\nu x_1)\,(\nu x_2)\,\cdots(\nu x_m)\,(q_1|q_2|\cdots|q_n)$$

where q_i are sequential processes, $n \leq n_0$, $x_i \neq x_j$ if $i \neq j$, and $x_i \in \texttt{fn}(q_1|\cdots|q_n)$.
First of all, we notice that — due to the operational semantics of the π-calculus — each component q_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
- $q_i = p_i\sigma_i$.

Since *Var* is finite, there is just a finite number of different possible candidates for p_i; so also the candidates for q_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 q_i which is finite up to bijective substitutions).
Since n is bounded by n_0, also the possible candidates for $q_1|q_2|\cdots|q_n$ 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 $x_i \in \mathtt{fn}(q_1|\cdots|q_n)$.
Therefore, the number of agents that can be reached in the construction of the HD-automaton $\mathcal{A}_p$ 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 appear 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. □

An important class of finitary agents which can be characterized syntactically is the class of the agents with *finite control*, i.e., the agents without parallel composition in the body of recursive definitions. In this case, after an initialization phase during which a finite set of processes acting in parallel is created, no new processes can be generated.

5 Bisimulation on HD-automata

In this section we introduce a notion of bisimulation on HD-automata and give some basic properties of this bisimulation. We also show that the definition of bisimulation on 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 (for instance, equivalent π-calculus agents can have different sets of free names).

Suppose that we want to check if states q_1 and q_2 are (strongly) bisimilar via the partial bijection $\delta : \mathsf{Q}[q_1] \hookrightarrow \mathsf{Q}[q_2]$ and suppose that q_1 can perform a transition $t_1 : q_1 \xrightarrow{\lambda} q_1'$. We assume for the moment that all the names of the label λ are of sorts `old` or `new`. Then we have to find a transition $t_2 : q_2 \xrightarrow{\lambda} q_2'$ 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:

a) if a name n of the label has sort `old`, then the corresponding names in the source states q_1 and q_2 must be in correspondence by δ (such names surely exist in q_1 and q_2, if the HD-automata are well-sorted);

b) if a name n of the label has sort `new`, then the corresponding names in the transitions t_1 and t_2 are put in correspondence (if the HD-automata are well-sorted, no names corresponding to n appear in the source states).

This behavior is obtained by requiring that a partial bijection $\zeta : \mathsf{T}[t_1] \hookrightarrow \mathsf{T}[t_2]$ exists such that: *i*) ζ coincides with δ if restricted to the names of the source states (obviously, via the embeddings $\mathsf{s}[t_1, q_1]$ and $\mathsf{s}[t_2, q_2]$); *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 situation is more complex if a name n of the label λ has sort `both`. We can distinguish three more cases:

c) the name n_1 in t_1 corresponding to n is already present in q_1 *and* is associated via δ' to a name of q_2; in this case a matching transition t_2 from q_2 must use for n this associated name;

d) the name n_1 in t_1 corresponding to n is already present in q_1 *and* it is *not* associated via δ to a name of q_2; in this case t_1 must be matched by a transition t_2 from q_2 which uses an universal name for n; the meaning of this is that name n_1 is handled as a special case in state q_1, but is handled by the default transition in q_2;

e) the name n_1 in t_1 corresponding to n is *not* already present in q_1 (i.e., n_1 is an universal name); in this case we require that t_1 is matched:

*e*1) by a transition from q_2 which uses an universal name for n (the two universal names are put in correspondence), and

*e*2) for each name n_2 of q_2 not appearing (via δ) in q_1, by a transition from q_2 that uses n_2 for n (in this case, a new correspondence is set for n_1 and n_2); the meaning of this is that the default transition in q_1 must match also the special cases of q_2 which are not contemplated by q_1.

This more complex behavior is obtained by requiring that, for each transition $t_1 : q_1 \xrightarrow{\lambda} q_1'$ and for each possible partial bijection ξ between the universal names of t_2 and the names of q_2 which do not already correspond to names of q_1, there is some transition $t_2 : q_2 \xrightarrow{\lambda} q_2'$ and some partial bijection $\zeta : \mathsf{T}[t_1] \longrightarrow \mathsf{T}[t_2]$ extending $\xi; \mathsf{s}[t_2, q_2]$ such that *i*) ζ satisfies the rules *a*-*e*) above[4]; *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. Notice that to a name of t_1 can correspond no name of t_2 via ζ, if no name is associated to it via δ and the name does not appear in the label.

Definition 5.1 (HD-bisimulation) Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two HD-automata on the same sorting Γ. A *HD-simulation* for $\mathcal{A}_1$ and $\mathcal{A}_2$ is a set of triples $\mathcal{R} \subseteq \{\langle q_1, \delta, q_2\rangle \mid q_1 \in Q_1, q_2 \in Q_2, \delta : \mathsf{Q}_1[q_1] \longrightarrow \mathsf{Q}_2[q_2]\}$ such that, whenever $\langle q_1, \delta, q_2\rangle \in \mathcal{R}$ then:

for each $t_1 : q_1 \xrightarrow{\lambda} q_1'$ in $\mathcal{A}_1$ and for each $\xi : \mathsf{T}_1[t_1]_{\mathtt{univ}} \longrightarrow \mathsf{Q}_2[q_2]$ such that $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$, there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ in $\mathcal{A}_2$ and some $\zeta : \mathsf{T}_1[t_1] \longrightarrow \mathsf{T}_2[t_2]$ such that:

- $\delta = \mathsf{s}_1[t_1, q_1]; \zeta; \mathsf{s}_2[t_2, q_2]^{-1}$;
- $\xi = \zeta|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}}; \mathsf{s}_2[t_2, q_2]^{-1}$;
- $\mathsf{l}_1[t_1, \lambda]; \zeta = \mathsf{l}_2[t_2, \lambda]$;
- $\langle q_1', \delta', q_2'\rangle \in \mathcal{R}$ where $\delta' \subseteq \mathsf{d}_1[t_1, q_1']; \zeta; \mathsf{d}_2[t_2, q_2']^{-1}$.

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 *(strongly) 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 $\delta \subseteq \sigma_{01}; \sigma_{02}^{-1}$.

5.1 Some basic properties of HD-bisimulation

In this section we present some basic properties of HD-bisimulations.

Proposition 5.2 *If $\mathcal{R}_i$ with $i \in I$ are 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 5.3 We denote with $\mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2}$ the greatest HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, i.e.:

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

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

[4]For rule *e*): let n_1 be an universal name of t_1; if $\xi(n_1) = n_2 \in \mathsf{Q}[q_2]$ then n_1 must be matched by n_2 (case *e*2); if $\xi(n_1)$ is undefined, an universal name of t_2 must match n_1 (case *e*1).

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

Proposition 5.4 *If $\mathcal{R}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ and $\mathcal{S}$ is a HD-bisimulations for $\mathcal{A}_2$ and $\mathcal{A}_3$ then $\mathcal{R} \frown \mathcal{S}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_3$, where:*

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

The proof of this proposition is developed in Appendix A.1.

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 5.5 *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 5.6 *If $\mathcal{R}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\widehat{\mathcal{R}}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, where:*

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

The proof can be found in Appendix A.1.

Corollary 5.7 *$\mathcal{R}_{\mathcal{A}}$ is closed for $\widehat{-}$ and $- \frown -$.*

Proposition 5.6 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 could show that an alternative definition of bisimulation with a stronger condition on the correspondence δ' for the destination states: in fact, we could require $\delta' = \mathsf{d}_1[t_1, q_1']; \zeta; \mathsf{d}_2[t_2, q_2']^{-1}$. Also with this alternative definition we would obtain the same equivalence on HD-automata and also the maximal bisimulation $\mathcal{R}_{\mathcal{A}_1;\mathcal{A}_2}$ would not change. However, some interesting properties hold only in the weaker context; this is the case for the concatenation property of Proposition 5.4.

5.2 Global states and global bisimulation

In Section 5.3 we show that the definition of HD-bisimulation applied to HD-automata obtained from π-calculus agents induces on the agents an equivalence which coincides with the strong, early bisimilarity relation $\sim_\pi$.

To obtain this it is very useful to give an alternative characterization of HD-bisimulation which is based on states with global (rather than local) names: in fact, names in π-calculus are global. Therefore, we assume to have an infinite denumerable set $\mathfrak{N}$ of names (for simplicity we use the same set which we have used to define π-calculus agents) and begin by defining global states and global transitions for a HD-automaton.

Definition 5.8 (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 $\sigma : \mathsf{Q}[q] \hookrightarrow \mathfrak{N}$. The set of all the global states of $\mathcal{A}$ is denoted by G. It can be extended to a named set G by defining $\mathsf{G}[(q, \sigma)] = \sigma(\mathsf{Q}[q])$.

A *global transition* is a pair $u = (t, \rho)$, where $t \in T$ and $\rho : \mathsf{T}[t] \hookrightarrow \mathfrak{N}$. The set of all the global transitions of $\mathcal{A}$ is denoted by U. It can be extended to a named set U by defining $\mathsf{U}[(t, \rho)] = \rho(\mathsf{T}[t])$ (and similarly for $\mathsf{U}[(t,\rho)]_{\mathtt{src}}$, $\mathsf{U}[(t,\rho)]_{\mathtt{univ}}$, $\mathsf{U}[(t,\rho)]_{\mathtt{new}}$, $\mathsf{U}[(t,\rho)]_{\mathtt{old}}$ and $\mathsf{U}[(t,\rho)]_{\mathtt{both}}$).

If $t : q \xrightarrow{\lambda} q'$ then we write $(t, \rho) : (q, \sigma) \xrightarrow{(\lambda, \kappa)} (q', \sigma')$, where $\sigma = \mathsf{s}[t, q]; \rho$, $\kappa = \mathsf{l}[t, \lambda]; \rho$ and $\sigma' = \mathsf{d}[t, q']; \rho$.

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

Definition 5.9 (global bisimulation) Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two HD-automata on the same sorting Γ. 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 $g_1 \mathrel{\mathcal{R}} g_2$ then:

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

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 "$\mathsf{U}_1[u_1]_{\mathtt{new}} \cap \mathsf{G}_2[g_2] = \emptyset$" 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 very similar to the clause "$\mathtt{bn}(\alpha) \cap \mathtt{fn}(p, q) = \emptyset$" in Definition 3.1.

Global bisimilarity coincides with HD-bisimilarity:

Proposition 5.10 *Two HD-automata are HD-bisimilar if and only if they are global bisimilar.*

Proof (sketch) To prove that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are HD-bisimilar then they are also global bisimilar, we show that if $\mathcal{R}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\mathcal{R}'$ is a global-bisimulation for them, where

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

Similarly, to prove the converse we show that if $\mathcal{R}$ is a global bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ then $\mathcal{R}'$ is a HD-bisimulation for them, where

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

The details of the proof are developed in Appendix A.2. □

5.3 Relating π-calculus bisimulation and HD-bisimulation

In this section 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 5.11 *Let (q, σ) be a global state of the HD-automaton $\mathcal{A}_p$ corresponding to a π-calculus agent p. Then:*

- *if $q\sigma \xrightarrow{\alpha} q''$ is a π-calculus transition and $(q', \sigma') = \mathtt{norm}(q'')$, then there is some global transition $(t, \rho) : (q, \sigma) \xrightarrow{(\lambda, \kappa)} (q', \sigma')$ of $\mathcal{A}_p$; and*
- *if $(t, \rho) : (q, \sigma) \xrightarrow{(\lambda, \kappa)} (q', \sigma')$ is a global transition of $\mathcal{A}_p$, then there is some π-calculus transition $q\sigma \xrightarrow{\alpha} q'\sigma'$*

where in both cases λ and κ are related to α as in Table 3.

The proof of this lemma and of the following theorem are reported Appendix A.3.

Theorem 5.12 *Let p_1 and p_2 be π-calculus agents. Then $p_1 \sim_\pi p_2$ iff $\mathcal{A}_{p_1} \sim \mathcal{A}_{p_2}$.*

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

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

By exploiting Lemma 5.11 we show that $\mathcal{R}$ is a global bisimulation for $\mathcal{A}_{p_1}$ and $\mathcal{A}_{p_2}$. Moreover $(q_{01}, \sigma_{01})\ \mathcal{R}\ (q_{02}, \sigma_{02})$, since $q_{01}\sigma_{01} = p_1 \sim_\pi p_2 = q_{02}\sigma_{02}$, so the two HD-automata are HD-bisimilar.
Conversely, suppose $\mathcal{A}_{p_1} \sim \mathcal{A}_{p_2}$ and let $\mathcal{S}$ be the largest global bisimulation for the two automata (i.e., the one corresponding to HD-bisimulation $\mathcal{R}_{\mathcal{A}_{p_1};\mathcal{A}_{p_2}}$). By exploiting Lemma 5.11 we show that

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

is a π-calculus bisimulation. In particular, since $(q_{01}, \sigma_{01})\ \mathcal{S}\ (q_{02}, \sigma_2)$ and $p_1 = q_{01}\sigma_{01}\ p_2 = q_{02}\sigma_{02}$, we obtain $p_1 \sim_\pi p_2$. □

5.4 Active names and irredundant automata

As explained in Section 5, there are equivalent states of a HD-automaton with different sets of names: this obliges to use partial correspondences of names in the bisimulations and function ξ to correctly handle universal names. Now we show how it is possible to discard unused names to guarantee that equivalent states have "exactly the same names", allowing so for the application of a simpler definition of bisimulation.

Definition 5.13 (active names) Let $\mathcal{A}$ be a HD-automaton and q a state of $\mathcal{A}$. Name $n \in \mathsf{Q}[q]$ is *inactive* for q if $\langle q, \mathrm{id}_{\mathsf{Q}[q] \setminus \{n\}}, q\rangle \in \mathcal{R}_{\mathcal{A}}$.
A name is *active* if it is not inactive. The active names of a state q of A are denoted by $\mathsf{Q}[q]_{\text{active}}$.

Lemma 5.14 *Given a finitely named HD-automaton $\mathcal{A}$ and a state q, we have $\langle q, \mathrm{id}_{\mathsf{Q}[q]_{\text{active}}}, q\rangle \in \mathcal{R}_{\mathcal{A}}$.*

Proposition 5.15 *Let $\mathcal{A}$ be a finitely named HD-automaton and let*

$$\mathcal{B}_{\mathcal{A}} \stackrel{\text{def}}{=} \{\langle p, \delta, q\rangle \in \mathcal{R}_{\mathcal{A}} \mid \delta \subseteq \mathsf{Q}[p]_{\text{active}} \times \mathsf{Q}[q]_{\text{active}}\}.$$

Then $\mathcal{B}_{\mathcal{A}}$ is a HD-bisimulation.

The proof can be found in Appendix A.4.

Notice that, by Lemma 5.14, $\widehat{\mathcal{B}_{\mathcal{A}}} = \mathcal{R}_{\mathcal{A}}$. In this sense, relation $\mathcal{B}_{\mathcal{A}}$ can be considered the "best" bisimulation for the HD-automaton $\mathcal{A}$: in fact, all the triples which appear in $\mathcal{R}_{\mathcal{A}}$ but not in $\mathcal{B}_{\mathcal{A}}$ are "redundant".

Notice also that the condition for $\mathcal{A}$ to be finitely named is necessary in Lemma 5.14 and in Proposition 5.15; in fact, it is possible to build HD-automata with infinite set of names associated to states and infinite set of names in the label for which the results of the lemma and of the proposition above do not hold.

Proposition 5.16 *Let $\mathcal{A}$ be a finitely named HD-automaton and let $\langle p, \delta, q\rangle \in \mathcal{B}_{\mathcal{A}}$. Then $\delta : \mathsf{Q}[p]_{\text{active}} \hookrightarrow \mathsf{Q}[q]_{\text{active}}$ is a total bijection.*

Proof Suppose that $\delta(n)$ is not defined for some $n \in \mathsf{Q}[p]_{\text{active}}$. Since $\langle p, \delta, q\rangle \in \mathcal{R}_{\mathcal{A}}$ we also have $\langle q, \delta^{-1}, p\rangle \in \mathcal{R}_{\mathcal{A}}^{-1} = \mathcal{R}_{\mathcal{A}}$. Therefore $\langle p, \delta; \delta^{-1}, p\rangle \in \mathcal{R}_{\mathcal{A}} \frown \mathcal{R}_{\mathcal{A}} = \mathcal{R}_{\mathcal{A}}$.
Since $\delta(n)$ is not defined, we have $\delta; \delta^{-1} \subseteq \mathrm{id}_{\mathsf{Q}[p] \setminus \{n\}}$. Hence $\langle p, \mathrm{id}_{\mathsf{Q}[p] \setminus \{n\}}, p\rangle \in \widehat{\mathcal{R}}_{\mathcal{A}} = \mathcal{R}_{\mathcal{A}}$.
By definition of active names, this implies $n \notin \mathsf{Q}[p]_{\text{active}}$, which is absurd.
Therefore δ is necessarily total on $\mathsf{Q}[p]_{\text{active}}$; analogously, δ^{-1} is total on $\mathsf{Q}[q]_{\text{active}}$. If $\langle p, \delta, q\rangle \in \mathcal{B}_{\mathcal{A}}$ then $\delta \subseteq \mathsf{Q}[p]_{\text{active}} \times \mathsf{Q}[q]_{\text{active}}$ and hence δ is a total and bijective function from $\mathsf{Q}[p]_{\text{active}}$ onto $\mathsf{Q}[q]_{\text{active}}$. □

Now we are ready to describe the fine structure of $\mathcal{R}_{\mathcal{A}}$, with $\mathcal{A}$ a finitely named HD-automaton. First of all, by Corollary 5.7, $\mathcal{R}_{\mathcal{A}}$ partitions the states Q of $\mathcal{A}$ in blocks of mutually equivalent states, i.e., relation

$$\{(p, q) \mid \langle p, \delta, q\rangle \in \mathcal{R}_{\mathcal{A}} \text{ for some } \delta\}$$

is an equivalence.

Consider a single state $q \in Q$, and define $\Delta_q = \{\delta \mid \langle q, \delta, q\rangle \in \mathcal{B}_{\mathcal{A}}\}$. By Corollary 5.7 and Proposition 5.16, Δ_q is a group of permutations of $\mathsf{Q}[q]_{\text{active}}$. Moreover, $\langle q, \delta, q\rangle \in \mathcal{R}_{\mathcal{A}}$ if and only if δ can be obtained by one (necessarily unique) permutation of Δ_q by adding some correspondences between the names of q which are inactive.

The following proposition strongly relates the active names of two equivalent states and hence formalized the assertion that "equivalent states have the same active names".

Proposition 5.17 *Let $\mathcal{A}$ be a finitely named HD-automaton. If $\langle p, \delta, q\rangle \in \mathcal{B}_{\mathcal{A}}$ for some δ, then $\mathsf{Q}[q]_{\text{active}} = \delta(\mathsf{Q}[p]_{\text{active}})$ and $\Delta_q = \{\delta^{-1}; \rho; \delta \mid \rho \in \Delta_p\}$.*

Proof The fact that $\mathsf{Q}[q]_{\text{active}} = \delta(\mathsf{Q}[p]_{\text{active}})$ is a direct consequence of Proposition 5.16.
Let $\rho \in \Delta_p$; then, by definition of Δ_p,

$$\langle p, \rho, p\rangle \in \mathcal{B}_{\mathcal{A}}.$$

Hence and from $\langle p, \delta, q\rangle \in \mathcal{B}_{\mathcal{A}}$ we deduce

$$\langle q, \delta^{-1}; \rho; \delta, q\rangle \in \mathcal{B}_{\mathcal{A}}^{-1} \frown \mathcal{B}_{\mathcal{A}} \frown \mathcal{B}_{\mathcal{A}}.$$

Since $\mathcal{B}_{\mathcal{A}}$ is closed for concatenation and symmetry, this implies

$$\langle q, \delta^{-1}; \rho; \delta, q\rangle \in \mathcal{B}_{\mathcal{A}}$$

and hence $\delta^{-1}; \rho; \delta \in \Delta_q$.
This proves that $\{\delta^{-1}; \rho; \delta \mid \rho \in \Delta_p\} \subseteq \Delta_q$; the converse inclusion is similar. □

Now we consider the possibility to discard the names which are not active in the states of a HD-automaton. In this manner we obtain a reduced HD-automaton whose equivalent states have "the same active names". When the inactive names are discarded from a state, also those transitions leaving the state have to be removed, which use in their labels names of the state which are inactive.

Definition 5.18 (active transitions) Let $t : q \xrightarrow{\lambda} q'$ be a transition of a HD-automaton $\mathcal{A}$. The transition is *active* if all the names of the source state which are used in the label are active, i.e., if

$$\mathrm{cod}(\mathsf{l}[t, \lambda]; \mathsf{s}[q, t]^{-1}) \subseteq \mathsf{Q}[q]_{\mathrm{active}}.$$

In this case, the *active names* of transition t are defined as follows:

$$\mathsf{T}[t]_{\mathrm{active}} = \mathsf{s}[t, q](\mathsf{Q}[q]_{\mathrm{active}}) \cup \mathrm{cod}(\mathsf{l}[t, \lambda]).$$

Definition 5.19 Let $\mathcal{A}$ be a HD-automaton. The *irredundant version* of $\mathcal{A}$ is the HD-automaton $\Downarrow\!\mathcal{A}$ such that:

- $Q_{\Downarrow} = Q$ and, for each $q \in Q$, $\mathsf{Q}_{\Downarrow}[q] = \mathsf{Q}[q]_{\mathrm{active}}$; in particular, $q_{0\Downarrow} = q_0$;
- $T_{\Downarrow}$ are the active transitions of $\mathcal{A}$ and, for each $t \in T_{\Downarrow}$, $\mathsf{T}_{\Downarrow}[t] = \mathsf{T}[t]_{\mathrm{active}}$;
- $\mathsf{L}_{\Downarrow} = \mathsf{L}$;
- $\mathsf{s}_{\Downarrow}$, $\mathsf{d}_{\Downarrow}$ and $\mathsf{l}_{\Downarrow}$ are the opportune restrictions of the corresponding named functions in $\mathcal{A}$.

An HD-automaton $\mathcal{A}$ is *irredundant* if $\mathcal{A} = \Downarrow\!\mathcal{A}$.

Proposition 5.20 *Let $\mathcal{A}$ be a HD-automaton. Then $\mathcal{A} \sim \Downarrow\!\mathcal{A}$.*

The proof can be found in Appendix A.4.

Irredundant automata are very convenient. In the next proposition, for instance, we show that the definition of HD-bisimulation is quite simpler on irredundant HD-automata.

Proposition 5.21 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be irredundant HD-automata. Then $\mathcal{R} \subseteq \{\langle q_1, \delta, q_2\rangle \mid q_1 \in Q_1, q_1 \in Q_2, \delta : \mathsf{Q}_1[q_1] \hookrightarrow \mathsf{Q}_2[q_2]\}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ if and only if whenever $\langle q_1, \delta, q_2\rangle \in \mathcal{R}$ then:*

- $\delta : \mathsf{Q}_1[q_1] \longleftrightarrow \mathsf{Q}_2[q_2]$ *(i.e., δ is a total bijection between $\mathsf{Q}_1[q_1]$ and $\mathsf{Q}_2[q_2]$);*
- *for each $t_1 : q_1 \xrightarrow{\lambda} q_1'$ in $\mathcal{A}_1$ and for each there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ in $\mathcal{A}_2$ and some $\zeta : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_2[t_2]$ such that:*
  - $\delta = \mathsf{s}_1[t_1, q_1]; \zeta; \mathsf{s}_2[t_2, q_2]^{-1}$;
  - $\mathsf{l}_1[t_1, \lambda]; \zeta = \mathsf{l}_2[t_2, \lambda]$;
  - $\langle q_1', \delta', q_2'\rangle \in \mathcal{R}$ *where* $\delta' = \mathsf{d}_1[t_1, q_1']; \zeta; \mathsf{d}_2[t_2, q_2']^{-1}$;
- *the converse, starting from the transitions of q_2.*

The proof of this proposition is developed in Appendix A.4.

Since the maps between names are total in the case of irredundant HD-automata, it is possible to map them on ordinary automata.

Definition 5.22 The *unfolding* corresponding to the HD-automaton $\mathcal{A}$ is the ordinary automaton $U = \mathtt{unf}(\mathcal{A})$ defined as follows:

- states of U are global states of $\mathcal{A}$ and transitions of U are global transitions of $\mathcal{A}$;
- labels of U have the form $(\lambda, \kappa, \gamma)$, where λ is a label of $\mathcal{A}$, $\kappa : \mathsf{L}[\lambda] \hookrightarrow \mathfrak{N}$ gives global meaning to the names of the label, and $\gamma \subseteq \mathsf{L}[\lambda]$ represents the fresh names introduced by the transition;
- the initial state of U is $q_{0U} = \langle q_{0\mathcal{A}}, \sigma_{0\mathcal{A}} \rangle \in Q_U$;
- whenever $(q, \sigma) \in Q_U$ and $(t, \rho) : (q, \sigma) \xrightarrow{(\lambda,\kappa)} (q', \sigma')$ is a global transition of $\mathcal{A}$ with

$$\max\{n \in \mathsf{U}[(t,\rho)]_{\mathtt{univ}} \cup \mathsf{U}[(t,\rho)]_{\mathtt{new}}\} < \min\{n \notin \mathsf{U}[(t,\rho)]\}$$

  then $(q', \sigma') \in Q_U$ and $(t, \rho) \in T_U$, with

  - $s_U(t, \rho) = (q, \sigma)$ and $d_U(t, \rho) = (q', \rho)$;
  - $l_U(t, \rho) = (\lambda, \kappa, \gamma)$, where $\gamma \stackrel{\text{def}}{=} \mathsf{U}[(t,\rho)]_{\mathtt{univ}} \cup \mathsf{U}[(t,\rho)]_{\mathtt{new}}$.

Now we show that two HD-automata are HD-bisimilar if and only if their unfoldings are bisimilar according to the ordinary notion of bisimulation on automata. This proof exploits the characterization of HD-bisimulation for irredundant HD-automata given in Proposition 5.21.

Theorem 5.23 *Let $\mathcal{A}$ and $\mathcal{B}$ be two irredundant HD-automata. Then $\mathcal{A} \sim \mathcal{B}$ if and only if* $\mathtt{unf}(\mathcal{A}) \sim \mathtt{unf}(\mathcal{B})$.

Proof (sketch) To prove that bisimilarity on the irredundant HD-automata implies bisimilarity on the corresponding unfoldings we show that, if $\mathcal{A}_1$ and $\mathcal{A}_2$ are HD-bisimilar via the HD-bisimulation $\mathcal{R}$, then $\mathtt{unf}(\mathcal{A}_1)$ and $\mathtt{unf}(\mathcal{A}_2)$ are bisimilar via

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

Similarly, to prove that bisimilarity on the unfoldings implies bisimilarity on the HD-automata we show that, if $\mathtt{unf}(\mathcal{A}_1)$ and $\mathtt{unf}(\mathcal{A}_2)$ are bisimilar via the bisimulation $\mathcal{R}$, then $\mathcal{A}_1$ and $\mathcal{A}_2$ are HD-bisimilar via the HD-bisimulation

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

The details of the proof are reported in Appendix A.5. □

In Section 8 we will exploit this theorem to describe how classical algorithms for bisimulation checking on ordinary automata can be applied for checking HD-bisimulation.

6 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 HD-automata.

6.1 CCS with localities

Here we briefly recall the approach to locality semantics for CCS that has been introduced in [7, 26]. In Section 6.2 we show how to map this semantics for 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, the agents $\alpha.\mathbf{0}|\beta.\mathbf{0}$ and $\alpha.\beta.\mathbf{0}+\beta.\alpha.\mathbf{0}$, which 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 [1] — where the distributed nature of agents is made explicit by assigning different locations to their parallel components, as in $a.(l :: p|m :: q)$ — in [7, 26] a more observational point of view is preferred. Location names are assigned *dynamically*, during the process of observation: the meaning of transition $l :: a.p \xrightarrow[lm]{a} l :: m :: p$ is that the observer sees an action a emanating from a particular sublocation of l and associates name m to this sublocation[5]. Here we follow the 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,

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

whereas

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

and m is a sublocation 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 $\mathtt{loc}(p)$ and an agent p is *pure* if $\mathtt{loc}(p) = \emptyset$.

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

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

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 $p \xrightarrow[u]{a} p'$, which are generated by the axiom schemata and by the inference rules of Table 4. 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.

Definition 6.1 (location equivalence) A relation $\mathcal{R}$ over location agents is a *location simulation* if whenever $p \mathrel{\mathcal{R}} q$ then:

[5] We refer to [10] for further comparisons of static and dynamic approach.

[PREF] $a.p \xrightarrow[l]{a} l :: p$	[LOC] $\dfrac{p \xrightarrow[u]{a} p'}{l :: p \xrightarrow[lu]{a} l :: p'}$
[SUM] $\dfrac{p_1 \xrightarrow[u]{a} p'}{p_1 + p_2 \xrightarrow[u]{a} p'}$	[PAR] $\dfrac{p_1 \xrightarrow[u]{a} p_1'}{p_1 \vert p_2 \xrightarrow[u]{a} p_1' \vert p_2}$
[RES] $\dfrac{p \xrightarrow[u]{a} p'}{(\nu\alpha)\, p \xrightarrow[u]{a} (\nu\alpha)\, p'}$ if $a \neq \alpha, \bar{\alpha}$	[IDE] $\dfrac{p_A \xrightarrow[u]{a} p'}{A \xrightarrow[u]{a} p'}$ if $A \stackrel{\text{def}}{=} p_A$

Table 4: Location transitions for CCS with localities.

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

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 equivalent*, 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 similar to the problems with π-calculus. 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 "$l \notin \mathtt{loc}(p, q)$" in Definition 6.1.

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., $\sigma : Loc \rightarrow Loc$; then we represent with $p[\sigma]$ the agent p whose localities have been updated according to σ.

Lemma 6.2 *Let p be a location agent and let $\sigma : Loc \longleftrightarrow Loc$ be a bijective 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 $v = \sigma(l_1)\sigma(l_2)\cdots\sigma(l_k)$ when $u = l_1 l_2 \cdots l_k$. 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 [7].

With a little abuse of notation, we will also write $p[\sigma]$ with $\sigma : \mathtt{loc}(p) \rightarrow Loc$, with the obvious meaning.

6.2 Representing agents with localities as HD-automata

A first 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, as in the case of the π-calculus. 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 injective renamings 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 injective renamings.

As a first step, 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

(del) $l :: m :: p \equiv m :: p$

would help. However, this axiom is not correct for the location equivalence of Definition 6.1, 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 6.3 (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 $n \notin \mathtt{loc}(p, q)$, 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 6.4 (incremental location equivalence) *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 assure 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 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 $m_i \notin \mathtt{loc}(p_{i-1}, q_{i-1})$. Hence, $m :: p_0 \sim_l m :: q_0$ and, by exploiting standard results for location equivalence, $p_0 \sim_l q_0$.

The details of the proof are reported in Appendix A.6. $\square$

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 [11] 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 6.5 *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 6.6 *If $p' \equiv_\Delta p$, $p \sim_{\Delta l} q$ and $q \equiv_\Delta q'$ then $p' \sim_{\Delta l} q'$.*

At this point we are ready to map location agents into HD-automata. This mapping is similar to the one defined in the previous section for the π-calculus. In this case, we can assume that the set *Loc* of locations is used as set $\mathfrak{N}$ of global names in the HD-automata.

We assume 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 bijective renaming of the locations. The injective renaming $\sigma : \mathtt{loc}(q) \hookrightarrow \mathtt{loc}(p)$ associates a location of p to each location of q, so that $p \equiv_\Delta q[\sigma]$. Agent q can be obtained in practice, by first transforming p in an agent p' that is normalized w.r.t. $\equiv_\Delta$; for instance, by resorting on Proposition 6.5, we could have $p' = (\nu\alpha_1)\cdots(\nu\alpha_m)\,(p_0|l_1 :: p_1|\cdots|l_n :: p_n)$. Then the locations of p' can be renamed so that l_i is the i-th location in *Loc*.

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 $m = \min\{Loc \setminus \mathtt{loc}(p)\}$.

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. The sort of name n_{loc} is `old`, whereas the sort of name n_{new} is `new`. Notice that no names of sort `both` are used in this case; hence, the sorting is not necessary to deal with localities — and, in fact, it does not appear in [35].

Definition 6.7 (from location agents to HD-automata) The HD-automaton $\mathcal{A}_p$ corresponding to the pure CCS agent p is defined as follows:

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

- $t \in T$ and $\mathsf{T}[t] = \mathtt{loc}(q)$;
- $s(t) = q$, $d(t) = q''$, $\mathsf{s}[t, q] = \mathrm{id}_{\mathtt{loc}(q)}$ and $\mathsf{d}[t, q''] = \sigma$;
- $l(t) = \tau$.

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

Theorem 6.8 *Given a pure location agent p, the HD-automaton $\mathcal{A}_p$ is finite if and only if p is finitary according to Definition 2.3.*

The proof is similar to that of Theorem 4.9.

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

Theorem 6.9 *Let p_1 and p_2 be two pure CCS agents. Then $p_1 \sim_l p_2$ iff $\mathcal{A}_{p_1} \sim \mathcal{A}_{p_2}$.*

As in the case of π-calculus agents, to prove this result we exploit the global characterization of HD-bisimulation; it is based on the following lemma.

Lemma 6.10 *Let (q, σ) be a global state of the HD-automaton $\mathcal{A}_p$ corresponding to a pure location agent p. Then:*

- *if $q[\sigma] \xrightarrow[umn]{a} q''$ is a location transition and $(q', \sigma') = \mathtt{norm}(q'')$, then there is some global transition $(t, \rho) : (q, \sigma) \xrightarrow{(a,\kappa)} (q', \sigma')$ of $\mathcal{A}_p$ and $q'' = q'[\sigma']$; and*
- *if $(t, \rho) : (q, \sigma) \xrightarrow{(a,\kappa)} (q', \sigma')$ is a global transition of $\mathcal{A}_p$, then there is some location transition $q[\sigma] \xrightarrow[umn]{a} q'[\sigma']$*

where in both cases $\kappa = \mathsf{l}[t, \lambda]; \rho$, and $\mathsf{l}[t, \lambda](n_{\mathtt{loc}}) = m$, $\mathsf{l}[t, \lambda](n_{\mathtt{new}}) = n$. A similar property holds for the τ transitions.

The proof is similar to the proof of the corresponding lemma for the π-calculus — Lemma 5.11. It relies on Lemma 6.2.

6.3 Petri nets

In this section we consider an example of history-dependent formalism outside the field of process calculi, namely Petri nets.

In the case of Petri nets with labelled transitions, an automaton is used to represent the *case graph* of a net, i.e., the graph of the reachable markings of the net; 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. This fact leads to a notion of equivalence of nets. However, for Petri nets it is preferred to apply a more refined notion of equivalence, that also considers the causal relations on the firings of the net.

Processes have been defined in [16] 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*, which takes into account the partial order behavior has been defined in [40] for event structures. The same notion has been introduced before in [15] for process calculi, using mixed ordering observations. History-preserving bisimulation has been applied to Petri nets in [8]: for two nets to simulate, it is required not only that they can 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 has problems similar to the ones of π-calculus and CCS with localities.

Now we present the basic definitions on Petri nets. Most of the definitions and of the notations are from [16].

Definition 6.11 (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 6.12 (place/transition net) A *(labelled, marked) place/transition net* (or simply *P/T net*) N is a tuple (S, T, F, W, l, m_0), where (S, T, F) is a net and:

- $W : F \to \mathbb{N}^+$ 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;
- $m_0 : S \to \mathbb{N}$ is the *initial marking*.

A *marking* is a mapping $m : S \to \mathbb{N}$. 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 6.13 (n-safe nets) A P/T net N is n-safe if for each reachable marking m (i.e., for each m such that $m_{0N} \to \cdots \to m$) we have:

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

Definition 6.14 (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 6.15 (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[6] 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 [8] under the name of *fully concurrent bisimulation*.

[6] Notice that the initial process of a net is unique only up to isomorphism of the set of initial conditions.

Definition 6.16 (event structure) The *(deterministic) event structure* for process $\pi = (C, E, G, p)$ of net N is the tuple $\mathtt{ev}(\pi) = (E, F^+\big|_E, l_N \circ p\big|_E)$. An *isomorphism* between two event structures is a bijective function between their events which respects ordering and labels.

Definition 6.17 (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 $\mathtt{ev}(\pi_1)$ and $\mathtt{ev}(\pi_2)$;
- $(\pi_0(N_1), \emptyset, \pi_0(N_2)) \in \mathcal{R}$;
- 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 $f'\big|_{\mathtt{ev}(\pi_1)} = f$;
- 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 $f'\big|_{\mathtt{ev}(\pi_1)} = f$.

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.

6.4 Representing Petri nets as HD-automata

Now we show that also the history-preserving semantics for Petri nets can be modeled by HD-automata. The classical definition of history-preserving bisimulation is based on processes. Since processes continue to grow during a computation, 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, as we have done in the previous section for CCS with localities.

The first step consists in 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 6.18 (configuration) Let N be a P/T net. A *configuration* for N is a tuple $c = (E, \leq, \rho)$, where:

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

We require that $\sum_{s \in S_N} \rho(s, e) > 0$ for each $e \in E$.
The *initial configuration* for N is the configuration $c_0(N) = (\emptyset, \rho_0, \emptyset)$, where $\rho_0(s, \mathtt{init}) = m_0(s)$ 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[7]:

- $E' \subseteq E \cup \{\bar{e}\}$ for some $\bar{e} \notin E$;
- $\leq' = \big(\leq \cup\, (\mathcal{IC}(c \xrightarrow{\bar{t}}_{\bar{e}} c') \times \{\bar{e}\})\big)^* \cap (E' \times E')$;
- $\rho(s, e) > \rho'(s, e)$ for all $s \in S_N$ and $e \in (E \cup \{\mathtt{init}\})$, and $\sum_{e \in (E \cup \{\mathtt{init}\})} (\rho(s, e) - \rho'(s, e)) = W_N(s, \bar{t})$ for all $s \in S$;
- $\rho'(s, \bar{e}) = W_N(\bar{t}, s)$ for all $s \in S_N$;

[7] For simplicity, in this definition we suppose that $\rho(s, e) = 0$ if $e \notin E$ and, similarly, $\rho'(s, e) = 0$ if $e \notin E'$.

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 $c \xrightarrow{t}_{\bar{e}} c'$ is *representative* if $\bar{e} = \min\{\mathfrak{N} \setminus E\}$.

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; instead of 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. It is possible to show, instead, that two matching events must have the same sets of maximal causes.

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

Definition 6.19 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 6.20 *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 6.21 (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 $\leq = G^*|_{E'}$;
- $\rho(s,e) = |p^{-1}(s) \cap \pi^\circ \cap e^\bullet|$ for all $s \in S_N$ and $e \in E$; moreover $\rho(s, \texttt{init}) = |p^{-1}(s) \cap \pi^\circ \cap {}^\circ\pi|$ 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 6.22 *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 6.23 (incremental history-preserving bisimulation) A set $\mathcal{R}$ of triples is a *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 process 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 a history-preserving bisimulation for them.

Proposition 6.24 *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; we show that

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

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

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

is a history-preserving bisimulation. This is sufficient to prove that $N_1 \sim_{\Delta hp} N_2$ implies $N_1 \sim_{hp} N_2$.
The details of the proof are in Appendix A.7. □

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 6.25 (isomorphic configurations) Let N be a P/T net. Two configurations $c = (E, \leq, \rho)$ and $c' = (E', \leq', \rho')$ for 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 $\mathtt{norm}(c) = (c', \sigma)$ 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 $\sigma(c) \stackrel{\text{def}}{=} c'$.

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_{hp} of labels is defined as follows: $L_{hp} = \{(\alpha, k) \mid \alpha \in Act \wedge k \in \mathbb{N}\}$ and the $n+1$ names $n_1, \ldots, n_k, n_{\text{new}}$ correspond to (α, k) for any $\alpha \in Act$. The names n_i for $i = 1 \cdots k$ have sort `old`, whereas name n_{new} has sort `new`.

Definition 6.26 (from Petri nets to HD-automata) The HD-automaton $\mathcal{A}_N$ that corresponds to the P/T net N is defined as follows:

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

This construction generates finite causal automata for the finite nets which 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 [44].

Theorem 6.27 *Given a finite P/T net N, the HD-automaton $\mathcal{A}_N$ is finite if and only if N is n-safe for some n.*

Proof We show that $\mathcal{A}_N$ 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 $\mathcal{A}_N$ is a configuration $c = (E, \leq, \rho)$ of N, where:

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

By Proposition 6.20, 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 to 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 6.28 *Let N_1 and N_2 be two P/T nets. Then $N_1 \sim_{hp} N_2$ iff $\mathcal{A}_{N_1} \sim \mathcal{A}_{N_2}$.*

As in the case of π-calculus agents, 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 5.11 in the case of π-calculus.

Lemma 6.29 *Let (c, ρ) be a global state of the HD-automaton $\mathcal{A}_N$ corresponding to a P/T net N. Then:*

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

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

7 A categorical approach to HD-automata

In this section we give an alternative characterization of HD-automata and of HD-bisimulation in a categorical framework. As we will see, these definitions are clean extensions of classical categorical definitions of ordinary automata and of bisimulation for categorical models.

7.1 The category of HD-automata

In a categorical framework, an ordinary automaton can be defined as a diagram

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

in the category **Set** of sets. In this case, the initial state q_0 is designated as $i(*)$.

Given two automata A_1 and A_2 on the same set L of labels, a *morphism* $m : A_1 \to A_2$ 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

$$\begin{tikzcd} & T_1 \arrow[dl, "l_1"'] \arrow[d, "m_T"] \arrow[r, bend left, "s_1"] \arrow[r, bend right, "d_1"'] & Q_1 \arrow[d, "m_Q"] & \\ L & & & \{*\} \arrow[ul, "i_1"'] \arrow[dl, "i_2"] \\ & T_2 \arrow[ul, "l_2"] \arrow[r, bend left, "s_2"] \arrow[r, bend right, "d_2"'] & Q_2 & \end{tikzcd}$$

commute in the obvious way.

The category $\mathbf{Aut}_L$ of the automata on labels L is defined by using automata with labels L as objects and morphisms between such automata as arrows; identity arrow and composition between arrows are defined in the obvious way.

HD-automata can be defined in a similar way; we have just to replace the category **Set** with a category of named sets.

Definition 7.1 (category of named sets) The *category* **NSet** *of named sets* has named sets as objects and named functions as arrows; in particular:

- if E is a named set, then $\mathsf{id_E}$ is the named function such that, for each $e \in E$, $id_E(e) = e$ and $\mathsf{id_E}[e, e] = \mathrm{id}_{\mathsf{E}[e]}$;
- if $\mathsf{m} : \mathsf{E}_1 \to \mathsf{E}_2$ and $\mathsf{m}' : \mathsf{E}_2 \to \mathsf{E}_3$ are two named functions, then $\mathsf{m}; \mathsf{m}' : \mathsf{E}_1 \to \mathsf{E}_3$ is the named function such that, for each $e \in E_1$, $m; m'(e) = m'(m(e))$ and, if $m(e) = e'$ and $m'(e') = e''$ then $(\mathsf{m}; \mathsf{m}')[e, e''] = \mathsf{m}'[e', e'']; \mathsf{m}[e, e']$.

Definition 7.2 (category of HD-automata) Let Start be the named set with $*$ as singleton element and $\mathsf{Start}[*] = \mathfrak{N}$. A HD-automaton is a diagram

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

in the category **NSet** of named sets. In this case, the initial state q_0 is designated as $i(*)$ and the initial embedding σ_0 is given by $\mathsf{i}[*, q_0]$.

Given two HD-automata $\mathcal{A}_1$ and $\mathcal{A}_2$ on the same named set L of labels, a *morphism* $m : \mathcal{A}_1 \to \mathcal{A}_2$ is a pair of arrows $\mathsf{m}_Q : \mathsf{Q}_1 \to \mathsf{Q}_2$ and $\mathsf{m}_T : \mathsf{T}_1 \to \mathsf{T}_2$ that respect sources, destinations, labels, and the initial states, i.e., such that the two overlapped diagrams

$$
\begin{tikzcd}
 & \mathsf{T}_1 \arrow[dl, "\mathsf{l}_1"'] \arrow[r, bend left, "\mathsf{s}_1"] \arrow[r, bend right, "\mathsf{d}_1"'] \arrow[dd, "\mathsf{m}_T"'] & \mathsf{Q}_1 \arrow[dd, "\mathsf{m}_Q"] & \\
\mathsf{L} & & & \mathsf{Start} \arrow[ul, "\mathsf{i}_1"'] \arrow[dl, "\mathsf{i}_2"] \\
 & \mathsf{T}_2 \arrow[ul, "\mathsf{l}_2"] \arrow[r, bend left, "\mathsf{s}_2"] \arrow[r, bend right, "\mathsf{d}_2"'] & \mathsf{Q}_2 &
\end{tikzcd}
$$

commute in the obvious way.
The category $\mathbf{HD}_{\mathsf{L}}$ has as objects the HD-automata with labels L that respect the following condition:

$$\mathsf{T}[t] = \mathrm{cod}(\mathsf{s}[t, s(t)]) \cup \mathrm{cod}(\mathsf{l}[t, l(t)]) \text{ for each } t \in T;$$

it has as arrows the morphisms between such HD-automata; identity arrows and composition between arrows are defined in the obvious way.
Let Γ be a sorting for L; the category $\mathbf{HD}_{\mathsf{L},\Gamma}$ of the HD-automata on L which are well-sorted with respect to Γ is the full subcategory of $\mathbf{HD}_{\mathsf{L}}$ whose objects are the HD-automata which respect the following condition:

for each $t \in T$ and $n \in \mathsf{L}[l(t)]$, if $\Gamma_{l(t)}(n) = \mathtt{old}$ then $\mathsf{l}[t, l(t)](n) \in \mathrm{cod}(\mathsf{s}[t, s(t)])$ and if $\Gamma_{l(t)}(n) = \mathtt{new}$ then $\mathsf{l}[t, l(t)](n) \notin \mathrm{cod}(\mathsf{s}[t, s(t)])$.

7.2 Open maps and bisimulations

Consider a morphism $m : A_1 \to A_2$ 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 A_1 and A_2. In fact, assume $q_1 \; \mathcal{R} \; q_2$ and $t_1 : q_1 \xrightarrow{\lambda} q_1'$; then we have $t_2 : q_2 \xrightarrow{\lambda} q_2'$ and $q_1' \; \mathcal{R} \; q_2'$ by taking $t_2 = m_T(t_1)$ and $q_2' = m_Q(q_1')$. Moreover $q_{01} \; \mathcal{R} \; q_{02}$.

Therefore, a morphism $m : A_1 \to A_2$ expresses the fact that all the transitions of A_1 can be simulated in A_2, starting from the initial states. In general, however, it is not true that all the transitions of A_2 can be simulated in A_1.

However, it is possible to define a particular class of "bisimulation" morphisms, such that the existence of such a morphism from A_1 to A_2 guarantees not only that the transitions of A_1 can be adequately simulated in A_2 but also the converse; i.e., the existence of a "bisimulation" morphism guarantees that A_1 and A_2 are bisimilar. In general, it is not true the converse, i.e., there exist bisimilar automata A_1 and A_2 such that no "bisimulation" morphism (nor generic morphisms) can be found between them. However, whenever two automata A_1 and A_2 are bisimilar, it is possible to find a common predecessor A and a span of "bisimulation" morphisms $m_1 : A \to A_1$ and $m_2 : A \to A_2$ between them:

$$
\begin{tikzcd}
 & A \arrow[dl, "m_1"'] \arrow[dr, "m_2"] & \\
A_1 & & A_2
\end{tikzcd}
$$

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 [9, 36], *zig-zag morphisms* in [3], *transition preserving homomorphisms* in [18, 19], and *open maps* in [24]. Here we just consider the approach of open maps; the approach of [24] has the advantage that it is 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 $\mathbf{M}$:

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

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 (which equates y via m) but we can make this sure by requiring that there is an arrow y' such that the diagram

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

commutes. Given m, 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 7.3 (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 [24] 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 2.5.

7.3 Application to the HD-automata

In the case of 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 7.4 (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 $t \in T$, $\mathsf{d}[t, d(t)] : \mathsf{Q}[d(t)] \hookrightarrow \mathsf{T}[t]$ is bijective.

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., $\mathsf{m_Q}[q, m_Q(q)]$ 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* **Exp** *of HD-experiments* is the subcategory of **HD** with HD-experiments as objects and name preserving morphisms as arrows.
Category $\mathbf{Exp}_{\mathsf{L},\Gamma}$ is the full subcategory of **Exp** whose objects are $\mathbf{HD}_{\mathsf{L},\Gamma}$-automata.

Now we show that the notion of HD-bisimulation given in Definition 5.1 is also obtained by the uniform technique of the open maps.

Theorem 7.5 *Two well-sorted HD-automata on the same labels L and sorting Γ are HD-bisimilar iff they are open-bisimilar w.r.t. experiments* $\mathbf{Exp}_{\mathsf{L},\Gamma}$.

Proof (sketch) Let $m : \mathcal{A} \to \mathcal{B}$ be an **Exp**-open map. We can prove that A and B are HD-bisimilar by showing that

$$\mathcal{R} = \{\langle q_a, \delta, q_b \rangle \mid q_a \text{ reachable}, q_b = m_Q(q_a), \delta = \mathsf{m_Q}[q_a, q_b]^{-1}\}$$

is a HD-bisimulation for $\mathcal{A}$ and $\mathcal{B}$; so $\mathcal{A}$ and $\mathcal{B}$ are HD-bisimilar.
From this, by transitivity of $\sim$, it is easy to conclude that two HD-automata are HD-bisimilar if there is a span of open maps for them.
For the converse, we show that, if $\mathcal{A}$ and $\mathcal{B}$ are HD-bisimilar via the HD-bisimulation $\mathcal{R}$, then there exists a HD-automaton $\mathcal{A} \otimes \mathcal{B}$ and a span of open maps:

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

Essentially, the states of the HD-automaton $\mathcal{A} \otimes \mathcal{B}$ are the triples of $\mathcal{R}$ and the names of a state $\langle q_a, \delta, q_b \rangle$ are the union of the names of q_a and q_b, where names related by δ are identified; the transitions from a state $\langle q_a, \delta, q_b \rangle$ correspond to all the transitions from q_a and to all those from q_b.
The proof is developed in the details in Appendix A.8. $\square$

8 Reusing algorithms on HD-automata

In this section we show that the theory presented in Section 5 can be applied to develop algorithms that check bisimulation equivalence for history dependent formalisms.

According to the theory, to check whether two π-calculus agents p_1 and p_2 are bisimilar, we can check whether $\mathtt{unf}(\Downarrow\mathcal{A}_{p_1}) \sim \mathtt{unf}(\Downarrow\mathcal{A}_{p_2})$. Hence, to check the bisimulation of the two agents, we build the two unfoldings and then reuse on them the classical algorithms for bisimulation equivalence on ordinary automata [37, 25]. The possibility of reusing classical algorithms has been particularly convenient for a fast development of a verification environment for history dependent formalisms [17].

The construction of the HD-automata $\mathcal{A}_{p_1}$ and $\mathcal{A}_{p_2}$ can be done effectively whenever the agents p_1 and p_2 are finitary: in fact, Definition 4.6 can be translated directly in an algorithm for building HD-automata, and Theorem 4.9 assures that the algorithm ends whenever the agents are finitary.

Also the unfolding of a HD-automaton can be done effectively, and the construction terminates if the starting HD-automaton is finite.

The problematic step is the construction of the irredundant HD-automata $\Downarrow\mathcal{A}_{p_1}$ and $\Downarrow\mathcal{A}_{p_2}$. In fact, the definition of irredundant HD-automata requires to know the active names and, in general, an algorithm that decides whether a name is inactive in a state of a HD-automaton is as difficult as an algorithm for checking whether two HD-automata are bisimilar. Hence, even if it is possible in theory to compute the active names of a generic finite HD-automaton[8], this would require to develop complex algorithms on the HD-automata, which is not our aim.

[8] To compute the active names it is sufficient to compute the greatest HD-bisimulation of a HD-automaton. The greatest bisimulation of a finite HD-automaton can be effectively build: there are just a finite number of relations that can be potential bisimulation and each of then can be checked in a finite amount of time. There are obviously also better algorithms for building the greatest bisimulation of a HD-automaton.

We prefer to give a much more efficient characterization of active names — and hence of the irredundant reduction — which however works only for a subset of finite HD-automata. To this purpose, we exploit the idea of *observed name*. Essentially, a name in a state of a HD-automaton is observed if there is some computation starting from that state such that the name is observed in the labels of this computation.

Definition 8.1 (observed names) Let $t_i = q_i \xrightarrow{\lambda_i} q_{i+1}$ for $i = 1 \dots k$ be a computation of a HD-automaton $\mathcal{A}$ and let $n_i \in \mathsf{T}[t_i]$ and $m_i \in \mathsf{Q}[q_i]$ for $i = 1 \dots k$ be names of $\mathcal{A}$ that respect the following conditions:

1. $m_i = \mathsf{s}[t_i, q_i](n_i)$ and $m_{i+1} = \mathsf{d}[t_i, q_{i+1}](n_i)$ for $i = 1 \dots k$;
2. $m_i \notin \mathrm{cod}(\mathsf{l}[t_i, \lambda_i])$ for $i = 1 \dots k-1$;
3. $m_k \in \mathsf{T}[t_k]_{\mathtt{old}}$.

Then we say that name n_1 is *observed* from q_1.
The set of the observed names of a state q are denoted by $\mathsf{Q}[q]_{\mathrm{obs}}$.

All the names n_i and m_i are the "local" versions of name n_1 in q_1 (Condition 1). We require that the name occurs in the label λ_k of the computation as a name of sort `old` (Condition 3); names which appear with sort `new` and `univ` are not considered as observed, since both sorts are used to model the creation of new names. Condition 2 assures that the name does not appear in any label of the computation before λ_k.

Notice that, in the conditions of the definition, it is not only true that m_1 is observed from q_1: more in general, it holds that m_i is observed from q_i for all $i = 1 \dots k$.

It is easy to show that all the observed names are also active, i.e., that $\mathsf{Q}[q]_{\mathrm{obs}} \subseteq \mathsf{Q}[q]_{\mathrm{active}}$. Unfortunately, in general the converse is not true: there are active names which are not observed. However, there are interesting cases in which all the active names are also observed: this is true for instance of all those HD-automata without names of sort `both`. The HD-automata obtained from CCS with localities and Petri nets fall in this category:

Proposition 8.2 *Let $\mathcal{A}$ be the HD-automaton corresponding to a CCS agent with localities (Definition 6.7) or to a Petri net (Definition 6.26). Then the active names and the observed names coincide in all the states of $\mathcal{A}$.*

However, this is also true for categories of HD-automata where names of `both` are used.

Proposition 8.3 *Let $\mathcal{A}$ be the HD-automaton corresponding to an agent in the fragment of π-calculus without matching. Then the active names and the observed names coincide in all the states of $\mathcal{A}$.*

The advantage of observed names it that they can be computed via a simple algorithm that applies, for each transition $t : q \xrightarrow{\lambda} q'$, the following two rules:

Rule A if $\mathsf{s}[t, q](n) \in \mathsf{T}[t]_{\mathtt{old}}$ then $n \in \mathsf{Q}[q]_{\mathrm{obs}}$;

Rule B if $\mathsf{s}[t, q](n) \in \mathsf{d}[t, q'](\mathsf{Q}[q']_{\mathrm{obs}}) \setminus \mathrm{cod}(\mathsf{l}[t, \lambda])$ then $n \in \mathsf{Q}[q]_{\mathrm{obs}}$.

The following two rules are just an inductive rewriting of Definition 8.1; in fact, **Rule A** assures that m_k is observed from q_k and an iterative application of **Rule B** assures that m_i is observed form q_i, for $i = k-1, k-2, \dots, 1$.

Whenever this efficient technique can be applied for discovering the active names, the following algorithm can be used for checking history-preserving equivalence of two history dependent systems:

1. construct (separately) the HD-automata corresponding to the systems;
2. discover (separately) the active names of the two automata and get the irredundant reductions: start marking the names that are active due to **Rule A** above and continue marking all the names reachable following the dependencies in **Rule B**; at the end discard the unmarked names and the corresponding inactive transitions;
3. unfold (separately) the obtained irredundant HD-automata;

4. use a standard algorithm for checking bisimulation equivalence of the obtained ordinary automata (for instance, partition refinement [25, 37]).

Notice that, while step 1 depends on the formalism in which the systems are described, steps 2–4 work for generic HD-automata and are hence common to all these formalisms. To add a new formalism, moreover, it is sufficient to define a new function which maps systems described in this new formalism into HD-automata; obviously, this function must map equivalent systems into bisimulation equivalent HD-automata.

The proposed algorithm can also be used to generate the minimal automaton corresponding to a system; to obtain this, the same procedure has to be applied by starting with just a system and, at the end, a minimization algorithm has to be applied.

9 Possible extensions and concluding remarks

In this section we discuss some directions in which we plan to extend the approach described in this 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.

9.1 Alternative semantics for the π-calculus

In Section 3 we have presented the *early* semantics of the π-calculus. However, this is not the only semantics that has been proposed for this language. The semantics that was proposed from the very beginning for the π-calculus [30], for instance, was *late*. The difference 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[9]. Agent $p = a(x).p'$ performs transition

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

where name x is the placeholder (hence the usage of the parenthesis in the label). 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

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

We denote with $\sim_{\text{late}}$ the late bisimulation equivalence.

In [20] 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 modelled 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 [20] it is proved that this approach coincides with the original late semantics of [30].

We can exploit this idea also to capture the late semantics of the π-calculus within the framework of the HD-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 name n_{sub} of sort $\mathtt{old}$ is associated to label $\mathtt{in}_{\mathtt{start}}$, whereas one name n_{obj} of sort $\mathtt{both}$ is associated to label $\mathtt{in}_{\mathtt{end}}$. Also the definition of the mapping from π-calculus agents to HD-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 HD-automata corresponding to agent p is finite whenever agent p is finitary; moreover:

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

Theorem 9.1 *Let p and q be two π-calculus agents and let $\mathcal{A}_p$ and $\mathcal{A}_q$ be the corresponding HD-automata, according to the late translation. Then $p \sim_{\mathrm{late}} q$ iff $\mathcal{A}_p \sim \mathcal{A}_q$.*

The *open* semantics is yet a different semantics for the π-calculus; it was introduced in [42]. In this case the names can be seen as variables rather than as constants and a transitions can occur only under particular constraints on these variables. The definition of bisimulation is quite different from the classical one, since it has to deal with the fact that a transition which can occur under a certain constraint can be matched by a transition which can occur under a looser constraint.

We do not foresee how open semantics could be captured by HD-automata. In fact, these do not have the features to deal with constraints in the transitions. However, in [39] Davide Sangiorgi and the second author have shown that also the open semantics can be characterized as an ordinary bisimulation on ordinary automata. There, global names are used in states and transitions and therefore technical difficulties arise in the choice of the fresh names for input and bound output transitions. It would be interesting to define an extended version of HD-automata that can deal with constraints in the transitions; the results of [39] could be reformulated on these enriched HD-automata, with the advantage of using local names.

HD-automata apply also to the asynchronous π-calculus [22]; however the encoding is not satisfactory from a practical point of view, since in this setting an agent can receive a message on any channel at any moment. We are studying more efficient encodings, that exploit the alternative characterizations of asynchronous bisimulation of [2].

9.2 The weak semantics

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

Differently from the strong approach, where τ actions are managed 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:

- $\stackrel{\epsilon}{\Rightarrow} = (\stackrel{\tau}{\rightarrow})^*$, and
- $\stackrel{\alpha}{\Rightarrow} = \stackrel{\epsilon}{\Rightarrow}\stackrel{\alpha}{\rightarrow}\stackrel{\epsilon}{\Rightarrow}$ for $\alpha \neq \tau$.

Notice that $p \stackrel{\epsilon}{\Rightarrow} p'$ 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 behavioural 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 [37, 25] — so the standard algorithms can be used also in this case.

This approach can also be followed for the history-dependent calculi. Weak transitions can be defined on the HD-automata: we assume that in the labels of the HD-automaton there is a distinguished action τ with no associated names; we can define weak transitions as we did for the ordinary automata, but in this case we have to deal with names. For instance, if a weak transition $t : q \stackrel{\epsilon}{\Rightarrow} q''$ is obtained from the two transitions $t' : q \stackrel{\tau}{\rightarrow} q'$ and $t'' : q' \stackrel{\tau}{\rightarrow} q''$, then we would have $\mathsf{T}[t] = \mathsf{T}[t']$, $\mathsf{s}[t,q] = \mathsf{s}[t',q]$ and $\mathsf{d}[t,q''] = \mathsf{d}[t'',q'']; \mathsf{s}[t',q']^{-1}; \mathsf{d}[t,q']$.

Also in this case, once the strong HD-automaton is built, it is possible to compute the weak transitions, thus obtaining a weak HD-automaton.

A weak ordinary automaton can then be obtained by unfolding the (previously made irredundant) weak HD-automaton and the algorithms described in Section 8 can be applied also for the weak semantics.

In this way, the weak semantics of π-calculus and of CCS with localities is captured by the HD-bisimulation by just changing the translation from agents to HD-automata.

Notice that a "weak" ordinary automaton can be obtained for a history-dependent formalism by following a different sequence of steps: it is possible to build the *strong* HD-automaton, make it irredundant, and unfold it, thus obtaining the *strong* ordinary automaton, and finally to transform this one in a *weak* automaton. In this way, however, wrong equivalences are obtained. In fact, the names of an agent which are active w.r.t. the strong semantics can be different form the active names w.r.t. the weak semantics. For instance, agents

$$p = a(x).[x = b]\tau.\mathbf{0} \qquad \text{and} \qquad q = a(x).\mathbf{0}$$

are weak bisimilar and name b is inactive in the first agent according to the weak semantics. However, they are not equivalent according to the strong semantics, and name b is active in this case. Therefore, it is used as input value in the *strong* ordinary automaton corresponding to the first agent, and it is still present as input value also when this automaton is transformed in the weak one, thus leading to a wrong weak ordinary automaton for p.

9.3 CCS with causality

In Section 6.1 we have presented the location semantics of CCS as a way to define a truly concurrent behavioural 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 [14] 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 [13, 26] 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 [26], which is similar to the location approach of Section 6.1. Also here we extend the language with prefixes of the form $c :: p$; in this case, 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 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 to discard unused causes. A mapping to HD-automata can now be defined and the HD-automata are finite whenever the agents are finitary. Moreover, HD-bisimulation exactly matches causal bisimulation:

Theorem 9.2 *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 $\mathcal{A}_1 \sim \mathcal{A}_2$.*

9.4 Causality/localities and the π-calculus

In [43] a localities semantics was proposed for the π-calculus by using an approach very similar to the one described in Section 6.1 for CCS. Moreover, it was also shown 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 [5] similar results were obtained for the π-calculus with causality. Thus, the usage of names in the π-calculus is general enough to represent the tree structure of localities and the partial order structure of the causes. However, the encodings defined in [43] and in [5] do not preserve finitary agents: the obtained agents are not finitary whenever the starting agents can perform infinite computations.

HD-automata can be exploited also to capture the localities and the causal semantics of π-calculus. In fact, it is sufficient to combine the techniques described in Section 4.2 and in Sections 6.2 and 6.4 respectively for the π-calculus names and for the locations or causes. The advantage of this approach w.r.t. the embeddings of [43, 5] is that the obtained HD-automata are finite whenever the starting π-calculus agents are finitary.

9.5 A verification environment based on HD-automata

A verification environment is being developed, that exploits the features of HD-automata to verify history dependent formalisms. The logical structure of the verification environment is illustrated in Figure 2. It provides a number of front ends that translate several formalisms into HD-automata, and a set of tools on HD-automata to edit, visualize, compose and check them. HD-automata can also be translated into ordinary automata and an existing environment for process calculi, the JACK systems [6], is used to work on the ordinary automata (equivalence checking and minimization). Moreover, a model checker for verifying logical properties of systems has also been implemented. The model checker allows the user to check behavioral properties (expressed in a variant of Hennessy-Milner logic) directly on HD-automata, or to translate them in equivalent properties for the ordinary automata. A preliminary report on the development of the verification environment appeared in [17].

9.6 Concluding remarks

In this paper we have introduced HD-automata as an operational model adequate to deal with history-dependent calculi. The two results we have stressed are the following:

- the possibility of representing history-preserving formalisms via HD-automata and the bisimulation equivalences defined for these formalisms via a general definition of bisimulation equivalence on HD-automata; and
- the possibility of defining HD-automata and HD-bisimulation via uniform categorical techniques. Notice that the categorical definition of HD-bisimulation avoids the cumbersome case analysis of the rôle of names that appears in Definition 5.1.

Besides these results, we have also shown that the theory above can be applied to develop algorithms for checking significant subsets of the history-dependent calculi.

[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.

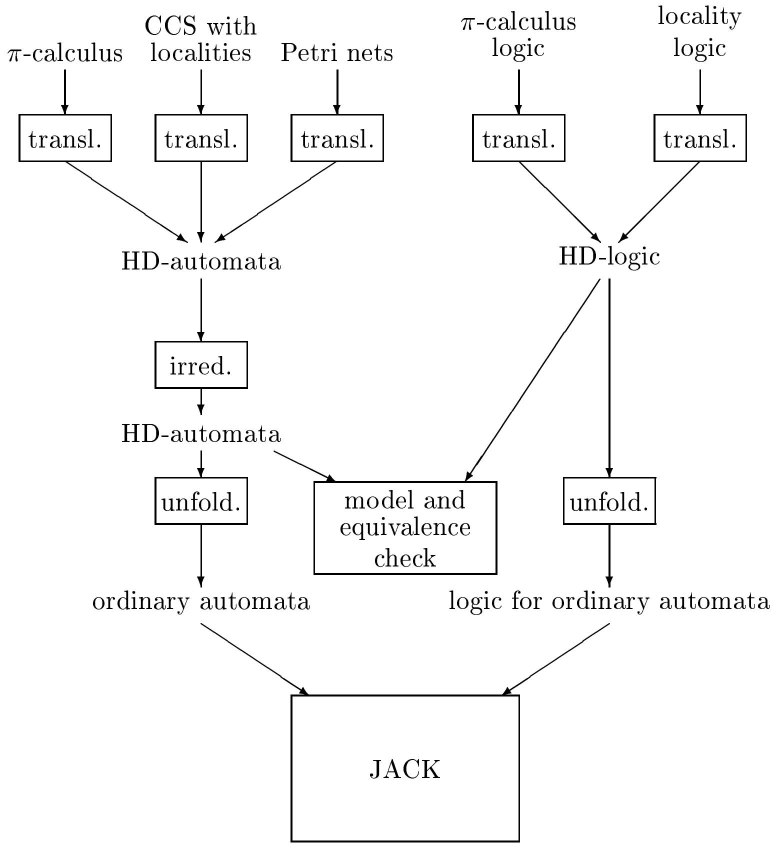

Figure 2: Structure of the verification environment.

Moreover, the approach is sufficiently general to be applied also to other formalisms: the fundamental steps are the following:

- Identify the "entities" of the formalism that should be mapped into the names of the HD-automata: these are the channel names for the π-calculus, the locations for CCS with localities, and the events for the history preserving semantics of Petri nets.

- Define an "incremental" semantics of the formalism that allows for an efficient reusage of the "entities": this is already done in the ordinary semantics of the π-calculus, since channel names can be discarded during a computation when they do not appear in the destination state of a transition; in the case of CCS with localities, we had to define an incremental semantics which allows us to discard locations, and in the case of Petri nets, we had to move from processes to configurations. The obtained semantics should be independent from the particular choice of "entities", i.e., it should be possible to work up to bijective renamings and to define an appropriate function `norm`.

  Notice that the reusage of "entities" has to be efficient — so that a significant subset of the programs is mapped into finite HD-automata — however, it has not to be optimal: there are the active names to this purpose. Notice also that normally the incremental definition of bisimulation has still to deal explicitly with the creation of new "entities", to guarantee that these are fresh for both the states in relation. This is the rôle, for instance, of clause "$n \notin \mathtt{loc}(p, q)$" in Definition 6.3.

- A mapping from the formalism to HD-automata is defined; this exploits function `norm`. A correspondence between the operational semantics of the formalism and the (global) operational semantics of HD-automata is shown. Therefore the coincidence of the bisimulation semantics is derived. The general results on HD-automata can then be exploited to map the formalism on ordinary automata and to define an effective algorithm for verifying bisimulation equivalence.

References

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

[2] R. Amadio, I. Castellani and D. Sangiogi. On bisimulations for the asynchronous π-calculus. In *Proc. CONCUR'96*, LNCS 1119. Springer Verlag, 1996.

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

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

[5] M. Boreale and D. Sangiorgi. A fully abstract semantics of causality in the π-calculus. In *Proc. STACS'95*, LNCS 900. Springer Verlag, 1995.

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

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

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

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

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

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

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

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

[14] P. Degano, R. De Nicola and U. Montanari. CCS is an (augmented) contact free C/E system. In *Proc. Adv. Sch. on Mathematical Models for the Semantics of Parallelism*, LNCS 280. Springer Verlag, 1986.

[15] P. Degano, R. De Nicola and U. Montanari. Partial orderings descriptions and observations of nondeterministic concurrent processes. In *Proc. REX School/Workshop on Linear Time, Branching Time and Partial Orders in Logica and Models for Concurrency*, LNCS 354. Springer Verlag, 1989.

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

[17] 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*, LNCS 1217. Springer Verlag, 1997.

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

[19] 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.

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

[21] C.A.R. Hoare, S.D. Brookes and A.W. Roscoe. A theory of communicating processes. *Journal of ACM*, 31(3):560–599, 1984.

[22] K. Honda and M. Tokoro. On asynchronous communication semantics. In *Proc. ECOOP'91*, LNCS 612. Springer Verlag, 1992.

[23] P. Inverardi and C. Priami. Evaluation of tools for the analysis of communicating systems. *Bulletin of the EATCS*, 45:158–185, 1991.

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

[25] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

[39] M. Pistore and D. Sangiorgi. A partition refinement algorithm for the π-calculus. In *Proc. CAV'96*, LNCS 1102. Spinger Verlag, 1996.

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

[41] D. Sangiorgi. *Expressing Mobility in Process Algebras: First-Order and Higher-Order Paradigms.* PhD Thesis CST-99-93, University of Edinburgh, 1992.

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

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

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

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

A Appendix

In this appendix we carry out the proofs of the results of the paper.

A.1 Proofs for Section 5.1

Lemma A.1 *Let $\langle q_1, \delta, q_2\rangle \in \mathcal{R}$ for some HD-simulation $\mathcal{R}$ and let $t_1 : q_1 \xrightarrow{\lambda} q_1'$, $t_2 : q_2 \xrightarrow{\lambda} q_2'$ and $\zeta : \mathsf{T}_1[t_1] \longrightarrow \mathsf{T}_2[t_2]$ satisfy all the conditions of Definition 5.1.*

Let $n_1 \in \mathsf{T}_1[t_1]$ and $n_2 \in \mathsf{T}_2[t_2]$ such that $\zeta(n_1) = n_2$. Then:

- $n_1 \in \mathsf{T}_1[t_1]_{\mathtt{new}}$ *iff* $n_2 \in \mathsf{T}_2[2_2]_{\mathtt{new}}$, *and*
- $n_1 \in \mathsf{T}_1[t_1]_{\mathtt{univ}} \cup \mathsf{T}_1[t_1]_{\mathtt{src}}$ *iff* $n_2 \in \mathsf{T}_2[2_2]_{\mathtt{univ}} \cup \mathsf{T}_2[t_2]_{\mathtt{src}}$.

Proof The second item is a consequence of the first, since $\mathsf{T}[t]_{\mathtt{new}}$, $\mathsf{T}[t]_{\mathtt{univ}}$ and $\mathsf{T}[t1]_{\mathtt{src}}$ form a partition of $\mathsf{T}[t]$. Now we prove the first item.
By definition, if $n_1 \in \mathsf{T}_1[t_1]_{\mathtt{new}}$, then there is some $n_\lambda \in \mathsf{L}[\lambda]$ such that

$$n_1 = \mathsf{l}_1[t_1, \lambda](n_\lambda) \tag{1}$$

and n_λ has sort `new`.
Now we show that $n_2 = \mathsf{l}_2[t_2, \lambda](n_\lambda)$:

$$\begin{aligned} n_2 &= \zeta(n_1) && \text{by hypothesis} \\ &= \mathsf{l}_1[t_1, \lambda]; \zeta(n_\lambda) && \text{by (1)} \\ &= \mathsf{l}_2[t_2, \lambda](n_\lambda) && \text{by Definition 5.1.} \end{aligned}$$

Since n_λ has sort `new`, it must be $n_2 \in \mathsf{T}_2[t_2]_{\mathtt{new}}$. □

Proposition 5.4 *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 $\mathcal{A}_3$.*

Proof We show that if $\mathcal{R}$ and $\mathcal{S}$ are HD-simulations then $\mathcal{R} \frown \mathcal{S}$ is a HD-simulation.
Suppose $\langle q_1, \delta, q_3\rangle \in \mathcal{R} \frown \mathcal{S}$ and suppose we fix $t_1 : q_1 \xrightarrow{\lambda} q_1'$ and $\xi : \mathsf{T}_1[t_1]_{\mathtt{univ}} \longrightarrow \mathsf{Q}_3[q_3]$ so that

$$\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset. \tag{2}$$

Since $\langle q_1, \delta, q_3\rangle \in \mathcal{R} \frown \mathcal{S}$, we know that 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 $\delta = \delta'; \delta''$.
Let $\xi' : \mathsf{T}_1[t_1]_{\mathtt{univ}} \longrightarrow \mathsf{Q}_2[q_2]$ be defined as follows:

$$\xi' \stackrel{\text{def}}{=} \xi; \delta''^{-1}. \tag{3}$$

It holds that $\mathrm{cod}(\xi') \cap \mathrm{cod}(\delta') = \emptyset$. In fact, suppose $n \in \mathrm{cod}(\xi') \cap \mathrm{cod}(\delta')$; by definition of ξ', $n \in \mathrm{cod}(\xi; \delta''^{-1})$ and so $\delta''(n)$ is defined and is in $\mathrm{cod}(\xi)$; since $n \in \mathrm{cod}(\delta')$ we also have $\delta''(n) \in \mathrm{cod}(\delta'; \delta'') = \mathrm{cod}(\delta)$; so we have $\delta''(n) \in \mathrm{cod}(\xi) \cap \mathrm{cod}(\delta)$, which is absurd by (2). This concludes the proof that $\mathrm{cod}(\xi') \cap \mathrm{cod}(\delta') = \emptyset$
Since $\mathcal{R}$ is a HD-simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ and $\zeta' : \mathsf{T}_1[t_1] \longrightarrow \mathsf{T}_2[t_2]$ such that:

$$\delta' = \mathsf{s}_1[t_1, q_1]; \zeta'; \mathsf{s}_2[t_2, q_2]^{-1} \tag{4}$$

$$\xi' = \zeta'|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}}; \mathsf{s}_2[t_2, q_2]^{-1} \tag{5}$$

$$\mathsf{l}_1[t_1, \lambda]; \zeta' = \mathsf{l}_2[t_2, \lambda] \tag{6}$$

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

$$\gamma' \subseteq \mathsf{d}_1[t_1, q_1']; \zeta'; \mathsf{d}_2[t_2, q_2']^{-1}. \tag{7}$$

Now, let us define $\xi'' : \mathsf{T}_2[t_2]_{\mathtt{univ}} \hookrightarrow \mathsf{Q}_3[q_3]$ as follows:

$$\xi'' \stackrel{\text{def}}{=} {\zeta'}^{-1}|_{\mathsf{T}_2[t_2]_{\mathtt{univ}}}; \xi. \tag{8}$$

It holds that

$$\mathrm{cod}(\delta'') \cap \mathrm{cod}(\xi'') = \emptyset. \tag{9}$$

In fact, suppose $n \in \mathrm{cod}(\delta'') \cap \mathrm{cod}(\xi'')$. Then, $n \in \mathrm{cod}(\xi'')$ would imply $n = \xi''(m)$ for some $m \in \mathsf{T}_2[t_2]_{\mathtt{univ}}$. Hence, by (8), $n = {\zeta'}^{-1}; \xi(m)$. Moreover $n \in \mathrm{cod}(\delta'')$ and hence ${\zeta'}^{-1}; \xi; {\delta''}^{-1}(m)$ would be also defined. Now,

$$\begin{aligned}
{\zeta'}^{-1}; \xi; {\delta''}^{-1}(m) &= {\zeta'}^{-1}; \xi'(m) && \xi; {\delta''}^{-1} = \xi' \text{ by } (3)\\
&= {\zeta'}^{-1}; \zeta'; \mathsf{s}_2[t_2, q_2]^{-1}(m) && \xi' = \zeta'|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}}; \mathsf{s}_2[t_2, q_2]^{-1} \text{ by } (5)\\
&= \mathsf{s}_2[t_2, q_2]^{-1}(m) && \text{since } \zeta^{-1}; \zeta(m) = m.
\end{aligned}$$

Hence $\mathsf{s}_2[t_2, q_2]^{-1}(m)$ would be defined, which is absurd, since $m \in \mathsf{T}_2[t_2]_{\mathtt{univ}}$. This concludes the proof of (9).

Since $\mathcal{S}$ is a HD-simulation for $\mathcal{A}_2$ and $\mathcal{A}_3$, there are some $t_3 : q_3 \xrightarrow{\lambda} q_3'$ and $\zeta'' : \mathsf{T}_2[t_2] \hookrightarrow \mathsf{T}_3[t_3]$ such that:

$$\delta'' = \mathsf{s}_2[t_2, q_2]; \zeta''; \mathsf{s}_3[t_3, q_3]^{-1} \tag{10}$$

$$\xi'' = \zeta''|_{\mathsf{T}_2[t_2]_{\mathtt{univ}}}; \mathsf{s}_3[t_3, q_3]^{-1} \tag{11}$$

$$\mathsf{l}_2[t_2, \lambda]; \zeta'' = \mathsf{l}_3[t_3, \lambda] \tag{12}$$

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

$$\gamma'' \subseteq \mathsf{d}_2[t_2, q_2']; \zeta''; \mathsf{d}_3[t_3, q_3']^{-1}. \tag{13}$$

Let us define $\zeta : \mathsf{T}_1[t_1] \hookrightarrow \mathsf{T}_3[t_3]$ as follows:

$$\zeta \stackrel{\text{def}}{=} \zeta'; \zeta''. \tag{14}$$

Fact. If $n_1 \in \mathsf{T}_1[t_1]_{\mathtt{src}}$ and $\zeta(n) \in \mathsf{T}_3[t_3]_{\mathtt{src}}$, then also $\zeta'(n_1) \in \mathsf{T}_2[t_2]_{\mathtt{src}}$.

Proof of the fact Let $m = \zeta'(n_1)$. By (14), $\zeta''(n_2) = \zeta(n_1)$.
Since $\zeta''(n_2) \in \mathsf{T}_3[t_3]_{\mathtt{src}}$, due to Lemma A.1, it holds $n_2 \notin \mathsf{T}_2[t_2]_{\mathtt{new}}$.
Assume now $n_2 \in \mathsf{T}_2[t_2]_{\mathtt{univ}}$. Since $\zeta''(n_2) \in \mathsf{T}_3[t_3]_{\mathtt{src}}$ and since (11), $\xi''(n_2)$ would be defined and

$$\xi''(n_2) = \zeta''; \mathsf{s}_3[t_3, q_3](n_2).$$

If $\xi''(n_2)$ were defined, by (8) also ${\zeta'}^{-1}; \xi(n_2)$ would be defined. Now, ${\zeta'}^{-1}(n_2) = n$, so $\xi(n_1)$ would be also defined, which would imply $n_1 \in \mathsf{T}_1[t_1]_{\mathtt{univ}}$. This is impossible, since $n_1 \in \mathsf{T}_1[t_1]_{\mathtt{src}}$ by hypothesis. Therefore, it cannot be $n_2 \in \mathsf{T}_2[t_2]_{\mathtt{univ}}$.
Since $n_2 \notin \mathsf{T}_2[t_2]_{\mathtt{new}}$ and $n_2 \notin \mathsf{T}_2[t_2]_{\mathtt{univ}}$, it must be $n_2 \in \mathsf{T}_2[t_2]_{\mathtt{src}}$. **(Fact)**

Now we are ready to show that transition t_3 satisfies all the condition of Definition 5.1 w.r.t. transition t_1 and partial correspondence ξ:

- $\delta = \mathsf{s}_1[t_1, q_1]; \zeta; \mathsf{s}_3[t_3, q_3]^{-1}$. In fact, let $m \in \mathsf{Q}_1[q_1]$; then

$$\begin{aligned}
\delta(m) &= \mathsf{s}_1[t_1, q_1]; \zeta'; \mathsf{s}_2[t_2, q_2]^{-1}; \mathsf{s}_2[t_2, q_2]; \zeta''; \mathsf{s}_3[t_3, q_3]^{-1}(m) && \text{by } \delta = \delta'; \delta'', (4) \text{ and } (10)\\
&= \mathsf{s}_1[t_1, q_1]; \zeta'; \zeta''; \mathsf{s}_3[t_3, q_3]^{-1}(m) && \text{since } \mathsf{s}_1[t_1, q_1]; \zeta'; \mathsf{s}_2[t_2, q_2]^{-1}(m)\\
& && \text{is defined by Fact}\\
&= \mathsf{s}_1[t_1, q_1]; \zeta; \mathsf{s}_3[t_3, q_3]^{-1}(m) && \text{by } (14)\ \zeta'; \zeta'' = \zeta.
\end{aligned}$$

- $\xi = \zeta|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}}; \mathsf{s}_3[t_3,q_3]^{-1}$. In fact, consider $n \in \mathsf{T}_1[t_1]_{\mathtt{univ}}$; by Lemma A.1 $\zeta'(n)$ is defined and either $\zeta'(n) \in \mathsf{T}_2[t_2]_{\mathtt{univ}}$, or $\zeta'(n) \in \mathsf{T}_2[t_2]_{\mathtt{src}}$. In the first case,

$$\begin{aligned} \zeta; \mathsf{s}_3[t_3,q_3]^{-1}(n) &= \zeta';\zeta'';\mathsf{s}_3[t_3,q_3]^{-1}(n) && \text{by (14)} \\ &= \zeta';\xi''(n) && \text{by (11), exploiting } \zeta'(n) \in \mathsf{T}_2[t_2]_{\mathtt{univ}} \\ &= \zeta';\zeta'^{-1};\xi(n) = \xi(n) && \text{by (8).} \end{aligned}$$

  In the second case,

$$\begin{aligned} \zeta; \mathsf{s}_3[t_3,q_3]^{-1}(n) &= \zeta';\zeta'';\mathsf{s}_3[t_3,q_3]^{-1}(n) && \text{by (14)} \\ &= \zeta';\mathsf{s}_2[t_2,q_2]^{-1};\delta''(n) && \text{by (10)} \\ &= \xi';\delta''(n) && \text{by (5)} \\ &= \xi;\delta''^{-1};\delta''(n) && \text{by (3)} \\ &= \xi(n) && \text{since } \xi'(n) = \xi;\delta''^{-1}(n) \text{ is defined.} \end{aligned}$$

- $\mathsf{l}_1[t_1,\lambda];\zeta = \mathsf{l}_3[t_3,\lambda]$, since:

$$\begin{aligned} \mathsf{l}_1[t_1,\lambda];\zeta &= \mathsf{l}_1[t_1,\lambda];\zeta';\zeta'' && \text{by (14)} \\ &= \mathsf{l}_2[t_2,\lambda];\zeta'' && \text{by (6)} \\ &= \mathsf{l}_3[t_3,\lambda] && \text{by (12).} \end{aligned}$$

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

$$\begin{aligned} \gamma';\gamma'' &\subseteq \mathsf{d}_1[t_1,q_1'];\zeta';\mathsf{d}_2[t_2,q_2']^{-1};\mathsf{d}_2[t_2,q_2'];\zeta'';\mathsf{d}_3[t_3,q_3']^{-1} && \text{by (7) and (13)} \\ &\subseteq \mathsf{d}_1[t_1,q_1'];\zeta';\zeta'';\mathsf{d}_3[t_3,q_3']^{-1} && \text{since } \mathsf{d}_2[t_2,q_2']^{-1};\mathsf{d}_2[t_2,q_2'] \subseteq \mathrm{id}_{\mathsf{T}_2[t_2]} \\ &= \mathsf{d}_1[t_1,q_1'];\zeta;\mathsf{d}_3[t_3,q_3']^{-1} && \text{by (14).} \end{aligned}$$

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. □

Proposition 5.6 *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 $\mathcal{A}_2$.*

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 $t_1 : q_1 \xrightarrow{\lambda} q_1'$ and that $\xi : \mathsf{T}_1[t_1]_{\mathtt{univ}} \longrightarrow \mathsf{Q}_2[q_2]$ is such that $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$. Then also $\mathrm{cod}(\xi) \cap \mathrm{cod}(\bar{\delta}) = \emptyset$ and, since $\mathcal{R}$ is a HD-bisimulation, there exist $t_2 : q_2 \xrightarrow{\lambda} q_2'$ and $\bar{\zeta} : \mathsf{T}_1[t_1] \longrightarrow \mathsf{T}_2[t_2]$ such that

$$\bar{\delta} = \mathsf{s}_1[t_1,q_1];\bar{\zeta};\mathsf{s}_2[t_2,q_2]^{-1} \tag{15}$$

$$\xi = \bar{\zeta}|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}};\mathsf{s}_2[t_2,q_2]^{-1} \tag{16}$$

$$\mathsf{l}_1[t_1,\lambda];\bar{\zeta} = \mathsf{l}_2[t_2,\lambda] \tag{17}$$

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

$$\delta' \subseteq \mathsf{d}_1[t_1,q_1'];\bar{\zeta};\mathsf{d}_2[t_2,q_2']^{-1}. \tag{18}$$

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

$$\zeta(n) \stackrel{\text{def}}{=} \begin{cases} \mathsf{s}_1[t_1,q_1];\delta;\mathsf{s}_2[t_2,q_2]^{-1}(n) & \text{if } n \in \mathsf{T}_1[t_1]_{\mathtt{src}} \\ \bar{\zeta}(n) & \text{if } n \in \mathsf{T}_1[t_1]_{\mathtt{univ}} \cup \mathsf{T}_1[t_1]_{\mathtt{new}} \end{cases} \tag{19}$$

Since $\bar{\delta} \subseteq \delta$ we also have $\bar{\zeta} \subseteq \zeta$. Then:

- $\delta = \mathsf{s}_1[t_1, q_1]; \zeta; \mathsf{s}_2[t_2, q_2]^{-1}$ by (19);
- $\xi = \zeta|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}}; \mathsf{s}_2[t_2, q_2]^{-1}$ by (16) and (19);
- $\mathsf{l}_1[t_1, \lambda]; \zeta = \mathsf{l}_2[t_2, \lambda]$; in fact, $\mathsf{l}_1[t_1, \lambda]; \zeta \subseteq \mathsf{l}_1[t_1, \lambda]; \bar{\zeta}$ by (19) and $\mathsf{l}_1[t_1, \lambda]; \bar{\zeta} = \mathsf{l}_2[t_2, \lambda]$ by (18), so $\mathsf{l}_1[t_1, \lambda]; \zeta \subseteq \mathsf{l}_2[t_2, \lambda]$; since the function $\mathsf{l}_2[t_2, \lambda]$ is a total map, it must be $\mathsf{l}_1[t_1, \lambda]; \zeta = \mathsf{l}_2[t_2, \lambda]$;
- $\langle q_1', \delta', q_2' \rangle \in \widehat{\mathcal{R}}$ holds since $\langle q_1', \delta', q_2' \rangle \in \mathcal{R}$ and $\mathcal{R} \subseteq \widehat{\mathcal{R}}$; finally, $\delta' \subseteq \mathsf{d}_1[t_1, q_1']; \zeta; \mathsf{d}_2[t_2, q_2']^{-1}$ by (18) and (19).

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. $\square$

A.2 Proof of Proposition 5.10

We should prove that two HD-automata are global-bisimilar iff they are HD-bisimilar. This is done in the following two lemmas.

Lemma A.2 *If $\mathcal{R}$ is a HD-simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, then*

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

is a global simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$.

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

$$\rho_1(\mathsf{T}_1[t_1]_{\mathtt{new}}) \cap \sigma_2(\mathsf{Q}_2[q_2]) = \emptyset. \tag{20}$$

We have to show that there is some $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(\lambda, \kappa)} (q_2', \sigma_2')$ with $\mathsf{l}_1[t_1, \lambda]; \rho_1 = \mathsf{l}_2[t_2, \lambda]; \rho_2$ and $(q_1', \sigma_1')\ \mathcal{R}'\ (q_2', \sigma_2')$.
From $(q_1, \sigma_1)\ \mathcal{R}'\ (q_2, \sigma_2)$ we obtain, by definition of $\mathcal{R}'$, that $\langle q_1, \delta, q_2 \rangle \in \widehat{\mathcal{R}}$ with

$$\delta = \sigma_1; \sigma_2^{-1}. \tag{21}$$

From $(t_1, \rho_1) : (q_1, \sigma_1) \xrightarrow{(\lambda, \kappa)} (q_1', \sigma_1')$ by definition of global transition we obtain $t_1 : q_1 \xrightarrow{\lambda} q_1'$ and:

$$\sigma_1 = \mathsf{s}_1[t_1, q_1]; \rho_1 \tag{22}$$
$$\kappa = \mathsf{l}_1[t_1, \lambda]; \rho_1 \tag{23}$$
$$\sigma_1' = \mathsf{d}_1[t_1, q_1']; \rho_1 \tag{24}$$

Now, let us define $\xi : \mathsf{T}_1[t_1]_{\mathtt{univ}} \longleftrightarrow \mathsf{Q}_2[q_2]$ as follows:

$$\xi \stackrel{\text{def}}{=} \rho_1|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}}; \sigma_2^{-1} \tag{25}$$

We show that

$$\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset. \tag{26}$$

By (21) and (25), and since σ_2^{-1} is a partial correspondence, it is sufficient to show that

$$\mathrm{cod}(\rho_1|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}}) \cap \mathrm{cod}(\sigma_1) = \emptyset.$$

Now,

$$\begin{aligned}\mathrm{cod}(\sigma_1) &= \mathrm{cod}(\mathsf{s}_1[t_1,q_1];\rho_1) && \text{by (22)}\\ &= \mathrm{cod}(\rho_1|_{\mathrm{cod}(\mathsf{s}_1[t_1,q_1])})\\ &= \mathrm{cod}(\rho_1|_{\mathsf{T}_1[t_1]_{\mathtt{src}}}).\end{aligned}$$

Since ρ_1 is injective and $\mathsf{T}_1[t_1]_{\mathtt{src}} \cap \mathsf{T}_1[t_1]_{\mathtt{univ}} = \emptyset$, we conclude $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$.

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

$$\delta = \mathsf{s}_1[t_1,q_1];\zeta;\mathsf{s}_2[t_2,q_2]^{-1} \tag{27}$$

$$\xi = \zeta|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}};\mathsf{s}_2[t_2,q_2]^{-1} \tag{28}$$

$$\mathsf{l}_1[t_1,\lambda];\zeta = \mathsf{l}_2[t_2,\lambda] \tag{29}$$

$$\langle q_1',\delta',q_2'\rangle \in \widehat{\mathcal{R}} \text{ with } \delta' = \mathsf{d}_1[t_1,q_1'];\zeta;\mathsf{d}_2[t_2,q_2']^{-1}. \tag{30}$$

Let

$$\rho_2 \stackrel{\mathrm{def}}{=} (\mathsf{s}_2[t_2,q_2]^{-1};\sigma_2) \cup (\zeta^{-1};\rho_1). \tag{31}$$

Fact. $\rho_2 : \mathsf{T}_2[t_2] \longrightarrow \mathfrak{N}$.

Proof of the fact We first show that ρ_2 is defined on every $n \in \mathsf{T}_2[t_2]$, i.e., for each $n \in \mathsf{T}_2[t_2]$ there is some $(n,m) \in \rho_2$.
By the observation condition,

$$\mathsf{T}_2[t_2] = \mathrm{cod}(\mathsf{s}_2[t_2,q_2]) \cup \mathrm{cod}(\mathsf{l}_2[t_2,\lambda]). \tag{32}$$

If $n \in \mathrm{cod}(\mathsf{s}_2[t_2,q_2])$ then $\mathsf{s}_2[t_2,q_2]^{-1}(n)$ is defined and, since σ_2 is total, there exists some m such that

$$m = \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2(n)$$

and $(n,m) \in \rho_2$.
If $n \in \mathrm{cod}(\mathsf{l}_2[t_2,\lambda])$ then $\mathsf{l}_2[t_2,\lambda]^{-1}(n)$ is defined, and hence there exists some m such that

$$m = \mathsf{l}_2[t_2,\lambda]^{-1};\mathsf{l}_1[t_1,\lambda];\rho 1(n).$$

Now, by (29),

$$m = \zeta^{-1};\rho_1(n)$$

and $(n,m) \in \rho_2$.
Now we show that, whenever $(n,m) \in \rho_2$ and $(n',m') \in \rho_2$, $n = n'$ iff $m = m'$.
The cases $(n,m),(n',m') \in \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2$ and $(n,m),(n',m') \in \zeta^{-1};\rho_1$ are obvious, since $\mathsf{s}_2[t_2,q_2]^{-1};\sigma_2$ and $\zeta^{-1};\rho_1$ are partial correspondences. Assume, without loss of generality, that $(n,m) \in \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2$ and $(n',m') \in \zeta^{-1};\rho_1$.
Suppose $n = n'$; we distinguish three cases, depending on the sort of $\zeta^{-1}(n)$:

- $\zeta^{-1}(n) \in \mathsf{T}_1[t_1]_{\mathtt{new}}$: this case cannot happen. In fact, by Lemma A.1 it would imply $n \in \mathsf{T}_2[t_2]_{\mathtt{new}}$, which is absurd, since $n \in \mathrm{cod}(\mathsf{s}_2[t_2,q_2])$ and hence $n \in \mathsf{T}_2[t_2]_{\mathtt{src}}$.

- $\zeta^{-1}(n) \in \mathsf{T}_1[t_1]_{\mathtt{src}}$: then $\zeta^{-1};\mathsf{s}_1[t_1,q_1]^{-1}(n)$ is defined and, by (27):

$$\zeta^{-1};\mathsf{s}_1[t_1,q_1]^{-1};\delta(n) = \mathsf{s}_2[t_2,q_2]^{-1}(n) \tag{33}$$

and hence

$$\begin{aligned}
m &= \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2(n) && \text{since } (n,m) \in \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2\\
&= \zeta^{-1};\mathsf{s}_1[t_1,q_1]^{-1};\delta;\sigma_2(n) && \text{by (33)}\\
&= \zeta^{-1};\mathsf{s}_1[t_1,q_1]^{-1};\sigma_1(n) && \text{by (21)}\\
&= \zeta^{-1};\rho_1(n) && \text{by (22)}\\
&= m' && \text{since } (n,m') \in \zeta^{-1};\rho_1.
\end{aligned}$$

- $\zeta^{-1}(n) \in \mathsf{T}_1[t_1]_{\mathtt{univ}}$: then

$$\begin{aligned}
m &= \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2(n) && \text{since } (n,m) \in \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2\\
&= \zeta^{-1};\xi;\sigma_2(n) && \text{by (28)}\\
&= \zeta^{-1};\rho_1(n) && \text{by (25)}\\
&= m' && \text{since } (n,m') \in \zeta^{-1};\rho_1.
\end{aligned}$$

Now suppose $m = m'$; we distinguish three cases, depending on the sort of $\zeta^{-1}(n')$:

- $\zeta^{-1}(n') \in \mathsf{T}_1[t_1]_{\mathtt{new}}$: this case cannot happen by Lemma A.1.
- $\zeta^{-1}(n') \in \mathsf{T}_1[t_1]_{\mathtt{src}}$: then

$$\begin{aligned}
n &= \sigma_2^{-1};\mathsf{s}_2[t_2,q_2](m) && \text{since } (n,m) \in \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2\\
&= \sigma_2^{-1};\mathsf{s}_2[t_2,q_2](m') && \text{since } m = m'\\
&= \zeta^{-1};\rho_1;\sigma_2^{-1};\mathsf{s}_2[t_2,q_2](n') && \text{since } (n',m') \in \zeta^{-1};\rho_1\\
&= \zeta^{-1};\mathsf{s}_1[t_2,q_1]^{-1};\sigma_1;\sigma_2^{-1};\mathsf{s}_2[t_2,q_2](n') && \text{by (22) and } \zeta^{-1}(n') \in \mathsf{T}_1[t_1]_{\mathtt{src}}\\
&= \zeta^{-1};\mathsf{s}_1[t_2,q_1]^{-1};\delta;\mathsf{s}_2[t_2,q_2](n') && \text{by (21)}\\
&= n' && \text{by (27).}
\end{aligned}$$

- $\zeta^{-1}(n') \in \mathsf{T}_1[t_1]_{\mathtt{univ}}$: then

$$\begin{aligned}
n &= \sigma_2^{-1};\mathsf{s}_2[t_2,q_2](m) && \text{since } (n,m) \in \mathsf{s}_2[t_2,q_2]^{-1};\sigma_2\\
&= \sigma_2^{-1};\mathsf{s}_2[t_2,q_2](m') && \text{since } m = m'\\
&= \zeta^{-1};\rho_1;\sigma_2^{-1};\mathsf{s}_2[t_2,q_2](n') && \text{since } (n',m') \in \zeta^{-1};\rho_1\\
&= \zeta^{-1};\xi;\mathsf{s}_2[t_2,q_2](n') && \text{by (25)}\\
&= n' && \text{by (28) and } \zeta^{-1}(n') \in \mathsf{T}_1[t_1]_{\mathtt{univ}}.
\end{aligned}$$

(Fact)

Now we show that

$$\zeta = \rho_1;\rho_2^{-1} \tag{34}$$

i.e., that $\zeta^{-1} = \rho_2;\rho_1^{-1}$. Inclusion $\zeta^{-1} \subseteq \rho_2;\rho_1^{-1}$ comes from (31). For the other inclusion, assume $m = \rho_1;\rho_2^{-1}(n)$; we show that $m = \zeta(n)$.
By the observation condition, either $n \in \mathrm{cod}(\mathsf{s}_1[t_1,q_1])$ or $n \in \mathrm{cod}(\mathsf{l}_1[t_1,\lambda])$. In the first case,

$$\begin{aligned}
m &= \rho_1;\rho_2^{-1}(n)\\
&= \mathsf{s}_1[t_1,q_1]^{-1};\sigma_1;\sigma_2^{-1};\mathsf{s}_2[t_2,q_2](n) && \text{by (31) and (22)}\\
&= \mathsf{s}_1[t_1,q_1]^{-1};\delta\mathsf{s}_2[t_2,q_2](n) && \text{by (21)}\\
&= \zeta(n) && \text{by (27).}
\end{aligned}$$

In the second case $\zeta(n)$ is defined by (29), and $\zeta(n) = m$ by (31). This concludes the proof that $\zeta = \rho_1; \rho_2^{-1}$. Moreover

$$\sigma_2 = \mathsf{s}_2[t_2, q_2]; \rho_2. \tag{35}$$

In fact by (31)

$$\sigma_2 \subseteq \mathsf{s}_2[t_2, q_2]; \rho_2$$

and, since σ_2 is total, the equality must hold.
By (29), $\mathsf{l}_1[t_1, \lambda]; \rho_1; \rho_2^{-1} = \mathsf{l}_2[t_2, \lambda]$ and, since ρ_2 is total on $\mathsf{T}_2[t_2]$,

$$\mathsf{l}_1[t_1, \lambda]; \rho_1 = \kappa = \mathsf{l}_2[t_2, \lambda]; \rho_2. \tag{36}$$

By definition of global transition, exploiting (35) and (36), there is some $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(\lambda,\kappa)} (q_2', \sigma_2')$, where

$$\sigma_2' = \mathsf{d}_2[t_2, q_2']; \rho_2. \tag{37}$$

Finally,

$$\begin{aligned} \delta' &= \mathsf{d}_1[t_1, q_1']; \zeta; \mathsf{d}_2[t_2, q_2']^{-1} && \text{by (30)} \\ &= \mathsf{d}_1[t_1, q_1']; \rho_1; \rho_2^{-1}; \mathsf{d}_2[t_2, q_2']^{-1} && \text{by (34)} \\ &= \sigma_1'; \sigma_2'^{-1} && \text{by (30) and (37).} \end{aligned}$$

So, since $\langle q_1', \delta', q_2' \rangle \in \mathcal{R}$, by definition of $\mathcal{R}'$, $(q_1', \sigma_1')\ \mathcal{R}'\ (q_2', \sigma_2')$. This concludes the proof. $\square$

Lemma A.3 *If $\mathcal{R}$ is a global simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, then*

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

is a HD-simulation for $\mathcal{A}_1$ and $\mathcal{A}_2$.

Proof Suppose $\langle q_1, \delta, q_2 \rangle \in \mathcal{R}'$, $t_1 : q_1 \xrightarrow{\lambda} q_1'$ and $\xi : \mathsf{T}_1[t_1]_{\mathtt{univ}} \rightharpoonup \mathsf{Q}_2[q_2]$ such that

$$\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset. \tag{38}$$

We have to show that there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ and some $\zeta : \mathsf{T}_1[t_1] \rightharpoonup \mathsf{T}_2[t_2]$ such that:

- $\delta = \mathsf{s}_1[t_1, q_1]; \zeta; \mathsf{s}_2[t_2, q_2]^{-1}$;
- $\xi = \zeta|_{\mathsf{T}_2[t_2]_{\mathtt{univ}}}; \mathsf{s}_2[t_2, q_2]^{-1}$;
- $\mathsf{l}_1[t_1, \lambda]; \zeta = \mathsf{l}_2[t_2, \lambda]$;
- $\langle q_1', \delta', q_2' \rangle \in \mathcal{R}'$ where $\delta' \subseteq \mathsf{d}_1[t_1, q_1']; \zeta; \mathsf{d}_2[t_2, q_2']^{-1}$.

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

$$\delta = \sigma_1; \sigma_2^{-1}. \tag{39}$$

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

$$\begin{gathered} \sigma_1 = \mathsf{s}_1[t_1, q_1]; \rho_1 \\ \xi = \rho_1|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}}; \sigma_2^{-1} \\ \rho_1(\mathsf{T}_1[t_1]_{\mathtt{new}}) \cap \sigma_2(\mathsf{Q}_2[q_2]) = \emptyset \end{gathered}$$

There exists at least one of such functions.
In fact, let us define

$$
\begin{aligned}
\rho_1\big|_{\mathsf{T}_1[t_1]_{\mathtt{src}}} &= \mathsf{s}_1[t_1,q_1]^{-1};\sigma_1\\
\rho_1\big|_{\mathrm{dom}(\xi)} &= \xi;\sigma_2\\
\rho_1\big|_{\mathsf{T}_1[t_1]\smallsetminus(\mathsf{T}_1[t_1]_{\mathtt{src}}\cup\mathrm{dom}(\xi))} &: \mathsf{T}_1[t_1]\smallsetminus(\mathsf{T}_1[t_1]_{\mathtt{src}}\cup\mathrm{dom}(\xi))\longrightarrow\mathfrak{N}\smallsetminus(\mathrm{cod}(\sigma_2,\rho_1\big|_{\mathsf{T}_1[t_1]_{\mathtt{src}}\cup\mathrm{dom}(\xi)})).
\end{aligned}
$$

Notice that $\mathsf{T}_1[t_1]_{\mathtt{src}}\cap\mathrm{dom}(\xi)=\emptyset$; to have that ρ_1 is an injective function it is sufficient to show that $\mathrm{cod}(\mathsf{s}_1[t_1,q_1]^{-1};\sigma_1)\cap\mathrm{cod}(\delta;\sigma_2)=\emptyset$. We show that $\mathrm{cod}(\sigma_1)\cap\mathrm{cod}(\xi;\sigma_2)=\emptyset$. Suppose $n\in\mathrm{cod}(\sigma_1)\cap\mathrm{cod}(\xi;\sigma_2)$; then $\sigma_2^{-1}(n)\in\mathrm{cod}(\sigma_1;\sigma_2^{-1})\cap\mathrm{cod}(\xi;\sigma_2;\sigma_2^{-1})$ and hence $\sigma^{-1}(n)\in\mathrm{cod}(\xi)$. This implies $\mathrm{cod}(\delta)\cap\mathrm{cod}(\xi)\neq\emptyset$, which is absurd.
It is clear that ρ_1 satisfies (A.2), (A.2) and (A.2).
So, $(t_1,\rho_1):(q_1,\sigma_1)\xrightarrow{(\lambda,\kappa)}(q_1',\sigma_1')$ with

$$\sigma_1'=\mathsf{d}_1[t_1,q_1'];\rho_1. \tag{40}$$

By exploiting (A.2) there exists some $(t_2,\rho_2):(q_2,\sigma_2)\xrightarrow{(\lambda,\kappa)}(q_2',\sigma_2')$ such that $(q_1',\sigma_1')\ \mathcal{R}\ (q_2',\sigma_2')$,

$$\sigma_2=\mathsf{s}_2[t_2,q_2];\rho_2 \tag{41}$$
$$\sigma_2'=\mathsf{d}_2[t_2,q_2'];\rho_2 \tag{42}$$
$$\mathsf{l}_1[t_1,\lambda];\rho_1=\kappa=\mathsf{l}_2[t_2,\lambda];\rho_2. \tag{43}$$

Hence $t_2:q_2\xrightarrow{\lambda}q_2'$. Let

$$\zeta\stackrel{\mathrm{def}}{=}\rho_1;\rho_2^{-1}:\mathsf{T}_1[t_1]\longrightarrow\mathsf{T}_2[t_2]. \tag{44}$$

We have:

$$
\begin{aligned}
\delta &= \sigma_1;\sigma_2^{-1} && \text{by (39)}\\
&= \mathsf{s}_1[t_1,q_1];\rho_1;\rho_2^{-1};\mathsf{s}_2[t_2,q_2]^{-1} && \text{by (A.2) and (41)}\\
&= \mathsf{s}_1[t_1,q_1];\zeta;\mathsf{s}_2[t_2,q_2]^{-1} && \text{by (44).}
\end{aligned}
$$

Moreover,

$$
\begin{aligned}
\xi &= \rho_1\big|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}};\sigma_2^{-1} && \text{by (A.2)}\\
&= \rho_1\big|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}};\rho_2^{-1};\mathsf{s}_2[t_2,q_2]^{-1} && \text{by (41)}\\
&= \zeta\big|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}};\mathsf{s}_2[t_2,q_2]^{-1} && \text{by (44).}
\end{aligned}
$$

Also,

$$
\begin{aligned}
\mathsf{l}_1[t_1,\lambda];\zeta &= \mathsf{l}_1[t_1,\lambda];\rho_1;\rho_2^{-1} && \text{by (44)}\\
&= \mathsf{l}_2[t_2,\lambda];\rho_2;\rho_2^{-1} && \text{by (43)}\\
&= \mathsf{l}_2[t_2,\lambda]. && \text{since } \rho_2 \text{ is total on } \mathsf{T}_2[t_2].
\end{aligned}
$$

Finally, $(q_1',\sigma_1')\ \mathcal{R}\ (q_2',\sigma_2')$ implies $\langle q_1',\delta',q_2'\rangle\in\mathcal{R}'$ for $\delta'\stackrel{\mathrm{def}}{=}\sigma_1';\sigma_2'^{-1}$, and

$$
\begin{aligned}
\delta' &= \sigma_1';\sigma_2'^{-1}\\
&= \mathsf{d}_1[t_1,q_1'];\rho_1;\rho_2^{-1};\mathsf{d}_2[t_2,q_2']^{-1} && \text{by (40) and (42)}\\
&= \mathsf{d}_1[t_1,q_1'];\zeta;\mathsf{d}_2[t_2,q_2']^{-1} && \text{by (44).}
\end{aligned}
$$

This concludes the proof. □

Proposition 5.10 *Two HD-automata are HD-bisimilar if and only if they are global bisimilar*

Proof From Lemma A.2 we derive that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are HD-bisimilar via HD-bisimulation $\mathcal{R}$ then $\mathcal{A}_1$ and $\mathcal{A}_2$ are global-bisimilar via global-bisimulation

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

In fact, if $\mathcal{R}^{-1}$ is a HD-simulation for $\mathcal{A}_2$ and $\mathcal{A}_1$ then $\mathcal{R}'^{-1}$ is a global simulation for $\mathcal{A}_2$ and $\mathcal{A}_1$, and if $\langle q_{01}, \sigma_{01}; \sigma_{02}^{-1}, q_{02} \rangle \in \mathcal{R}$ then $(q_{01}, \sigma_{01})\ \mathcal{R}'\ (q_{02}, \sigma_{02})$.
Similarly, by Lemma A.3, if $\mathcal{A}_1$ and $\mathcal{A}_2$ are global-bisimilar via the global bisimulation $\mathcal{R}$ then $\mathcal{A}_1$ and $\mathcal{A}_2$ are HD-bisimilar via the HD-bisimulation

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

$\square$

A.3 Proof of Lemma 5.11

The following two results are standard for the π-calculus. They can be proved by induction on the derivation of the transitions (refer to [30] for the proofs).

Lemma A.4 *Let $p \xrightarrow{\bar{a}(x)} p'$ and $y \notin \mathtt{fn}(p)$. Then $p \xrightarrow{\bar{a}(y)} p'\{x/y\ y/x\}$.*

Proposition A.5 *Let p be a π-calculus process and let σ be an injective substitution. Whenever $p \xrightarrow{\alpha} p'$ then $p\sigma \xrightarrow{\beta} p'\sigma$, where:*

- *$\beta = \tau$ if $\alpha = \tau$;*
- *$\beta = \bar{c}d$ if $\alpha = \bar{a}b$ and $c = \sigma(a)$, $d = \sigma(b)$;*
- *$\beta = cd$ if $\alpha = ab$ and $c = \sigma(a)$, $d = \sigma(b)$;*
- *$\beta = \bar{c}(y)$ if $\alpha = \bar{a}(x)$ and $c = \sigma(a)$, $y = \sigma(x)$.*

Lemma 5.11 *Let (p, σ) be a global state of the HD-automaton $\mathcal{A}_p$ corresponding to a π-calculus agent p. Then:*

- *if $q\sigma \xrightarrow{\alpha} q''$ is a π-calculus transition and $(q', \sigma') = \mathtt{norm}(q'')$, then there is some global transition $(t, \rho) : (q, \sigma) \xrightarrow{(\lambda, \kappa)} (q', \sigma')$ of $\mathcal{A}_p$; and*
- *if $(t, \rho) : (q, \sigma) \xrightarrow{(\lambda, \kappa)} (q', \sigma')$ is a global transition of $\mathcal{A}_p$, then there is some π-calculus transition $q\sigma \xrightarrow{\alpha} q'\sigma'$*

where in both cases λ and κ are related to α as in Table 3.

Proof Here we just consider the case of the bound output transitions; the other forms of transitions can be dealed in similar ways.
Suppose $q\sigma \xrightarrow{\bar{b}(y)} q_0''$ and $(q', \sigma') = \mathtt{norm}(q_0'')$. By Proposition A.5, $q\sigma \xrightarrow{\bar{b}(y)} q_0''$ implies the existence of some transition $q \xrightarrow{\bar{a}(x)} q''$ with $q_0'' = q''\sigma$, $b = \sigma(a)$ and $y = \sigma(x)$.
By Lemma 4.5 there is some representative transition

$$t : q \xrightarrow{\bar{a}(z)} q''\{z/x\ x/z\}.$$

Notice that q_0'' and $q''\{z/x\ x/z\}$ differ only for an injective substitution: in fact

$$q_0'' = q''\sigma\{x/z\ z/x\} \tag{45}$$

and $\sigma\{x/z\ z/x\}$ is clearly injective. Therefore, according to the definition of norm, they have the same normalized representative, i.e., $\mathtt{norm}(q''\{z/x\ x/z\}) = (q', \sigma'')$ for some injective substitution σ''. Moreover, since $q'\sigma' = q''_0 = q''\sigma\{x/z\ z/x\} = q'\sigma''\sigma\{x/z\ z/x\}$, it must hold that

$$\sigma' = \sigma''\sigma\{x/z\ z/x\}, \tag{46}$$

at least on $\mathtt{fn}(q')$, and we can choose σ'' so that (46) holds on $\mathfrak{N}$.
By Definition 4.6 $t : q \xrightarrow{\mathtt{bout}} q'$ is a transition of $\mathcal{A}_p$, where $\mathsf{s}[t, q] = \mathrm{id}_{\mathtt{fn}(q)}$, $\mathsf{d}[t, q'] = \sigma''$ and $\mathsf{l}[t, \mathtt{bout}](n_{\mathrm{sub}}) = a$, $\mathsf{l}[t, \mathtt{bout}](n_{\mathrm{obj}}) = z$.
Let $\rho : \mathtt{fn}(q) \cup \{z\} \longleftrightarrow \mathfrak{N}$ be defined as follows: $\rho\big|_{\mathtt{fn}(q)} = \sigma$ and $\rho(z) = y$; moreover, let $\kappa : \{n_{\mathrm{sub}}, n_{\mathrm{obj}}\} \longleftrightarrow \mathfrak{N}$ be defined as follows: $\kappa(n_{\mathrm{sub}}) = b$ and $\kappa(n_{\mathrm{obj}}) = y$. By Definition 5.8

$$(t, \rho) : (q, \sigma) \xrightarrow{(\lambda,\kappa)} (q', \sigma')$$

is a global transition of $\mathcal{A}_p$. In fact:

- $\mathsf{s}[t, q]; \rho = \mathrm{id}_{\mathtt{fn}(q)}; \rho = \sigma$;
- $\mathsf{d}[t, q']; \rho = \sigma''; (\sigma\{x/z\ z/x\}) = \sigma''\sigma\{x/z\ z/x\} = \sigma'$; and
- $\mathsf{l}[t, \mathtt{bout}]; \rho(n_{\mathrm{sub}}) = \rho(a) = \sigma(a) = b = \kappa(n_{\mathrm{sub}})$ and $\mathsf{l}[t, \mathtt{bout}]; \rho(n_{\mathrm{obj}}) = \rho(z) = y = \kappa(n_{\mathrm{obj}})$.

This concludes the first direction of the proof.
Suppose now that $(t, \rho) : (q, \sigma) \xrightarrow{(\mathtt{bout},\kappa)} (q', \sigma')$ is a global transition of $\mathcal{A}_p$. Let $b = \kappa(n_{\mathrm{sub}})$ and $y = \kappa(n_{\mathrm{obj}})$. By Definition 5.8, there is some transition $t : q \xrightarrow{\mathtt{bout}} q'$ of $\mathcal{A}_p$ such that $\sigma = \mathsf{s}[t, q]; \rho$, $\sigma' = \mathsf{d}[t, q']; \rho$ and $\kappa = \mathsf{l}[t, \mathtt{bout}]; \rho$. Let $a = \mathsf{l}[t, \mathtt{bout}](n_s bj)$ and $x = \mathsf{l}[t, \mathtt{bout}](n_{\mathrm{obj}})$; then $b = \rho(a)$ and $y = \rho(x)$.
By Definition 4.6, there is some π-calculus transition $t : q \xrightarrow{\bar{a}(x)} q''$ such that $(q', \sigma'') = \mathtt{norm}(q'')$. Moreover $\mathsf{s}[t, q] = \mathrm{id}_{\mathtt{fn}(q)}$ and $\mathsf{d}[t, q'] = \sigma''$.
By Proposition A.5, there is some π-calculus transition $q\sigma \xrightarrow{\bar{b}(y)} q''\sigma\{x/y\ y/x\} = q''_0$. Since q'' and q''_0 differ just for an injective substitution, we have $\mathtt{norm}(q''_0) = (q', \sigma''')$. Moreover $q'\sigma''' = q''_0 = q''\sigma\{x/y\ y/x\} = q'\sigma''\sigma\{x/y\ y/x\}$, so it must be $\sigma''' = \sigma''\sigma\{x/y\ y/x\}$.
It remains to show that $q''_0 = q'\sigma'$. We have $q''_0 = q''\sigma\{x/y\ y/x\} = q'\sigma''\sigma\{x/y\ y/x\} = q'\sigma'$. □

A.4 Proofs for Section 5.4

Proposition 5.15 *Let $\mathcal{A}$ be a finitely named HD-automaton. Then $\mathcal{B}_\mathcal{A}$ is a HD-bisimulation.*

Proof We show that $\mathcal{B}_\mathcal{A}$ is a HD-simulation. The proof that $\mathcal{B}_\mathcal{A}^{-1}$ is a HD-simulation is similar.
Since $\mathcal{B}_\mathcal{A} \subseteq \mathcal{R}_\mathcal{A}$, we know that $\langle p, \delta, q\rangle \in \mathcal{B}_\mathcal{A}$ implies that for each $t : p \xrightarrow{\lambda} p'$ and for each $\xi : \mathsf{T}[t]_{\mathtt{univ}} \longleftrightarrow \mathsf{Q}[q]$ such that $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$, there exist some $t' : q \xrightarrow{\lambda} q'$ and some $\zeta : \mathsf{T}[t] \longleftrightarrow \mathsf{T}[t']$ such that:

- $\delta = \mathsf{s}[t, p]; \zeta; \mathsf{T}[t', q]^{-1}$;
- $\xi = \zeta\big|_{\mathsf{T}[t]_{\mathtt{univ}}}; \mathsf{s}[t', q]^{-1}$;
- $\mathsf{l}[t, \lambda]; \zeta = \mathsf{l}[t', \lambda]$;
- $\langle p', \delta', q'\rangle \in \mathcal{R}_\mathcal{A}$ where $\delta' \subseteq \mathsf{d}[t, p']; \zeta; \mathsf{d}[t', q']^{-1}$.

Let $\delta'' \stackrel{\text{def}}{=} \mathrm{id}_{\mathsf{Q}[p]_{\mathrm{active}}}; \delta'; \mathrm{id}_{\mathsf{Q}[q]_{\mathrm{active}}}$. By Lemma 5.14,

$$\langle p', \delta'', q'\rangle \in \mathcal{R}_\mathcal{A} \frown \mathcal{R}_\mathcal{A} \frown \mathcal{R}_\mathcal{A},$$

hence by Corollary 5.7 $\langle p', \delta'', q'\rangle \in \mathcal{R}_\mathcal{A}$.

By definition of δ'', it holds $\delta'' = \delta' \cap (\mathsf{Q}[p']_{\mathrm{active}} \times \mathsf{Q}[q']_{\mathrm{active}}) \subseteq \mathsf{Q}[p']_{\mathrm{active}} \times \mathsf{Q}[q']_{\mathrm{active}}$, so by definition of $\mathcal{B}_{\mathcal{A}}$

$$\langle p', \delta'', q' \rangle \in \mathcal{B}_{\mathcal{A}}.$$

Finally, $\delta'' \subseteq \delta' \subseteq \mathsf{d}[t, p']; \zeta; \mathsf{d}[t', q']^{-1}$, which concludes the proof. □

Proposition 5.20 *Let $\mathcal{A}$ be a HD-automaton. Then $\mathcal{A} \sim \Downarrow\!\mathcal{A}$.*

Proof We show that $\mathcal{B}_{\mathcal{A}}$ is a HD-bisimulation for $\mathcal{A}$ and $\Downarrow\!\mathcal{A}$. Notice that, if $\langle p_1, \delta, p_2 \rangle \in \mathcal{B}_{\mathcal{A}}$, then $p_1, p_2 \in Q = Q_{\Downarrow}$ and $\delta : \mathsf{Q}[p_1]_{\mathrm{active}} \longleftrightarrow \mathsf{Q}[p_2]_{\mathrm{active}}$ can also be seen as a partial correspondence $\delta : \mathsf{Q}[p_1] \leftharpoondown\!\!\rightharpoonup \mathsf{Q}_{\Downarrow}[p_2]$. Hence $\mathcal{B}_{\mathcal{A}}$ is well-sorted as a HD-bisimulation for $\mathcal{A}$ and $\Downarrow\!\mathcal{A}$.
Consider now a tuple $\langle p_1, \delta, p_2 \rangle \in \mathcal{B}_{\mathcal{A}}$. Suppose $t_1 : p_1 \xrightarrow{\lambda} p_1'$ and $\xi : \mathsf{T}[t_1]_{\mathtt{univ}} \leftharpoondown\!\!\rightharpoonup \mathsf{Q}_{\Downarrow}[q_2]$ such that $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$. We have to show that there is some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ in $\Downarrow\!\mathcal{A}$ and some $\zeta : \mathsf{T}[t_1] \leftharpoondown\!\!\rightharpoonup \mathsf{T}_{\Downarrow}[t_2]$ such that:

$$\delta = \mathsf{s}[t_1, q_1]; \zeta; \mathsf{s}_{\Downarrow}[t_2, q_2]^{-1} \tag{47}$$

$$\xi = \zeta|_{\mathsf{T}[t_1]_{\mathtt{univ}}}; \mathsf{s}_{\Downarrow}[t_2, q_2]^{-1} \tag{48}$$

$$\mathsf{l}[t_1, \lambda]; \zeta = \mathsf{l}_{\Downarrow}[t_2, \lambda] \tag{49}$$

and $\langle q_1', \delta', q_2' \rangle \in \mathcal{B}_{\mathcal{A}}$ for some

$$\delta' \subseteq \mathsf{d}[t_1, q_1']; \zeta; \mathsf{d}_{\Downarrow}[t_2, q_2']^{-1}. \tag{50}$$

Since $\mathcal{B}_{\mathcal{A}}$ is a HD-bisimulation for $\mathcal{A}$, there must be a transition t_2 is $\mathcal{A}$ which satisfies all the required properties. To deduce that t_2 is also in $\Downarrow\!\mathcal{A}$ we have to show that t_2 is active, i.e., that

$$\mathrm{cod}(\mathsf{l}[t_2, \lambda]; \mathsf{s}[q_2, t_2]^{-1}) \subseteq \mathsf{Q}[q_2]_{\mathrm{active}}.$$

By exploiting (48) this condition can be rewritten as

$$\mathrm{cod}(\mathsf{l}[t_1, \lambda]; \zeta; \mathsf{s}[q_2, t_2]^{-1}) \subseteq \mathsf{Q}[q_2]_{\mathrm{active}}. \tag{51}$$

Notice that, since $\mathrm{cod}(\zeta|_{\mathsf{T}[t_1]_{\mathtt{new}}}) \subseteq \mathsf{T}[t_2]_{\mathtt{new}}$, we have

$$\mathrm{cod}(\mathsf{l}[t_1, \lambda]; \zeta|_{\mathsf{T}[t_1]_{\mathtt{new}}}; \mathsf{s}[q_2, t_2]^{-1}) = \emptyset. \tag{52}$$

By (48) it holds that

$$\mathrm{cod}(\mathsf{l}[t_1, \lambda]; \zeta|_{\mathsf{T}[t_1]_{\mathtt{univ}}}; \mathsf{s}[q_2, t_2]^{-1}) = \mathrm{cod}(\mathsf{l}[t_1, \lambda]; \xi)$$

and ξ must be void, since δ is surjective on $\mathsf{Q}_{\Downarrow}[q_2] = \mathsf{Q}[q_2]_{\mathrm{active}}$, so

$$\mathrm{cod}(\mathsf{l}[t_1, \lambda]; \zeta|_{\mathsf{T}[t_1]_{\mathtt{univ}}}; \mathsf{s}[q_2, t_2]^{-1}) = \emptyset. \tag{53}$$

By exploiting (52) and (53), condition (51) can be rewritten as

$$\mathrm{cod}(\mathsf{l}[t_1, \lambda]; \zeta|_{\mathsf{T}[t_1]_{\mathtt{src}}}; \mathsf{s}[q_2, t_2]^{-1}) \subseteq \mathsf{Q}[q_2]_{\mathrm{active}}.$$

By (47) we have

$$\mathrm{cod}(\mathsf{l}[t_1, \lambda]; \zeta|_{\mathsf{T}[t_1]_{\mathtt{src}}}; \mathsf{s}[q_2, t_2]^{-1}) = \mathrm{cod}(\mathsf{l}[t_1, \lambda]; \mathsf{s}_1[q_1, t_1]^{-1}; \delta$$

which implies, by definition of active transition, that

$$\mathrm{cod}(\mathsf{l}[t_1, \lambda]; \zeta|_{\mathsf{T}[t_1]_{\mathtt{src}}}; \mathsf{s}[q_2, t_2]^{-1}) \subseteq \mathsf{s}[q_2, t_2]^{-1}(\mathsf{Q}[q_1]_{\mathrm{active}})$$

and finally, by Proposition 5.16,

$$\mathrm{cod}(\mathsf{l}[t_1,\lambda];\zeta|_{\mathsf{T}[t_1]_{\mathtt{src}}};\mathsf{s}[q_2,t_2]^{-1}) \subseteq \mathsf{Q}[q_2]_{\mathrm{active}}.$$

This concludes the proof of (51), i.e., that t_2 is active.

To conclude the proof we should prove that, given a transition $t_2 : q_2 \xrightarrow{\lambda} q_2'$ in $\Downarrow\mathcal{A}$ and a partial correspondence $\xi : \mathsf{T}_{\Downarrow}[t_2]_{\mathtt{univ}} \rightharpoonup \mathsf{Q}[q_2]$ such that $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$, there is some matching transition $t_1 : q_1 \xrightarrow{\lambda} q_1'$ in $\mathcal{A}$. This is easy, since t_2 is also a transition of $\mathcal{A}$ and hence the matching transition t_1 exists since $\mathcal{B}_{\mathcal{A}}$ is a HD-bisimulation for $\mathcal{A}$. $\square$

Proposition 5.21 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be irredundant HD-automata. Then $\mathcal{R} \subseteq \{\langle q_1,\delta,q_2\rangle \mid q_1 \in Q_1, q_1 \in Q_2, \delta : \mathsf{Q}_1[q_1] \rightharpoonup \mathsf{Q}_2[q_2]\}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ if and only if whenever $\langle q_1,\delta,q_2\rangle \in \mathcal{R}$ then:*

- *$\delta : \mathsf{Q}_1[q_1] \longleftrightarrow \mathsf{Q}_2[q_2]$ (i.e., δ is a total bijection between $\mathsf{Q}_1[q_1]$ and $\mathsf{Q}_2[q_2]$);*
- *for each $t_1 : q_1 \xrightarrow{\lambda} q_1'$ in $\mathcal{A}_1$ and for each there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ in $\mathcal{A}_2$ and some $\zeta : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_2[t_2]$ such that:*
  - *$\delta = \mathsf{s}_1[t_1,q_1];\zeta;\mathsf{s}_2[t_2,q_2]^{-1}$;*
  - *$\mathsf{l}_1[t_1,\lambda];\zeta = \mathsf{l}_2[t_2,\lambda]$;*
  - *$\langle q_1',\delta',q_2'\rangle \in \mathcal{R}$ where $\delta' = \mathsf{d}_1[t_1,q_1'];\zeta;\mathsf{d}_2[t_2,q_2']^{-1}$;*
- *the converse, starting from the transitions of q_2.*

Proof Let $\mathcal{R}$ be a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$ and let $\langle q_1,\delta,q_2\rangle \in \mathcal{R}$. We show that $\mathcal{R}$ satisfies all the conditions in the proposition.

First of all we show that $\delta : \mathsf{Q}_1[q_1] \longleftrightarrow \mathsf{Q}_2[q_2]$. This is true, since $\mathcal{R} \frown \mathcal{R}^{-1}$ and $\mathcal{R}^{-1} \frown \mathcal{R}$ are HD-bisimulations for $\mathcal{A}_1$ and for $\mathcal{A}_2$ respectively. Hence, by Proposition 5.16 it must be $\delta;\delta^{-1} : \mathsf{Q}_1[q_1] \longleftrightarrow \mathsf{Q}_1[q_1]$ and $\delta^{-1};\delta : \mathsf{Q}_2[q_2] \longleftrightarrow \mathsf{Q}_2[q_2]$, which is true if and only if δ and δ^{-1} are total.

Now, let $t_1 : q_1 \xrightarrow{\lambda} q_1'$ in $\mathcal{A}_1$; define $\xi : \mathsf{T}_1[t_1]_{\mathtt{univ}} \rightharpoonup \mathsf{Q}_2[q_2]$ as the empty relation (then obviously $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$); by definition of HD-bisimulation, there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ in $\mathcal{A}_2$ and some $\zeta : \mathsf{T}_1[t_1] \rightharpoonup \mathsf{T}_2[t_2]$ such that:

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

To conclude, if remains to show that $\delta' = \mathsf{d}_1[t_1,q_1'];\zeta;\mathsf{d}_2[t_2,q_2']^{-1}$, but this must hold, since $\delta' : \mathsf{Q}_1[q_1'] \longleftrightarrow \mathsf{Q}_2[q_2']$.

Now we prove the converse implication; let $\mathcal{R}$ be a relation that satisfies all the conditions in the proposition. Suppose $\langle q_1;\delta;q_2\rangle \in \mathcal{R}$ and $t_1 : q_1 \xrightarrow{\lambda} q_1'$ in $\mathcal{A}_1$. We have to prove that for each $\xi : \mathsf{T}_1[t_1]_{\mathtt{univ}} \rightharpoonup \mathsf{Q}_2[q_2]$ such that $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$, there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ in $\mathcal{A}_2$ and some $\zeta : \mathsf{T}_1[t_1] \rightharpoonup \mathsf{T}_2[t_2]$ such that:

- $\delta = \mathsf{s}_1[t_1,q_1];\zeta;\mathsf{s}_2[t_2,q_2]^{-1}$;
- $\xi = \zeta|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}};\mathsf{s}_2[t_2,q_2]^{-1}$;
- $\mathsf{l}_1[t_1,\lambda];\zeta = \mathsf{l}_2[t_2,\lambda]$;
- $\langle q_1',\delta',q_2'\rangle \in \mathcal{R}$ where $\delta' \subseteq \mathsf{d}_1[t_1,q_1'];\zeta;\mathsf{d}_2[t_2,q_2']^{-1}$.

Since $\mathcal{R}$ is a HD-bisimulation, there exists a transition t_2 that satisfies all the conditions except $\xi = \zeta|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}};\mathsf{s}_2[t_2,q_2]^{-1}$. This is true, since ξ and ζ are empty relations. In fact, ξ is empty since δ is total while $\zeta|_{\mathsf{T}_1[t_1]_{\mathtt{univ}}};\mathsf{s}_2[t_2,q_2]^{-1}$ is empty since $\zeta|_{\mathsf{T}_1[t_1]_{\mathtt{src}}} : \mathsf{T}_1[t_1]_{\mathtt{src}} \longleftrightarrow \mathsf{T}_2[t_2]_{\mathtt{src}}$ is a total bijection. $\square$

A.5 Proof of Theorem 5.23

Lemma A.6 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two irredundant HD-automata. If $\mathcal{R}$ is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$, then*

$$\mathcal{R}' \stackrel{\text{def}}{=} \{\langle (q_1, \sigma_1), (q_2, \sigma_2) \rangle \in Q_{\mathtt{unf}(\mathcal{A}_1)} \times Q_{\mathtt{unf}(\mathcal{A}_2)} \mid \langle q_1, \delta, q_2 \rangle \in \mathcal{R} \textit{ with } \delta = \sigma_1; \sigma_2^{-1}\}$$

is a bisimulation for $\mathtt{unf}(\mathcal{A}_1)$ and $\mathtt{unf}(\mathcal{A}_2)$.

Proof Suppose $(q_1, \sigma_1)\ \mathcal{R}'\ (q_2, \sigma_2)$ and let $(t_1, \rho_1) : (q_1, \sigma_1) \xrightarrow{(\lambda,\kappa,\gamma)} (q_1', \sigma_1')$ be a transition of $\mathtt{unf}(\mathcal{A})$. We have to show that there is some transition $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(\lambda,\kappa,\gamma)} (q_2', \sigma_2')$ of $\mathtt{unf}(\mathcal{A}_2)$ 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 $\langle q_1, \delta, q_2 \rangle \in \mathcal{R}$ with

$$\delta = \sigma_1; \sigma_2^{-1}. \tag{54}$$

Since (t_1, ρ_1) is a transition of $\mathtt{unf}(\mathcal{A}_1)$, by Definition 5.22 (t_1, ρ_1) is a global transition of $\mathcal{A}_1$ such that

$$\max\{n \in \mathsf{U}_1[(t_1, \rho_1)]_{\mathtt{univ}} \cup \mathsf{U}_1[(t_1, \rho_1)]_{\mathtt{new}}\} < \min\{n \notin \mathsf{U}_1[(t_1, \rho_1)]\} \tag{55}$$

and

$$\gamma = \mathsf{U}_1[(t_1, \rho_1)]_{\mathtt{univ}} \cup \mathsf{U}_1[(t_1, \rho_1)]_{\mathtt{new}}. \tag{56}$$

Since (t_1, ρ_1) is a global transition, we have $t_1 : q_1 \xrightarrow{\lambda} q_1'$ and:

$$\sigma_1 = \mathsf{s}_1[t_1, q_1]; \rho_1 \tag{57}$$

$$\kappa = \mathsf{l}_1[t_1, \lambda]; \rho_1 \tag{58}$$

$$\sigma_1' = \mathsf{d}_1[t_1, q_1']; \rho_1 \tag{59}$$

Since $\mathcal{R}$ is a HD-bisimulation on irredundant HD-automata, by Proposition 5.21 there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ and $\zeta : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_2[t_2]$ such that

$$\delta = \mathsf{s}_1[t_1, q_1]; \zeta; \mathsf{s}_2[t_2, q_2]^{-1} \tag{60}$$

$$\mathsf{l}_1[t_1, \lambda]; \zeta = \mathsf{l}_2[t_2, \lambda] \tag{61}$$

$$\langle q_1', \delta', q_2' \rangle \in \widehat{\mathcal{R}} \text{ with } \delta' = \mathsf{d}_1[t_1, q_1']; \zeta; \mathsf{d}_2[t_2, q_2']^{-1}. \tag{62}$$

Let $\rho_2 = \zeta^{-1}; \rho_1$. Then obviously $\zeta = \rho_1; \rho_2^{-1}$. Moreover,

$$\sigma_2 = \mathsf{s}_2[t_2, q_2]; \rho_2. \tag{63}$$

In fact, from (54) we have $\sigma_2 = \delta^{-1}; \sigma_1$. Hence, applying (60) and (57),

$$\sigma_2 = (\mathsf{s}_2[t_2, q_2]; \zeta^{-1}; \mathsf{s}_1[t_1, q_1]^{-1}); (\mathsf{s}_1[t_1, q_1]; \rho_1) = \mathsf{s}_2[t_2, q_2]; \zeta^{-1}; \rho_1.$$

Then (63) follows by exploiting $\zeta = \rho_1; \rho_2^{-1}$.
By (61), $\mathsf{l}_1[t_1, \lambda]; \rho_1; \rho_2^{-1} = \mathsf{l}_2[t_2, \lambda]$ and, since ρ_2 is total on $\mathsf{T}_2[t_2]$,

$$\mathsf{l}_1[t_1, \lambda]; \rho_1 = \kappa = \mathsf{l}_2[t_2, \lambda]; \rho_2. \tag{64}$$

By definition of global transition, exploiting (63) and (64), there is some global transition $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(\lambda,\kappa)} (q_2', \sigma_2')$, where

$$\sigma_2' = \mathsf{d}_2[t_2, q_2']; \rho_2. \tag{65}$$

To show that $(t_2, \rho_2) : (q_2, \sigma_2) \xrightarrow{(\lambda,\kappa,\gamma)} (q_2', \sigma_2')$ is also a transition of $\mathtt{unf}(\mathcal{A}_2)$, we have to show that

$$\max\{n \in \mathsf{U}_2[(t_2,\rho_2)]_{\mathtt{univ}} \cup \mathsf{U}_2[(t_2,\rho_2)]_{\mathtt{new}}\} < \min\{n \notin \mathsf{U}_2[(t_2,\rho_2)]\}$$

and

$$\gamma = \mathsf{U}_1[(t_2, \rho_1 2]_{\mathtt{univ}} \cup \mathsf{U}_2[(t_2,\rho_2)]_{\mathtt{new}}.$$

These hold from (55) and (56) since $\mathsf{U}_2[(t_2,\rho_2)] = \mathsf{U}_1[(t_1,\rho_1)]$ and similarly for the universal and new names. In fact, according to Definition 5.8, $\mathsf{U}_2[(t_2,\rho_2)] = \rho_2(\mathsf{T}_2[t_2])$ and $\mathsf{U}_1[(t_1,\rho_1)] = \rho_1(\mathsf{T}_1[t_1])$, and $\rho_2(\mathsf{T}_2[t_2]) = \rho_1(\mathsf{T}_1[t_1])$ since $\zeta = \rho_1;\rho_2^{-1}$ is a total bijection between $\mathsf{T}_1[t_1]$ and $\mathsf{T}_2[t_2]$.
Finally, by $\zeta = \rho_1;\rho_2^{-1}$ and (62),

$$\begin{aligned}
\sigma_1';\sigma_2'^{-1} &= \mathsf{d}_1[t_1,q_1'];\rho_1;\rho_2^{-1};\mathsf{d}_2[t_2,q_2']^{-1}\\
&= \mathsf{d}_1[t_1,q_1'];\zeta;\mathsf{d}_2[t_2,q_2']^{-1}\\
&= \delta'
\end{aligned}$$

hence $(q_1', \sigma_1')\ \mathcal{R}'\ (q_2', \sigma_2')$. □

Lemma A.7 *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two irredundant HD-automata. If $\mathcal{R}$ is a bisimulation for $\mathtt{unf}(\mathcal{A}_1)$ and $\mathtt{unf}(\mathcal{A}_2)$, then*

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

is a HD-bisimulation for $\mathcal{A}_1$ and $\mathcal{A}_2$.

Proof Suppose $\langle q_1, \delta, q_2\rangle \in \mathcal{R}'$ and $t_1 : q_1 \xrightarrow{\lambda} q_1'$.
By Proposition 5.21, we have to show that there exist some $t_2 : q_2 \xrightarrow{\lambda} q_2'$ such that

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

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

$$\delta = \sigma_1;\sigma_2^{-1}. \tag{66}$$

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

$$\sigma_1 = \mathsf{s}_1[t_1,q_1];\rho_1 \tag{67}$$

$$\max\{n \in \mathsf{U}_1[(t_1,\rho_1)]_{\mathtt{univ}} \cup \mathsf{U}_1[(t_1,\rho_1)]_{\mathtt{new}}\} < \min\{n \notin \mathsf{U}_1[(t_1,\rho_1)]\}. \tag{68}$$

There exists at least one of such functions. In fact, it is sufficient to define ρ_1 so that it maps $\mathsf{U}_1[(t_1,\rho_1)]_{\mathtt{univ}} \cup \mathsf{U}_1[(t_1,\rho_1)]_{\mathtt{new}}$ into the smallest names in $\mathfrak{N} \setminus \mathrm{cod}(\sigma_1)$.
Then $(t_1,\rho_1) : (q_1,\sigma_1) \xrightarrow{(\lambda,\kappa,\gamma)} (q_1', \sigma_2')$ is a transition in $\mathtt{unf}(\mathcal{A}_1)$ with

$$\gamma = \mathsf{U}_1[(t_1,\rho_1)]_{\mathtt{univ}} \cup \mathsf{U}_1[(t_1,\rho_1)]_{\mathtt{new}}.$$

Since $\mathcal{R}$ is a bisimulation for $\mathtt{unf}(\mathcal{A}_1)$ and $\mathtt{unf}(\mathcal{A}_2)$, there exists some transition $(t_2,\rho_2) : (q_2,\sigma_2) \xrightarrow{(\lambda,\kappa,\gamma)} (q_2', \sigma_2')$ in $\mathtt{unf}(\mathcal{A}_2)$ such that $(q_1',\sigma_1')\ \mathcal{R}\ (q_2',\sigma_2')$.
Now, $(t_2,\rho_2) : (q_2,\sigma_2) \xrightarrow{(\lambda,\kappa)} (q_2',\sigma_2')$ is a global transition of $\mathcal{A}_2$, so

$$\sigma_2 = \mathsf{s}_2[t_2,q_2];\rho_2 \tag{69}$$

$$\sigma_2' = \mathsf{d}_2[t_2,q_2'];\rho_2 \tag{70}$$

$$\mathsf{l}_1[t_1,\lambda];\rho_1 = \kappa = \mathsf{l}_2[t_2,\lambda];\rho_2. \tag{71}$$

Hence $t_2 : q_2 \xrightarrow{\lambda} q_2'$ is a transition of $\mathcal{A}_2$. Let

$$\zeta = \rho_1; \rho_2^{-1} : \mathsf{T}_1[t_1] \longleftrightarrow \mathsf{T}_2[t_2]. \tag{72}$$

By exploiting (66), (67) and (69) we obtain:

$$\begin{aligned}\delta &= \sigma_1; \sigma_2^{-1} \\ &= \mathsf{s}_1[t_1, q_1]; \rho_1; \rho_2^{-1}; \mathsf{s}_2[t_2, q_2]^{-1} \\ &= \mathsf{s}_1[t_1, q_1]; \zeta; \mathsf{s}_2[t_2, q_2]^{-1}.\end{aligned}$$

Also, since ρ_2 is total on $\mathsf{T}_2[t_2]$:

$$\begin{aligned}\mathsf{l}_1[t_1, \lambda]; \zeta &= \mathsf{l}_1[t_1, \lambda]; \rho_1; \rho_2^{-1} \\ &= \mathsf{l}_2[t_2, \lambda]; \rho_2; \rho_2^{-1} \\ &= \mathsf{l}_2[t_2, \lambda].\end{aligned}$$

Finally, $(q_1', \sigma_1')\ \mathcal{R}\ (q_2', \sigma_2')$ implies $\langle q_1', \delta', q_2' \rangle \in \mathcal{R}'$ for $\delta' = \sigma_1'; {\sigma_2'}^{-1}$. Since

$$\begin{aligned}\mathsf{d}_1[t_1, q_1']; \zeta; \mathsf{d}_2[t_2, q_2']^{-1} &= \mathsf{d}_1[t_1, q_1']; \rho_1; \rho_2^{-1}; \mathsf{d}_2[t_2, q_2']^{-1} \\ &= \sigma_1'; {\sigma_2'}^{-1} \\ &= \delta,\end{aligned}$$

we can conclude. □

Theorem 5.23 *Let $\mathcal{A}$ and $\mathcal{B}$ be two irredundant HD-automata. Then $\mathcal{A} \sim \mathcal{B}$ if and only if $\mathtt{unf}(\mathcal{A}) \sim \mathtt{unf}(\mathcal{B})$.*

Proof From Lemma A.6 it is easy to derive that if $\mathcal{A}_1$ and $\mathcal{A}_2$ are HD-bisimilar via the HD-bisimulation $\mathcal{R}$ then $\mathtt{unf}(\mathcal{A}_1)$ and $\mathtt{unf}(\mathcal{A}_2)$ are bisimilar via

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

In fact, the lemma assures that if $\mathcal{R}$ is a HD-bisimulation $\mathcal{R}'$ is a bisimulation for the unfoldings and, moreover, if $\langle q_{01}, \sigma_{01}; \sigma_{02}^{-1}, q_{02} \rangle \in \mathcal{R}$ then $(q_{01}, \sigma_{01})\ \mathcal{R}'\ (q_{02}, \sigma_{02})$.
Similarly, by Lemma A.7, if $\mathtt{unf}(\mathcal{A}_1)$ and $\mathtt{unf}(\mathcal{A}_2)$ are bisimilar via the bisimulation $\mathcal{R}$ then $\mathcal{A}_1$ and $\mathcal{A}_2$ are HD-bisimilar via the HD-bisimulation

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

□

A.6 Proofs for Section 6.4

Here we want to show that incremental location equivalence coincides with the standard location equivalence. Some preliminary results are required.

Lemma A.8 *Let $p \sim_l q$; then $m :: p \sim_l m :: q$ for each location m.*

Lemma A.9 *Let p and q be two CCS agents with localities and let p' and q' be the two corresponding pure CCS agents. Then $p \sim_l q$ implies $p' \sim_l q'$.*

Definition A.10 Let p be a CCS agent with localities. We define loctree(p) as follows:

$$\begin{array}{rclrcl} \text{loctree}(\mathbf{0}) &=& \emptyset & \text{loctree}(\mu.p) &=& \{\epsilon\} \\ \text{loctree}(l :: p) &=& \{lu \mid u \in \text{loctree}(p)\} & \text{loctree}((\nu\alpha)\, p) &=& \text{loctree}(p) \\ \text{loctree}(p + q) &=& \text{loctree}(p) \cup \text{loctree}(q) & \text{loctree}(p|q) &=& \text{loctree}(p) \cup \text{loctree}(q) \end{array}$$

where ϵ is the empty sequence, i.e., $u\epsilon = u$ for every u.

Lemma A.11 *If* $p \xrightarrow[um]{\alpha} p'$ *then* $u \in \mathrm{loctree}(p)$.

Proof By induction on the derivation of the transition. $\square$

Lemma A.12 *Let* $p_0 = l_0 :: p$ *for some pure process* p *and assume*

$$p_0 \xrightarrow{\mu_1} p_1 \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_k} p_k$$

where $\xrightarrow{\mu_i}$ *is either* $\xrightarrow{\tau}$ *or* $\xrightarrow[u_i m_i]{a_i}$ *with* $m_i \notin \mathtt{loc}(p_{i-1})$. *If* $um, vm \in \mathrm{loctree}(p_k)$ *then* $u = v$.

Proof This can be proved by induction on k, by exploiting the fact that $m_i \notin \mathtt{loc}(p_i)$. $\square$

Lemma A.13 *Let* $p_0 = l_0 :: p$ *and* $q_0 = l_0 :: q$ *for two pure processes* p *and* q *and assume*

$$l_0 :: p_0 \xrightarrow{\mu_1} p_1 \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_k} p_k$$
$$l_0 :: q_0 \xrightarrow{\mu_1} q_1 \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_k} q_k$$

where $\xrightarrow{\mu_i}$ *is either* $\xrightarrow{\tau}$ *or* $\xrightarrow[u_i m_i]{a_i}$ *with* $m_i \notin \mathtt{loc}(p_{i-1}, q_{i-1})$. *If* $p_k \xrightarrow[umn]{a} p_{k+1}$ *and* $q_k \xrightarrow[vmn]{a} q_{k+1}$ *then* $u = v$.

Proof By Lemma A.11, $p_k \xrightarrow[umn]{a} p_{k+1}$ implies $um \in \mathrm{loctree}(p_k)$ and, similarly, $q_k \xrightarrow[vmn]{a} q_{k+1}$ implies $vm \in \mathrm{loctree}(q_k)$.
The conclusion follows from Lemma A.12. $\square$

Proposition 6.4 *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 The proof that $p_0 \sim_l q_0$ implies $m :: p_0 \sim_{\Delta l} m :: q_0$ is easy. In fact, each location bisimulation is by definition also an incremental location bisimulation, so it is sufficient to prove that $m :: p_0 \sim_l m :: q_0$, which is true by Lemma A.8.
Now we prove the converse, i.e., that $m :: p_0 \sim_{\Delta l} m :: q_0$ implies $p_0 \sim_l q_0$. Let relation $\mathcal{R}$ be defined as follows:

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

where $\xrightarrow{\mu_i}$ is either $\xrightarrow{\tau}$ or $\xrightarrow[u_i m_i]{a_i}$ with $m_i \notin \mathtt{loc}(p_{i-1}, q_{i-1})$. Then, by Lemma A.13, $\mathcal{R}$ is a location bisimulation. Hence, $m :: p_0 \sim_l m :: q_0$. By applying Lemma A.9 we obtain $p_0 \sim_l q_0$. $\square$

A.7 Proof of Proposition 6.24

Proposition 6.24 *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 Let $\mathcal{R}$ be a history-preserving bisimulation for N_1 and N_2; we show that

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

is an incremental history-preserving bisimulation. This is sufficient to prove that $N_1 \sim_{hp} N_2$ implies $N_1 \sim_{\Delta hp} N_2$.
Let $(c_1, g, c_2) \in \mathcal{R}'$, and suppose $c_1 \xrightarrow{t_1} c_1'$. By definition of $\mathcal{R}'$ we have that $c_1 = c_{\pi_1}$, $c_2 = c_{\pi_2}$ and

$$g = f \cap (E_{c_1} \times E_{c_2}) \tag{73}$$

for $(\pi_1, f, \pi_2) \in \mathcal{R}$. By Proposition 6.22, moreover, $\pi_1 \xrightarrow{t_1} \pi_1'$ and $c_1' = c_{\pi_1'}$.
Since $(\pi_1, f, \pi_2) \in \mathcal{R}$ and $\mathcal{R}$ is a history-preserving bisimulation, there must be some transition $\pi_2 \xrightarrow{t_2} \pi_2'$ such that $(\pi_1', f', \pi_2') \in \mathcal{R}$ and $f'\big|_{\mathtt{ev}(\pi_1)} = f$. Moreover, since f is an isomorphism between $\mathtt{ev}(\pi_1)$ and $\mathtt{ev}(\pi_2)$, it must be $f(\bar{e}_1) = \bar{e}_2$, and $l_1(t_1) = l_2(t_2)$.

From $\pi_2 \xrightarrow{t_2} \pi'_2$ we deduce $c_2 \xrightarrow{t_2} c'_2$ for $c'_2 = c_{\pi'_2}$; from $(\pi'_1, f', \pi'_2) \in \mathcal{R}$ we deduce $(c'_1, g', c'_2) \in \mathcal{R}'$ for

$$g' = f' \cap (E_{c'_1} \times E_{c'_2}). \tag{74}$$

To conclude, we should prove that $g' = (g \cup (\bar{e}_1, \bar{e}_2)) \cap (E_{c'_1} \times E_{c'_2})$, which is true by (73) and (74) since $f' = f \cup (\bar{e}_1, \bar{e}_2)$.
Now the converse. Let $\mathcal{R}$ be an incremental history-preserving bisimulation. Then

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

is a history-preserving bisimulation. This is sufficient to prove that $N_1 \sim_{\Delta hp} N_2$ implies $N_1 \sim_{hp} N_2$.
The proof is similar to the one of the other implication and hence omitted. □

A.8 Proof of Theorem 7.5

Lemma A.14 *If state q in the HD-automaton $\mathcal{A}$ is reachable from the initial state then there is an experiment $\mathcal{X}$ with states $\{q_0, \dots, q_n\}$ and a morphism $x : \mathcal{X} \to \mathcal{A}$ such that $x_Q(q_n) = q$.*

Lemma A.15 *Let $m : \mathcal{A} \to \mathcal{B}$ be an* **Exp**-*open map. Then A and B are HD-bisimilar.*

Proof We show that

$$\mathcal{R} = \{\langle q_a, \delta, q_b\rangle \mid q_a \text{ reachable}, q_b = m_Q(q_a), \delta = \mathsf{m_Q}[q_a, q_b]^{-1}\}$$

is a HD-bisimulation for $\mathcal{A}$ and $\mathcal{B}$.
Suppose that $\langle q_a, \delta, q_b\rangle \in \mathcal{R}$ and that $t_a : q_a \xrightarrow{\lambda} q'_a$. Since all the names of q_b also appear in q_a, the only possible choice for ξ is the void partial bijection. For the definition of morphisms between HD-automata, transition t_a can be matched with transition $m_T(t_a) = t_b : q_b \xrightarrow{\lambda} q'_b$, choosing $\zeta = \mathsf{m_T}[t_a, t_b]^{-1}$. If fact:

- since $\mathsf{s_a}; \mathsf{m_Q} = \mathsf{m_T}; \mathsf{s_b}$, we also have $\mathsf{s_a}[t_a, q_a]; \mathsf{m_Q}[q_a, q_b] = \mathsf{m_T}[t_a, t_b]; \mathsf{s_b}[t_b, q_b]$ and hence $\delta = \mathsf{m_Q}[q_a, q_b]^{-1} = \mathsf{s_a}[t_a, q_a]; \mathsf{m_T}[t_a, t_b]^{-1}; \mathsf{s_b}[t_b, q_b]^{-1} = \mathsf{s_a}[t_a, q_a]; \zeta; \mathsf{s_b}[t_b, q_b]^{-1}$.
- $\zeta|_{\mathsf{T_a}[t_a]_{\mathtt{univ}}}; \mathsf{s_a}[t_b, q_b]^{-1} = \mathsf{m_T}[t_a, t_b]^{-1}|_{\mathsf{T_a}[t_a]_{\mathtt{univ}}}; \mathsf{s_b}[t_b, q_b]^{-1} = \emptyset = \xi$ since all the old names of t_b are mapped into old names of t_a;
- since $\mathsf{l_a} = \mathsf{m_T}; \mathsf{l_b}$ we also have $\mathsf{l_a}[t_a, \lambda] = \mathsf{l_b}[t_b, \lambda]; \mathsf{m_T}[t_a, t_b] = \mathsf{l_b}[t_b, \lambda]; \zeta^{-1}$ and hence $\mathsf{l_a}[t_a, \lambda]; \zeta = \mathsf{l_b}[t_b, \lambda]$;
- $\langle q'_a, \delta', q'_b\rangle \in \mathcal{R}$ where $\delta' = \mathsf{m_Q}[q'_a, q'_b]^{-1}$; exploiting $\mathsf{d_a}; \mathsf{m_Q} = \mathsf{m_T}; \mathsf{d_b}$ and hence $\mathsf{m_Q}[q'_a, q'_b]; \mathsf{d_a}[t_a, q'_a] = \mathsf{d_b}[t_b, q'_b]; \mathsf{m_T}[t_a, t_b]$ we also obtain $\delta' = \mathsf{d_a}[t_a, q'_a]; \mathsf{m_T}[t_a, t_b]^{-1}; \mathsf{d_b}[t_b, q'_b]^{-1} = \mathsf{d_a}[t_a, q'_a]; \zeta; \mathsf{d_b}[t_b, q'_b]^{-1}$.

Suppose, conversely, that $t_b : q_b \xrightarrow{\lambda} q'_b$ and $\xi : \mathsf{T_b}[t_b]_{\mathtt{univ}} \longleftrightarrow \mathsf{Q_a}[q_a]$ such that $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$. Since q_a is reachable, by Lemma A.14, there is an experiment $\mathcal{X}$ and a morphism $x : \mathcal{X} \to \mathcal{A}$ so that experiment $\mathcal{X}$ terminates on q_a. Via $x; m$, experiment $\mathcal{X}$ can be executed in $\mathcal{B}$, ending in state q_b.
Now we extend experiment $\mathcal{X}$ to an experiment $\mathcal{Y}$, obtained by adding a transition corresponding to t_b. 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 $t_{n+1} : q_n \xrightarrow{\lambda} q_{n+1}$. The definition of the name-preserving map $f : \mathcal{X} \to \mathcal{Y}$ is hence obvious, since $\mathcal{X}$ is a prefix of $\mathcal{Y}$. Let us define $\mathsf{T_y}[t_{n+1}] = \mathsf{Q_y}[q_{n+1}] = \mathsf{Q_x}[q_n] \uplus (\mathsf{T_b}[t_b] \setminus \mathsf{T_b}[t_b]_{\mathtt{src}} \setminus \mathrm{dom}(\xi))$. Notice that we can define a map $\alpha : \mathsf{T_b}[t_b] \longleftrightarrow \mathsf{T_y}[t_{n+1}]$ as follows: let $m \in \mathsf{T_B}[t_b]$;

- if $m \in \mathsf{T_B}[t_b]_{\mathtt{src}}$ then $\alpha(m) = \mathsf{s_b}[t_b, q_b]^{-1}; (\mathsf{x}; \mathsf{m})_{\mathsf{Q}}[q_n, q_b](m) \in \mathsf{Q_x}[q_n] \subseteq \mathsf{T_y}[t_{n+1}]$;
- if $m \in \mathrm{dom}(\xi)$ then $\alpha(m) = \xi; \mathsf{x_Q}[q_n, q_a](m) \in \mathsf{Q_x}[q_n] \subseteq \mathsf{T_y}[t_{n+1}]$;
- otherwise $\alpha(m) = m \in \mathsf{T_b}[t_b] \setminus \mathsf{T_b}[t_b]_{\mathtt{src}} \setminus \mathrm{dom}(\xi) \subseteq \mathsf{T_y}[t_{n+1}]$.

Maps $\mathsf{s_y}[t_{n+1}, q_n]$ and $\mathsf{d_y}[t_{n+1}, q_{n+1}]$ are obvious, whereas map $\mathsf{l_y}[t_{n+1}, \lambda] = \mathsf{l_b}[t_b, \lambda]; \alpha$.
Let us define morphism $y : \mathcal{Y} \to \mathcal{B}$ as follows: on $\mathcal{X}$, morphism y is defined by $x; m$ (this makes sure that the diagram

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

commutes); moreover transition t_{n+1} is mapped on t_b so that $\mathsf{y_T}[t_{n+1}, t_b] = \alpha$ and state q_{n+1} is mapped on Q'_b so that $\mathsf{y_Q}[q_{n+1}, q_b] = \mathsf{s_b}[t_b, q_b]; \alpha$.
Since m is an open map, a morphism $y' : \mathcal{Y} \to \mathcal{A}$ can be found so that diagram

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

commutes. Let us define $q'_a = y'_Q(q_{n+1})$ and $t_a = y'_T(t_{n+1})$; then $t_a : q_a \xrightarrow{\lambda} q'_a$. Let us define $\zeta = \mathsf{m_T}[t_a, t_b]$. Then:

- since $\mathsf{s_a}; \mathsf{m_Q} = \mathsf{m_T}; \mathsf{s_b}$, we also have $\mathsf{s_a}[t_a, q_a]; \mathsf{m_Q}[q_a, q_b] = \mathsf{m_T}[t_a, t_b]; \mathsf{s_b}[t_b, q_b]$ and hence $\delta = \mathsf{m_Q}[q_a, q_b] = \mathsf{s_b}[t_b, q_b]; \mathsf{m_T}[t_a, t_b]; \mathsf{s_a}[t_a, q_a]^{-1} = \mathsf{s_b}[t_b, q_b]; \zeta; \mathsf{s_a}[t_a, q_a]^{-1}$.
- $\xi = \alpha\big|_{\mathsf{T_y}[t_{n+1}]_{\mathtt{univ}}}; \mathsf{s_y}[t_{n+1}, q_n]^{-1}; \mathsf{y'_Q}[q_n, q_a]^{-1} = \mathsf{y_T}[t_{n+1}, t_b]\big|_{\mathsf{T_y}[t_{n+1}]_{\mathtt{univ}}}; \mathsf{y'_T}[t_{n+1}, t_a]^{-1}; \mathsf{s_a}[t_a, q_a]^{-1}$ and, since diagram commutes, $\xi = \mathsf{m_T}[t_a, t_b]\big|_{\mathsf{T_a}[t_a]_{\mathtt{univ}}}; \mathsf{s_a}[t_a, q_a]^{-1} = \zeta\big|_{\mathsf{T_a}[t_a]_{\mathtt{univ}}}; \mathsf{s_a}[t_a, q_a]^{-1}$.
- since $\mathsf{l_a} = \mathsf{m_T}; \mathsf{l_b}$ we also have $\mathsf{l_a}[t_a, \lambda] = \mathsf{l_b}[t_b, \lambda]; \mathsf{m_T}[t_a, t_b] = \mathsf{l_b}[t_b, \lambda]; \zeta$;
- $\langle q'_a, \delta', q'_b \rangle \in \mathcal{R}$ where $\delta' = \mathsf{m_Q}[q'_a, q'_b]^{-1}$; exploiting $\mathsf{d_a}; \mathsf{m_Q} = \mathsf{m_T}; \mathsf{d_b}$ and hence $\mathsf{m_Q}[q'_a, q'_b]; \mathsf{d_a}[t_a, q'_a] = \mathsf{d_b}[t_b, q'_b]; \mathsf{m_T}[t_a, t_b]$ we also obtain $\delta' = \mathsf{d_a}[t_a, q'_a]; \mathsf{m_T}[t_a, t_b]^{-1}; \mathsf{d_b}[t_b, q'_b]^{-1} = \mathsf{d_a}[t_a, q'_a]; \zeta^{-1}; \mathsf{d_b}[t_b, q'_b]^{-1}$.

$\square$

From this, by transitivity of $\sim$, it is easy to conclude that two HD-automata are HD-bisimilar if there is a span of open maps for them.

Lemma A.16 *Let $\mathcal{A}$ and $\mathcal{B}$ be HD-bisimilar automata. There exists a HD-automaton $\mathcal{A} \otimes \mathcal{B}$ and a span of open maps*

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

Proof Suppose that $\mathcal{A}$ and $\mathcal{B}$ are HD-bisimilar via the HD-bisimulation $\mathcal{R}$ on $\mathcal{A}$ and $\mathcal{B}$; by Proposition 5.6 we can suppose that $\mathcal{R} = \widehat{\mathcal{R}}$; hence $\langle i_a(*), \mathsf{i_a}[*, i_a(*)]; \mathsf{i_b}[*, i_B(*)]^{-1}, i_b(*) \rangle \in \mathcal{R}$). The HD-automaton $\mathcal{A} \otimes \mathcal{B}$ and m and m' are defined as follows:

- $Q_{ab} = \mathcal{R}$; 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 $\mathsf{Q_{ab}}[r]$, $\mathsf{m_Q}[r, q_a]$ and $\mathsf{m'_Q}[r, q_b]$ are defined by requiring that diagram

$$\begin{tikzcd}
\delta \arrow[r, "\pi_1"] \arrow[d, "\pi_2"] & \mathsf{Q_a}[q_a] \arrow[d, "{\mathsf{m_Q}[r,q_a]}"] \\
\mathsf{Q_b}[q_b] \arrow[r, "{\mathsf{m'_Q}[r,q_b]}"] & \mathsf{Q_{ab}}[r]
\end{tikzcd}$$

  is a pushout (hence $\mathsf{Q_{ab}}[r]$ is the union of the names of q_a and q_b, where names related by δ are identified);

- for each state $\langle q_a, \delta, q_b \rangle$, let $t_a : q_a \xrightarrow{\lambda} q'_a$ be a transition of $\mathcal{A}$ and $\xi : \mathsf{T_a}[t_a]_{\mathtt{univ}} \rightharpoonup \mathsf{Q_b}[q_b]$ be a partial bijection such that $\mathrm{cod}(\xi) \cap \mathrm{cod}(\delta) = \emptyset$ and suppose that $t_b : q_b \xrightarrow{\lambda} q'_b$, $\zeta : \mathsf{T_a}[t_a] \rightharpoonup \mathsf{T_b}[t_b]$ and δ' are chosen as in Definition 5.1 and such that $\delta' = \mathsf{d_a}[t_a, q'_a]; \zeta; \mathsf{d_b}[t_b, q'_b]^{-1}$ (this is possible since we have required $\mathcal{R} = \widehat{\mathcal{R}}$). Then $t = \langle t_a, \zeta, t_b \rangle : \langle q_a, \delta, q_b \rangle \xrightarrow{\lambda} \langle q'_a, \delta', q'_b \rangle$ is a transition of $\mathcal{A} \otimes \mathcal{B}$; we have $m_T(t) = t_a$ and $m'_T(t) = t_b$; moreover $\mathsf{T_{ab}}[t]$, $\mathsf{m_T}[t, t_a]$ and $\mathsf{m'_T}[t, t_b]$ are defined by requiring that diagram

$$\begin{tikzcd} \zeta \arrow[r, "\pi_1"] \arrow[d, "\pi_2"] & \mathsf{T_a}[t_a] \arrow[d, "\mathsf{m_T}[t,t_a]"] \\ \mathsf{T_b}[t_b] \arrow[r, "\mathsf{m'_T}[t,t_b]"] & \mathsf{T_{ab}}[t] \end{tikzcd}$$

is a pushout;

- similarly, starting from the transitions of q_b, transitions of the form $\langle t_b, \zeta, t_a \rangle$ with $\zeta : \mathsf{T_b}[t_b] \rightharpoonup \mathsf{T_a}[t_a]$ are added to $\mathcal{A} \otimes \mathcal{B}$;

- the starting state of $\mathcal{A} \otimes \mathcal{B}$ is $\langle i_a(*), \mathsf{i_a}[*, i_a(*)]; \mathsf{i_b}[*, i_b(*)]^{-1}, i_b(*) \rangle \in \mathcal{R}$.

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

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

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

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

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 $t_{n+1} : q_n \xrightarrow{\lambda} q_{n+1}$. 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 $\mathcal{A} \otimes \mathcal{B}$. Suppose that $y_T(t_{n+1}) = t_a : q_a \xrightarrow{\lambda} q'_a$ (with $q_a = y_Q(q_n)$ and $q'_a = y_Q(q_{n+1})$) and let $q = x_Q(q_n)$. Let $\xi = \mathsf{x_T}[t_{n+1}, t_a]|_{\mathsf{T_a}[t_a]_{\mathtt{univ}}}; \mathsf{s_y}[t_{n+1}, q_n]^{-1}; \mathsf{x_Q}[q_n, q]^{-1}$; it is easy to see that $\xi : \mathsf{T_a}[t_a]_{\mathtt{univ}} \rightharpoonup \mathsf{Q_{ab}}[q]$ and that $\mathrm{cod}(\delta) \cap \mathrm{cod}(\xi) = \emptyset$.

Hence, we can use transition $t = \langle t_b, \zeta, t_a \rangle : q \xrightarrow{\lambda} q'$ chosen in the third point of the construction of $\mathcal{A} \otimes \mathcal{B}$; 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 also the lower triangle commutes. □

Theorem 7.5 *Two HD-automata on the same labels L and sortinging* Γ *are HD-bisimilar iff they are open-bisimilar.*

Proof Lemma A.15 is the *if* implication, whereas Lemma A.16 is the *only if* implication. □