A novel class of automata for languages on infinite alphabets

Vincenzo Ciancia[1] and Emilio Tuosto[2]

[1] Dept. de Sistemas Informaticos y Computacion, Universidad Complutense de Madrid, ES
[2] Dept. of Computer Science, University of Leicester, UK

Abstract. Automata theory is fundamental in Computer Science; in formal language theory automata are used to characterise classes of languages of words over finite alphabets.
In this paper we set the basis for a formal language theory inspired by *nominal calculi* (e.g. the π-calculus) and based on an automata framework amenable to deal with languages over *infinite alphabets*.
Our aim is to develop a novel combination of formal languages over infinite alphabets and nominal calculi. More precisely, we focus on the basic definitions of recognisability of languages inspired by nominal calculi where new symbols may be "generated" to match symbols from an infinite alphabet. Our notion of recognisability is based on *history-dependent* (HDA). Noticeably, the most distinguished feature of HDA, namely *locality of names*, is pivotal in our approach.

1 Introduction

Dynamically reconfigurable (or mobile) systems are widely envisaged as an important class of systems whose investigation is of paramount importance nowadays. Mobility brings in pivotal engineering tools and makes it possible to specify dynamically composable and/or reconfigurable systems. Noteworthy, mobility introduces difficulties when one wants to reason about mobile systems as they typically exhibit infinite states computations.

A rather elegant way of modelling mobile systems is by allowing "fresh elements" to be introduced in their computations. The π-calculus [21, 14, 15] is a milestone in the study of mobile systems and has inspired the family of the so-called *nominal calculi*. The notion of *name* is central to nominal calculi and provides a unifying concept to represent communication channels, localities, sessions, cryptographic keys, etc. Nominal calculi model mobility through sophisticated forms of name passing and *name extrusion*. For instance, the π-calculus transition

$$x!y.P \mid x(z).Q \rightarrow P \mid Q[y/x] \tag{1}$$

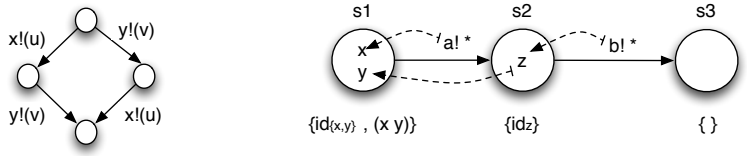
represents the computation of a system where, on the left-hand side there is a process $x!y.P$ sending y over the channel x in parallel with $x(z).Q$ waiting to input something on the channel x. When the synchronisation over x takes place, the system evolves as in the right-hand side of (1) where the continuation P of the sender runs in parallel with the continuation of the receiver where y is replaced for z (in symbols $P \mid Q[y/x]$).

A distinguished feature of nominal calculi is *scope extrusion*, that is the communication of freshly generated *names* so that processes can dynamically extend their reaction capabilities (a newly acquired name can be used by a process in its future interactions). For example, consider a slight variant of (1)

$$(\nu\, y)(x!y.P) \mid x(z).Q \rightarrow (\nu\, y)(P \mid Q[y/z])$$

where y does not occur in Q and is *local* to the sender (the restriction operator ν on the left-hand side binds y in P). Notice that after the synchronisation the receiver gets the fresh name y and can use it for further interactions with P; in fact, the scope of the restriction on y changes after the synchronisation as it includes also the continuation of the receiver.

History-Dependent automata (HDA) have been introduced as a syntax-free model of nominal calculi [16, 20, 18] and exploited to check behavioural equivalences [5, 6] (see [10, 8] for a comparison with other models). The basic idea is that states and transitions of HDA are equipped with (local) names the semantics of which is given by (injective) mappings which relate names of states along transitions of automata. The key features of HDA can be illustrated with a simple example. Consider as system S made of two parallel senders that halt after extruding a fresh name on x and y respectively (in π-calculus, $S \stackrel{\text{def}}{=} (\nu\, u)(x!u.0)|(\nu\, v)(y!v.0)$). The left-most figure below shows the labelled transition system (LTS) of the interleaving semantics of S while the right-most one yields its HDA representation.



In the automaton, names are local to states (in fact their identity is immaterial) and the injective maps among names are represented by dashed arrows (the name z in s2 is mapped to y of s1 while a on the transition from s1 is mapped to x). States of HDA are equipped with *symmetries* (reported below each state in the figure above) namely groups of permutations on the state names that "preserve" its transitions. Locality of names and symmetries permits to represent many traces on the LTS with a single trace in the HDA. For example, the single transition departing from s1 represents both transitions from the uppermost state of the LTS because the symmetry of s1 contains the transposition $(x\, y)$, namely the permutation that exchanges x with y. Thus, the role of x and y can be swapped along the transition of the HDA so to map a to y and z (as well as b) to x.

The LTS semantics of nominal calculi implicitly yields the notion of traces. Remarkably, traces of non-mobile systems can be thought of as words recognised by finite state automata[1]. We contend that a similar link can be established for mobile systems

[1] Finite state automata may possibly recognise infinite words, e.g. Büchi automata.

so to connect formal languages over infinite alphabets and nominal calculi. For this we propose HDA to develop a theory of languages inspired by nominal calculi.

Besides the theoretical insights of this research, we think that it may also yield novel interesting practical advantages. For instance, the ability to recognise languages inspired by nominal calculi paves the way to new verification techniques on mobile systems. For example, we expect that it is possible to determine when two mobile systems are trace equivalent by showing that their computations form languages recognised by the same automaton.

Structure of the paper § 2 briefly summarises the theoretical background of our work. HDA are defined in § 3 which also introduces the notion of language recognised by an HDN. § 4 discusses the expressiveness of our model by comparing it to *finite memory automata*. In § 5, a translation is presented from HDA to a kind of infinite automata, equipped with the ordinary notion of recognition, where the states form a nominal sets. Finally, in § 6, we discuss the role of the symmetries, both to guarantee the existence of canonical models, and to derive efficient algorithms exploiting properties of permutation groups.

2 Background

This section gives a brief account of *nominal sets* [9] and *named sets* [20], tailored to the bare needs of this work. We refer the reader to [9, 20, 10] and references therein for a deeper insight on the topics. The definitions given here are borrowed from [10] (apart from Definition 3, which is introduced here).

2.1 Notational conventions

We write respectively $dom(f)$, $cod(f)$ and $Im(f)$ to denote the domain, codomain and image of a function f; f can also be a partial relation, in which case $dom(f)$ denotes the domain where f is actually defined; $f\ x$ shall denote the application of f to an element in $dom(f)$. We write $f_{|S}$ for the restriction of the function f on its domain to S and id_A for the identity function on a set A.

The extension of a function f on a as b is written $f\left[{}^{b}/_{a}\right] : dom(f) \cup \{a\} \to cod(f) \cup \{b\}$ and defined as usual by stating that $f\left[{}^{b}/_{a}\right]\ a \stackrel{\text{def}}{=} b$ and $f\left[{}^{b}/_{a}\right]\ x \stackrel{\text{def}}{=} f\ x$ for $x \neq a$. Notice that this adds a to the domain of f if $a \notin dom(f)$ or updates f at $a \in dom(f)$. Given a function f and a set of functions G such that $dom(g) = cod(f)$ for any $g \in G$, $G \circ f$ denotes the set $\{g \circ f \mid g \in G\}$. (The set $f \circ G$ is defined similarly).

The set of natural numbers is denoted by $\mathbb{N}$. We denote with *Autf* the set of *finite-kernel* automorphisms, or permutations, of $\mathbb{N}$, that is, those bijections $\pi : \mathbb{N} \to \mathbb{N}$ such that the set $\{x \mid \pi x \neq x\}$ is finite. We recall that *Autf* is a group with respect to the function composition operation whose identity element is the identity function on $\mathbb{N}$.

2.2 Nominal sets

A *nominal set* [9] is a set A equipped with a group action for *Autf*, or *permutation action*, subject to a *finite support* condition, defined below.

Definition 1. *Given a set A, a permutation action maps each* π *in Autf to a function* $\pi_A : A \to A$ *so to preserve identity and composition. A function* $f : A \to B$*, where A and B have an associated permutation action, is called equivariant if it respects the permutation action, namely if for all* $a \in A$ *and* $\pi \in Autf$*,* $f(\pi_A a) = \pi_B(fa)$*.*

In the following we assume a fixed A with an associated permutation action. When no ambiguity is created we omit the subscript A, writing π_A as π. The *symmetry* of an element $a \in A$ (written $Sym\,a$) is the usual notion of symmetry from group theory $Sym\,a \stackrel{\text{def}}{=} \{\pi \in Autf \mid \pi a = a\}$. From this, the notion of *support* is obtained, which generalises the notion of *variables* in terms (with variables) over an algebraic signature [9].

Definition 2. *For each* $N \subseteq \mathbb{N}$*, let* $fix\,N \stackrel{\text{def}}{=} \{\pi \in Autf \mid \forall x \in \mathbb{N}.\pi\, x = x\}$*. When* $fix\,N \subseteq Sym\,a$ *we say that N supports a; if a is supported by a finite subset of* $\mathbb{N}$ *then we say that a is finitely supported. For any finitely supported* $a \in A$ *there is a unique minimal set* $Supp\,a$ *supporting it, called its support.*

Another group theoretical property of an element $a \in A$ is its *orbit* denoted by $Orb\,a$ and defined as $Orb\,a \stackrel{\text{def}}{=} \{\pi a \mid \pi \in Autf\}$. Hereafter, we denote with $Rep\,a$ and with $Norm\,a$ chosen elements of respectively $Orb\,a$ and of the set of *normalising* permutations of a, that is, $\{\pi \in Autf \mid \pi(Rep\,a) = a\}$. The results of this paper do not depend on the choices for $Norm\,a$ (by [10], Lemma 12), and $Rep\,a$.

Orbits enable us to introduce a notion of *finiteness* for nominal sets.

Definition 3. *A nominal set is orbit-finite iff it has a finite number of orbits.*

Orbit-finiteness of nominal sets generalises finiteness of ordinary sets; consider a *finite* set A equipped with the trivial permutation action $\pi a \stackrel{\text{def}}{=} a$ for $a \in A$. The support of each $a \in A$ is $\emptyset$; A is orbit-finite (A can be put in bijective correspondence with its orbits). At the best of our knowledge the notion of Definition 3 has not been used elsewhere in the literature.

Example 1. The set of natural numbers may be equipped with the "natural" permutation action $\pi_{\mathbb{N}} = \pi$ mapping each permutation π to itself. In other words, the permutation $\pi \in Autf$ acts on $n \in \mathbb{N}$ as expected and yields πn. $\Diamond$

It is a simple observation that the support of each $n \in \mathbb{N}$ is the singleton $\{n\}$. Also, as nominal set, $\mathbb{N}$ is orbit-finite; in fact, it has a single orbit.

2.3 Named sets

Let Grp denote the collection of all groups (w.r.t. composition) Φ whose elements are automorphisms of a finite subset of $\mathbb{N}$ (one for each Φ), denoted with $dom(\Phi)$.

Definition 4. *A named set is a pair* $\langle Q, \mathtt{G}_Q \rangle$ *where Q is a set and* $\mathtt{G}_Q : Q \to Grp$*. For each* $q \in Q$*,* $|q|_Q \stackrel{\text{def}}{=} dom(\mathtt{G}_Q(q))$ *is the set of local names of q (we often omit the subscript Q from* $\mathtt{G}_Q(q)$ *and* $|q|_Q$ *and denote a named set* $\langle Q, \mathtt{G}_Q \rangle$ *simply as Q).*

Basically, a named set is a set whose elements are equipped with a group of permutations over a finite set of *local* names. Roughly, these permutations specify when two local names of an element may be permuted without "changing" the element itself.

The essence of the results in [10, 8] (for the scope of this work) is that for each nominal set there is a corresponding named set having the same mathematical properties.

Definition 5. *Given a nominal set* $\langle A, \{\pi_A\}\rangle$ *the corresponding named set is defined as* $\langle A^\bullet, \mathtt{G}_{A^\bullet}\rangle$ *where* $A^\bullet \stackrel{\text{def}}{=} \{Rep\, a \mid a \in A\}$*, and* $\mathtt{G}(Rep\, a) \stackrel{\text{def}}{=} Sym\,(Rep\, a)_{|_{Supp\,(Rep\, a)}}$.

Definition 5 collapses orbits into single elements. Therefore, any orbit-finite nominal set corresponds to a finite named set. The intuitive interpretation of this construction is that quotienting orbits, names lose their "global" meaning (as all the elements obtained from each other by injective relabelling are identified).This has a number of consequences, including the necessity of using name relations in transitions of automata (see Definition 6), and to make the definition of recognised language depend upon an embedding of the local names of the starting state into the global names of strings (Definition 7). We remind the reader to [3, 2] for an extensive discussion of the effects of locality of names.

Example 2. The named set $\mathbb{N}^\bullet$ corresponding to the nominal set of natural numbers (Example 1) is defined as $\langle \{1\}, 1 \mapsto \mathrm{id}_{\{1\}}\rangle$, where 1 is the chosen representative of the only orbit of $\mathbb{N}$; its support is the set $\{1\} \subseteq \mathbb{N}$ and its symmetry is just the identity over $\mathrm{id}_{\{1\}}$. ◊

Example 2 shows that $\mathbb{N}$ corresponds to a named set with just one element.

3 History-dependent automata as recognisers

History-dependent automata (HDA) are a model of computation with names amenable to represent the behaviour of nominal calculi. The adequacy of HDA has been shown in [20, 2, 4] and in [23], where an algorithm based on HDA for checking bisimulation of nominal calculi has been given. This section shows how HDA can be used as language recognisers. For this purpose, we give a simplified definition of HDA.

Given $S \subset \mathbb{N}$ and a permutation π over S, in the following we let $S+1 \stackrel{\text{def}}{=} S \cup \{\star\}$ with $\star \in \mathbb{N} \setminus S$, and $\pi + 1 \stackrel{\text{def}}{=} \pi[^\star/_\star]$. We also let $\Sigma^\bullet$ denote a named set, the alphabet of the automaton.

Definition 6. *A history-dependent automaton (HDN) over* $\Sigma^\bullet$ *is a tuple* $\langle Q, tr, H\rangle$ *where*

- Q *is a named set of states of the automaton (with symmetry* $\mathtt{G}_Q$*);*
- $H \subseteq Q$ *is the named set of accepting states, with symmetry* $\mathtt{G}_Q$ *restricted to* H*;*
- tr *is the transition function returning for each* $q \in Q$ *a finite set of quadruples* $\langle q', a, \sigma_{q'}, \sigma_a\rangle$ *(called transitions) where* $q' \in Q$ *is the destination state,* $a \in \Sigma^\bullet$ *is the label of the transition, and* $\sigma'_q : |q'| \to |q|+1$*,* $\sigma_a : |a| \to |q|+1$ *are two injective functions.*

For each $q \in Q$, we demand the following closure property on $tr\,q$, namely

$$\langle q', a, \sigma_{q'}, \sigma_a \rangle \in tr\,q \implies \langle q', a, (\pi_q + 1) \circ \sigma_{q'} \circ \pi_{q'}, (\pi_q + 1) \circ \sigma_a \circ \pi_a \rangle \in tr\,q \qquad (2)$$

for any $\pi_q \in \mathtt{G}q$, $\pi_{q'} \in \mathtt{G}q'$, and $\pi_a \in \mathtt{G}a$.

The role of $\sigma_{q'}$ (resp. σ_a) is to keep track of the relationship between the *local* names in the destination state q' (resp. the label a) of the transition and the *local* names of the source state q. Transitions have a *name allocation* capability: if a name of the destination state q' or of the label a is mapped to the additional name $\star$, then it is symbolically treated as a "placeholder" which will be used to accept any unknown name.

The HDN in Example 3 is used in the rest of the paper and it is base on a resource-allocation scenario where several resources are shared among a countable and unbound set of processes; at each step of the computation, any available resource can be assigned to an arbitrary process which eventually releases it.

Example 3. For the scenario described above we use $\mathbb{N}$ to identify processes and assume that each process is assigned at most one resource at a time and initially all resources are available. A correct execution requires that all resources are released. The label $\uparrow^1$ (resp. $\downarrow^1$) represents the acquisition (resp. release) of a resource.

The alphabet is the named set $\{\uparrow^1, \downarrow^1\}$, where 1 is the canonical representative of $\mathbb{N}$ (Example 2) with the symmetry $\mathrm{id}_{\{1\}}$ for both $\uparrow^1$ and $\downarrow^1$. The named set of states is $\{q_0, \ldots, q_n\}$ where n is the number of resources, and each q_i has i local names and $\mathtt{G}q_i \stackrel{\text{def}}{=} \{id_{\{1,\ldots,i\}}\}$. The only accepting state is q_0 which has no names ($\mathtt{G}q_0 = \{\mathrm{id}_\emptyset\}$).

For each state q_i, we have an allocation transition $\langle q_{i+1}, \uparrow^1, \sigma, \sigma_1 \rangle$ from q_i where $\sigma_1(1) \stackrel{\text{def}}{=} \star$ and $\sigma \stackrel{\text{def}}{=} \mathrm{id}_{\{1,\ldots,i\}}[^\star/_{i+1}]$. This exploits the name allocation facility of HDA to go from the state q_i to the state q_{i+1} having one more name. Each state q_{i+1} has a release transition $\langle q_i, \downarrow^1, \sigma, \sigma_1 \rangle$ where $\sigma_1(1) \in \{1, \ldots, i+i\}$ and σ maps $\{1, \ldots, i\}$ into $\{1, \ldots, i+1\}$ respecting the order. The injective function σ_1 maps the local name of $\downarrow^1$ into one of q_{i+1} so that $\sigma_1(1)$ is released. $\Diamond$

Example 3 shows that, even if they work on infinite alphabets, HDA still have *finite memory*. Note that the alphabet of Example 3 corresponds (by Definition 5) to the infinite nominal set (having two orbits) $\{\uparrow^x | \, x \in \mathbb{N}\} \cup \{\downarrow^x | \, x \in \mathbb{N}\}$ whose permutation action is $\pi(\uparrow^x) \stackrel{\text{def}}{=} \uparrow^{\pi x}$ and $\pi(\downarrow^x) \stackrel{\text{def}}{=} \downarrow^{\pi x}$.

We now define the recognition process of HDA. In the following, we fix a nominal set (the alphabet) Σ and a HDN $\mathcal{H} \stackrel{\text{def}}{=} \langle Q, tr, H \rangle$ over $\Sigma^\bullet$ (Definition 5), The set of words over Σ is denoted with $\Sigma^\star$ and each word $w \in \Sigma^\star$ is either the empty word ε or the concatenation $a::w'$ of a symbol $a \in \Sigma$ and a word w'. The permutation action over Σ induces a permutation action over $\Sigma^\star$ defined inductively as $\pi\varepsilon \stackrel{\text{def}}{=} \varepsilon$ and $\pi(a::w') \stackrel{\text{def}}{=} \pi a::\pi w'$. It is easily verified that $Supp\,\varepsilon = \emptyset$ and $Supp\,(a::w') = Supp\,a \cup Supp\,w'$.

Definition 7. *A configuration is a triple $\langle q, w, \mathcal{R} \rangle$, where $q \in Q$, $w \in \Sigma^\star$, and $\mathcal{R} : |q| \to Supp\,w$ is an injective partial function. An accepting configuration has the form $\langle q, \varepsilon, \emptyset \rangle$ with $q \in H$ and $\emptyset$ the empty relation (necessarily, since the support of ε is empty).*

A configuration specifies the current state of the automaton, the word in input and a *partial* correspondence between the names of the state and of the word. By the partiality of $\mathcal{R}$, some names of q and w may be initially related, while others will be related subsequently.

The notion of *recognition step* defines how an HDN evolves in a non-deterministic way from a given configuration to a set of configurations.

Definition 8. *Consider a configuration t of the form $\langle q, a::w, \mathcal{R}\rangle$. The recognition step of t, denoted with $step_{\mathcal{H}}(t)$, returns the configurations $\langle q', w, \mathcal{R}'\rangle$ such that there is a transition of the form $\langle q', Repa, \sigma_{q'}, \sigma_{Repa}\rangle \in tr(q)$ and either of the following conditions holds:*

– $\forall x \in dom(\sigma_{Repa}).\sigma_{Repa}\, x \neq \star \implies \begin{cases} \mathcal{R} \circ \sigma_{Repa} = Norma_{|_{Suppa}} & (1) \\ \mathcal{R}' = \mathcal{R} \circ \sigma_{q'} & (2) \end{cases}$

– $\exists x \in dom(\sigma_{Repa}).\sigma_{Repa}\, x = \star \implies \begin{cases} \mathcal{R}\left[{}^{(Norma)(x)}/_{\star}\right] \circ \sigma_{Repa} = Norma_{|_{Suppa}} & (3) \\ \mathcal{R}' = \mathcal{R}\left[{}^{(Norma)(x)}/_{\star}\right] \circ \sigma_{q'} & (4) \\ (Norma)(x) \notin Im\mathcal{R} & (5) \end{cases}$

Equation (1) establishes that the names of the accepted symbol of the word, and of the label of the transition, are related in the correct way by $\mathcal{R}$. Equation (2) computes a relation between the names of q' and the names of w, enabling the process to be iterated. Equation (3) (resp. (4)) has the same purpose as (1) (resp (2)), but it also enforces a correspondence for the name $\star$. Equation (5) asserts that the name $(Norma)(x)$, which corresponds in a to the one mapped by σ_{Repa} into $\star$, must not already correspond by $\mathcal{R}$ to an existing name of q.

Definition 9. *Given a configuration $t \stackrel{\text{def}}{=} \langle q, w, \mathcal{R}\rangle$ with $q \in Q$, $rec_{\mathcal{H}}(t)$ is the recognition function for $\mathcal{H}$ and is defined as*

$$rec_{\mathcal{H}}(t) \stackrel{\text{def}}{=} \begin{cases} \{t\} & \text{if } w = \varepsilon \\ \bigcup_{t' \in step_{\mathcal{H}}(t)} rec_{\mathcal{H}}(t') & \text{otherwise} \end{cases}$$

We say that $\mathcal{H}$ accepts (or recognises) w with $\mathcal{R}$ from q if $\{q' \mid \langle q', \varepsilon, \emptyset\rangle \in rec(\langle q, w, \mathcal{R}\rangle)\} \cap H \neq \emptyset$. We call the set of all such words the language accepted by $\mathcal{H}$ from from the initial state q with $\mathcal{R}$.

Notice that the definition of accepted language is parametrised over the embedding $\mathcal{R}$ of the *local* names of the initial state into the *global* names that may appear in the strings of the language. However, it is possible to define a procedure that *computes* a set of suitable name relations instead of requiring $\mathcal{R}$ as input. A non-deterministic algorithm can be derived from Definition 9, taking as input only a state q and a word w, and using the conditions $(1) - (5)$ as *constraints* imposed to the possible values of $\mathcal{R}$ at each step of the computation. At each step of the recognition process in this case we have a state q, a word w and a set of constraints such that, for any relation $\mathcal{R}$ satisfying

them, w is accepted with $\mathcal{R}$ starting from q. A formal definition of such an algorithm does not present particular difficulties but it is left for future work, together with other algorithmic aspects, such as the definition of an optimised recognition procedure that exploits compact descriptions of permutation groups (more in §6), or the study of union, complementation, and intersection problems on HDA.

Example 4. We consider the language of the HDN of Example 3 from the state q_0. Since q_0 has no local names, we initially have $\mathcal{R} = \emptyset$. A string $w = a_1, \ldots, a_m \in \Sigma^\star$ is recognised iff $rec(\langle q_0, w, \emptyset \rangle) = \{q_0\}$, m is even, $a_1 = \uparrow^x$ for some $x \in \mathbb{N}$, with two additional properties: 1) if $a_i = \uparrow^x$ for a given name x, then $a_j = \downarrow^x$ for some $j > i$, with no other occurrence of $\uparrow^x$ between them; 2) for $j \in \{1, \ldots, m\}$, let k be the difference between the number of occurrences of symbols the form $\uparrow^x$ and $\downarrow^x$ in $w_1, \ldots, w_j$, then $0 \leq k \leq n$ (where n is the number of resources).

4 Expressiveness

In order to show the expressiveness of HDA, we compare them with *finite-memory automata* [12] (FMA, for short). More precisely, we show how FMA can be encoded into HDA. We are aware that more work is still required and we discuss here some of the related issues.

We first give the definition of FMA, borrowed from [12] (apart from some minor notational differences). Let $\aleph$ be an infinite alphabet and $\aleph_\# = \aleph \cup \{\#\}$ where $\# \notin \aleph$ is a distinguished symbol.

Definition 10 (adapted from [12], Def. 1). *A finite-memory automaton (FMA, for short) is a system $A = \langle S, q_0, \mathbf{w}, \mu, \rho, F \rangle$ where*

- *S is a finite set of states and $q_0 \in S$ is the initial state;*
- *$\mathbf{w} : \{1, \ldots, r\} \to \aleph_\#$ is the initial assignment, that is, a map from the set of registers $\{1, \ldots, r\}$ to $\aleph_\#$ such that if $w_i = w_j$ and $i \neq j$ then $w_i = \#$ (following [12], we write w_i for $\mathbf{w}$ i and $w_1 \cdots w_r$ for the assignment (of length r) that maps i to w_i for $i \in \{1, \ldots, r\}$);*
- *$\mu \subseteq S \times \{1, \ldots, r\} \times S$ is the transition relation;*
- *$\rho : S \to \{1, \ldots, r\}$ is a partial function called reassignment;*
- *$F \subseteq S$ is the set of final states.*

Intuitively, an FMA $A = \langle S, q_0, w_1 \cdots w_r, \rho, \mu, F \rangle$ is an automaton equipped with r memory registers that are shared among all the states; each register $i \in \{1, \ldots, r\}$ may either contain a symbol in $\aleph$ ($w_i \in \aleph$) or be empty ($w_i = \#$). Also, transitions of A are triplets $\langle q, i, q' \rangle$ that represents the fact that when A is in the state q and the i-th register contains the next input symbol, then A can move to state q'; conversely, if the input symbol does not appear in any register, it is not consumed and it is assigned to the register $\rho\ s$, provided that $\rho\ s$ is defined.

More precisely, a *configuration* for the automaton $A = \langle S, q_0, w_1 \cdots w_r, \mu, \rho, F \rangle$ is a pair $\langle q, \mathbf{u} \}$ where $s \in S$ and $\mathbf{u}$ is an assignment of length r. The transition relation μ induces the relation μ^c as follows: $\langle \langle q, \mathbf{u} \rangle, a, \langle q', \mathbf{u}' \rangle \rangle \in \mu^c$ for $a \in \aleph$ iff

– if there is $i \in \{1,\dots,r\}$ such that $u_i = a$ then $\mathbf{u} = \mathbf{u}'$ and $\langle q, i, q' \rangle \in \mu$, or
– if $a \notin Im\,\mathbf{u}$ and $\rho(q)$ is defined then $\mathbf{u}' = \mathbf{u}\left[{}^{a}/_{\rho\, q}\right]$ and $\langle q, \rho\, q, q' \rangle \in \mu$

The initial configuration of A is $\langle q_0, w_1 \cdots w_r \rangle$; the notions of run and language $\mathcal{L}_A$ of A are defined as expected.

The two definitions of recognition look similar, with the assignment $\mathbf{u}$ in FMA playing the role of the relation $\mathcal{R}$ in HDA. This is formalised in the rest of the section. We first map FMA to HDA assuming a countable alphabet, which we identify with the (nominal) set $\mathbb{N}$ of natural numbers (see Example 1).

Definition 11. *Let $A = \langle S, q_0, w_1 \cdots w_r, \mu, \rho, F \rangle$ be an FMA and $\langle S, \mathtt{G}_A \rangle$ be a named set with $\mathtt{G}_A\, q = \{id_{\{x_1,\dots,x_r\}}\}$ where all x_i are distinct names.*

The HDN $\mathcal{H}[A]$ corresponding to A is defined as $\langle S, tr_A, F \rangle$ where

– *for each $q \in S$,*

$$tr_A\, q = \{\langle q', 1, id_{\{x_1,\dots,x_r\}}, 1 \mapsto x_i \rangle \mid \langle q, i, q' \rangle \in \mu\}$$
$$\cup\ \{\langle q', 1, id_{\{x_1,\dots,x_r\}}\left[{}^{\star}/_{\rho\, q}\right], 1 \mapsto \star \rangle \mid \rho\, q \text{ is defined} \wedge \langle q, \rho\, q, q' \rangle \in \mu\}$$

– *$\mathtt{G}_F$ is the restriction of $\mathtt{G}_S$ to F.*

Theorem 1. *For each finite-memory automata A with initial assignment* $\mathbf{w}$ *and initial state q, the language accepted by A is the language accepted by $\mathcal{H}[A]$ starting from q with the relation* $\mathbf{w}$ *restricted to $\{x \mid \mathbf{w}(x) \neq \#\}$.*

Theorem 1 proves that HDA are as expressive as FMA. An interesting open question is if HDA are more expressive than FMA. We are currently investigating this aspect; the definition of FMA limits the choice of the reassigned register $\rho\, q$, in each state q, because it depends only from q and can not be different for each transition. It is not clear if this may pose problems to the definition of a mapping from FMA to HDA that preserves and reflects the accepted language.

Another line of research that we are planning to explore is to compare the expressiveness of HDA to other automata models for languages on infinite alphabets like those in [22] (e.g. *pebble automata*).

5 From history dependent automata to infinite automata

This section shows how the language recognised by history dependent automata can be also recognised by possibly infinite non-deterministic automata. In particular, we resort to automata whose set of states and of labels are nominal sets.[2] Indeed, these are also ordinary automata, by forgetting the interpretation of the permutations.

We now define a translation from history dependent automata to such a model. We assume an alphabet Σ which is a nominal set, and the corresponding named set $\Sigma^{\bullet}$. We also assume a history dependent automaton $\langle Q, tr, H \rangle$ over $\Sigma^{\bullet}$, and illustrate how to derive a corresponding ordinary automaton $\langle A, f, K \rangle$, where A is a nominal set, f maps states in A to subsets of $\Sigma \times A$ and describes the (possibly non-deterministic) transition function; finally, K is the set of accepting states.

[2] And whose transition function is equivariant, however this is not relevant here.

Definition 12. *The nominal set of states is* $A \stackrel{\text{def}}{=} \{\langle q, i \circ \mathtt{G}q\rangle \mid i : |q| \to \mathbb{N}\}$ *for i total and injective. The permutation action of A is* $\pi\langle q, i \circ \mathtt{G}q\rangle \stackrel{\text{def}}{=} \langle q, \pi \circ i \circ \mathtt{G}q\rangle$. *The states in K are those with* $q \in H$.

It is easy to see that $Supp\,\langle q, i \circ \mathtt{G}q\rangle = Im\,i$ and[3] $Sym\,\langle q, i \circ \mathtt{G}q\rangle = \{\pi \mid \pi_{|_{cod(i)}} \in i \circ \mathtt{G}q \circ i^{-1}\}$. Intuitively, i embeds the local names and symmetry of q into the global names $\mathbb{N}$ of words.

Definition 13. *The transition function* f *is defined as*

$$f(\langle q, i \circ \mathtt{G}q\rangle) = \bigcup_{\langle q', a, \sigma_{q'}, \sigma_a\rangle \in tr(q)} \{\langle \pi a, \langle q', i[^z/_\star] \circ \sigma_{q'} \circ \mathtt{G}q'\rangle\rangle \mid z \notin Im\,i\}$$

where πa *(labelling each transition) is the action of a permutation such that*[4] $\pi_{|_{Supp\,a}} = i[^z/_\star] \circ \sigma_a$.

Intuitively, for each transition of the HDN, if a name of the label a or of the state q' is mapped to $\star$, an infinite number of transitions is obtained. Otherwise, all the π are equal and a single transition is obtained. Intuitively, if $|q|$ is mapped into $\mathbb{N}$ by i, then the names of q' (resp. a) which are mapped into $|q|+1$ by $\sigma_{q'}$ (resp. σ_a), are mapped into $\mathbb{N}$ with $i[^z/_\star] \circ \sigma_{q'}$ (resp. $i[^z/_\star] \circ \sigma_a$) in an infinite number of ways, by replacing $\star$ with all possible names that are not in $dom(i)$. This allows the automaton to accept any "new" symbol πa.

Theorem 2. *Let* $q \in Q$. *For each total injective function* $i : |q| \to \mathbb{N}$, *the language accepted by* $\langle Q, tr, H\rangle$ *starting from q with i is the language accepted by* $\langle A, f, K\rangle$ *starting from state* $\langle q, i \circ \mathtt{G}q\rangle$.

Theorem 2 strictly depends on the assumption that i is a total function; such assumption requires to map into $\mathbb{N}$ *all* the names of the starting state, including those that may possibly not be used in the HDN. Therefore, such names can not be considered "fresh" during the recognition phase. A weaker result can be proved much in the same way, stating that the HDN accepts w from q with $\mathcal{R}$ (where $\mathcal{R}$ is partial) iff w is accepted in $\langle \mathbf{A}, f, K\rangle$ from state $\langle q, \hat{\mathcal{R}} \circ \mathtt{G}q\rangle$, where $\hat{\mathcal{R}}$ is a suitable completion of $\mathcal{R}$ to a total function that maps the names not in $dom(\mathcal{R})$ outside of $Supp\,w$. The choice of $\hat{\mathcal{R}}$ may determine different starting states in $\langle A, f, K\rangle$ for different words. Here is a very simple example.

Example 5. Let $Q = \{q, q'\}$, with $\mathtt{G}q = \{\mathrm{id}_{\{1\}}\}$ and $\mathtt{G}q' = \{\mathrm{id}_\emptyset\}$. Consider the automaton $\langle Q, tr, \{q'\}\rangle$ over the alphabet $\mathbb{N}^\bullet$ with a the single transition $t = \langle q', 1, \emptyset, 1 \mapsto \star\rangle$ from q. Notice that t allocates a name and does not use the name 1 in q. The language $\mathcal{L}$ accepted by the automaton with the empty relation $\mathcal{R}$ is isomorphic to $\mathbb{N}$ itself, since it is the set of strings in $\mathbb{N}^\star$ of length 1.

[3] Here we view i as an isomorphism from $|q|$ to $Im\,i$ to be able to use its inverse i^{-1}.

[4] The precise choice does not matter by [10], Lemma 12.

Observe that if the total relation $\mathcal{R} = 1 \mapsto x$ (for a given $x \in \mathbb{N}$) was used in Example 5, the automaton would have accepted the language $\mathbb{N} \setminus \{x\}$ which, by Theorem 2, is also the language accepted by $\langle A, f, K\rangle$ from $\langle q, i \circ \mathtt{G}q\rangle$. Since all the states of $\langle A, f, K\rangle$ are in the form $\langle q, i \circ \mathtt{G}q\rangle$, there is no single state that accepts $\mathcal{L}$. Instead, each word whose only symbol is x is accepted starting from all the states except $\langle q, (1 \mapsto x) \circ \mathtt{G}q\rangle$.

6 Symmetries and minimal models

An important question when dealing with models is the existence of canonical ones. For example, the existence of a unique canonical representative of each class of language-equivalent automata allows one to apply a partition refinement method [19] to decide the language equivalence problem. In both the theories of bisimulation and language equivalence canonical models are described by *minimal automata*. In the case of deterministic automata, bisimulation coincides with language equivalence.

Noteworthy, we can use the bisimulation-based theory of HDA to prove language equivalence in the deterministic case. First let us define a deterministic HDN.

Definition 14. *A deterministic history dependent automaton is an automaton such that the corresponding infinite automaton from Definitions 12 and 13 is deterministic.*

The minimal automaton for a deterministic HDN can be obtained using the algorithm for computing the final system given in [7]. As thoroughly explained in [20], without considering symmetries, final systems may not exist. The system computed by the algorithm presented in [7] (using symmetries) has the greatest possible symmetry up-to bisimulation (which is language equivalence for deterministic HDA).

Example 6. (adapted from [20], §6.6.2) Consider the alphabet $\mathbb{N}^\bullet$ (Example 2), the two automata $h_1 = \{Q, tr_1, H\}$ and $h_2 = \{Q, tr_2, H\}$ with $Q = H = \{q\}$, $\mathtt{G}q = \mathrm{id}_{\{1,2\}}$ and

$$tr_1(q) = \{\langle q, 1, \mathrm{id}_{|q|}, \sigma'\rangle, \langle q, 1, \mathrm{id}_{|q|}, \sigma''\rangle\}$$

$$tr_2(q) = \{\langle q, 1, swap_{|q|}, \sigma'\rangle, \langle q, 1, swap_{|q|}, \sigma''\rangle\}$$

where we denote with $swap_{|q|}$ the function swapping the two names of q, and with σ' (resp. σ'') the function mapping 1 to 1 (resp. to 2). The two automata recognise the same strings with the same mappings, and they are already minimal. Their canonical representative is the automaton $h = \langle Q, tr, H\rangle$ where now the symmetry of q is $\{\mathrm{id}_{\{1,2\}}, swap_{\{1,2\}}\}$, and $tr(q) = tr_1(q) \cup tr_2(q)$.

Minimisation is only possible for HDA with symmetries; the next example illustrates a case where the minimal automaton contains states with non trivial symmetries.

Example 7. The minimal automaton corresponding to the one of Example 3 is $\langle Q, tr, H\rangle$ where $Q = \{q_i | i \in \{0, \ldots, n\}\}$ and $H = \{q_0\}$ The symmetry of each state q_i is the *symmetric* group over $\{1, \ldots, i\}$, that is, it contains all the permutations of this set.

For each $i \in \{0, \ldots, n-1\}$ we have an allocation $\uparrow^1$ from the state q_i to $\langle q_{i+1}, \uparrow^1, \sigma_q, \sigma_1\rangle$, where $\sigma_1(1) \stackrel{\text{def}}{=} \star$ and σ_q is the identity on $\{1, \ldots, i\}$, and sends $i+1$ to $\star$. For

each name $i \in \{1,\dots,n\}$ we have a release transition, from q_i to $\langle q_{i-1}, \downarrow^1, \sigma_q, \sigma_1 \rangle$ where $\sigma_1(1) = i$, and σ_q is the inclusion of $\{1,\dots,i-1\}$ into $\{1,\dots,i\}$. The remaining transitions are obtained by closure with respect to the symmetry of states (closure property, Definition 6). $\Diamond$

An interesting remark to add to Example 7 regards the conciseness of HDA with symmetries. In fact, symmetries have a compact representation; this is due to a nice theorem of group theory: a finite permutation group G of cardinality $|G|$ can be conveniently represented using a set of *generators* [1] of size smaller than $\log_2 |G|$ (see e.g. [11], §3.3), whose closure w.r.t. composition is G.

Example 8. In Example 7, we only used the symmetric group over each set $\{1,\dots,n\}$. Such a group, for all $n > 1$, can be represented using two generators: a *swap* between two arbitrary elements and a *round shift* of all the elements. Therefore, the minimal automaton is actually a compressed representation of the original one. $\Diamond$

7 Concluding remarks

We have studied a class of automata over infinite alphabets, whose main feature is the presence of *names* and *name allocation*. We have shown how these can be mapped into ordinary, possibly infinite automata, and how the presence of *local* names gives a notion of memory to the considered models, so that they can faithfully represent finite-memory automata.

We regard this work as the grounds of a theory of resource-allocation and symmetry-reduction for automata and languages, aimed at using these models for applications such as *trace analysis* and *run-time monitoring* of systems that may have to deal with infinite sets of resources. Therefore, we plan to investigate efficient recognition procedures, making use of the compact representation of a group by its generators, and of well known polynomial time algorithms for permutation groups [13]. We have seen that FMA can be faithfully encoded in HDA. An advantage of using HDA in place of FMA is the presence of symmetries, that we discussed in §6. It will be important, starting from this assumption, to study the computational properties, and the efficiency of algorithms related to the use of HDA as language recognisers, also to advance, as a side effect, the research on FMA and other models based on finite memory.

A parallel research line is that of characterising the *languages* recognised by HDA by the means of grammars with names, or generalised forms of regular expressions, possibly exploiting category-theoretical methods.

A valuable application of the framework is the possibility to use *process calculi* such as the *CCS* or the π-calculus to *specify* HDA. Using the semantics of various process calculi over HDA (e.g. [17, 20, 7]), and the results in this work, it is possible to use parallel composition, replication and synchronisation as operators to specify languages (e.g., to specify monitors, trace-analyses, and parsers) for systems with infinite alphabets.

Of high interest is also the study of closure properties of HDA w.r.t. set-theoretical operations. The efficient implementation of such operations may lead to the design of novel space-efficient data structures. In particular, an application we have in mind is

that of *storing* large quantities of traces from a system that has some intrinsic symmetry. This is a hot topic in computer science due to the necessity to record large quantities of data logs (e.g. the *clickstream* of an user). These logs frequently have causally independent sub-traces, whose order does not matter, causing an unuseful exponential blowup in the stored data.

References

1. Charles C. Sims. Computational methods in the study of permutation groups. In John Leech, editor, *Computational Problems in Abstract Algebra*, pages 169–183. Pergamon (Oxford), 1970.
2. V. Ciancia. *Accessible Functors and Final Coalgebras for Named Sets*. PhD thesis, University of Pisa, 2008.
3. V. Ciancia and U. Montanari. A name abstraction functor for named sets. *Electr. Notes Theor. Comput. Sci.*, 203(5):49–70, 2008.
4. G. Ferrari, U. Montanari, and M. Pistore. Minimizing Transition Systems for Name Passing Calculi: A Co-algebraic Formulation. In M. Nielsen and U. Engberg, editors, *FOSSACS 2002*, volume 2303 of *LNCS*, pages 129–143. Springer, 2002.
5. G. Ferrari, U. Montanari, and E. Tuosto. Coalgebraic Minimisation of HD-automata for the π-Calculus in a Polymorphic λ-Calculus. *TCS*, 331:325–365, 2005.
6. G. Ferrari, U. Montanari, E. Tuosto, B. Victor, and K. Yemane. Modelling and Minimising the Fusion Calculus Using HD-Automata. In J. Fiadeiro, N. Harman, M. Roggenbach, and J. Rutten, editors, *Algebra and Coalgebra in Computer Science*, volume 3629 of *LNCS*, pages 142 – 156. Springer, 2005.
7. G. L. Ferrari, U. Montanari, and E. Tuosto. Coalgebraic minimization of hd-automata for the pi-calculus using polymorphic types. *Theor. Comput. Sci.*, 331(2-3):325–365, 2005.
8. M. Fiore and S. Staton. Comparing Operational Models of Name-Passing Process Calculi. *Inf. and Comp.*, 4(204):524–560, 2006.
9. M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In G. Longo, editor, *Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS'99)*, pages 214–224, Trento, Italy, 1999. IEEE Computer Society Press.
10. F. Gadducci, M. Miculan, and U. Montanari. About permutation algebras, (pre)sheaves and named sets. *Higher-Order and Symbolic Computation*, 19(2-3):283–304, 2006.
11. John D. Dixon and Brian Mortimer. *Permutation Groups*, volume 163 of *Graduate Texts in Mathematics*. Springer-Verlag, 1996.
12. N. Kaminski, Michael Francez. Finite-memory automata. *TCS*, 134(2):329–363, 94.
13. E. M. Luks. Permutation Groups and Polynomial Time Computation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 11:139–175, 1993.
14. R. Milner. *Communicating and Mobile Systems: the π-calculus*. CUP, 1999.
15. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, I and II. *Inf. and Comp.*, 100(1):1–40,41–77, September 1992.
16. U. Montanari and M. Pistore. Checking Bisimilarity for Finitary π-Calculus. In I. Lee and S. Smolka, editors, *CONCUR*, volume 962 of *LNCS*, pages 42–56, Philadelphia, PA, USA, Aug. 1995. Springer.
17. U. Montanari and M. Pistore. Minimal transition systems for history-preserving bisimulation. In Proc. STACS'97*, LNCS 1200*. Springer Verlag, 1997.
18. U. Montanari and M. Pistore. π-Calculus, Structured Coalgebras, and Minimal HD-Automata. In M. Nielsen and B. Roman, editors, *MFCS*, volume 1983 of *LNCS*. Springer, 2000.

19. R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM J. Comput.*, 16(6):973–989, December 1987.
20. M. Pistore. *History Dependent Automata*. PhD thesis, Università di Pisa, Dipartimento di Informatica, 1999. available at University of Pisa as PhD. Thesis TD-5/99.
21. D. Sangiorgi and D. Walker. *The π-Calculus: a Theory of Mobile Processes*. CUP, 2002.
22. L. Segoufin. Automata and Logics for Words and Trees over an Infinite Alphabet. In *Computer Science Logic*, volume 4207 of *LNCS*, pages 41–57. Springer, 2006.
23. E. Tuosto. *Non-Functional Aspects of Wide Area Network Programming*. PhD thesis, Dipartimento di Informatica, Università di Pisa, May 2003. TD-8/03.

A Proofs

Proof. (*Theorem 2)* We prove the "only if" direction, that is, we assume that the history-dependent automaton is accepting and prove that also the one defined on nominal sets is. The other direction of the proof is easily obtained by a similar case analysis. If the word is empty, and the tuple is accepting, then $q \in H$ and therefore $q, \sigma_q \circ \mathtt{G}q \in K$, hence this case is proved. If we have a non-empty word $a::w$, according to Definition 8, we know that there is a configuration $\langle q', Rep\,a, \sigma_{q'}, \sigma_{Rep\,a}\rangle \in tr\,q$ and we have to inspect two cases.

Case 1: $Im\sigma_{Rep\,a} \subseteq |q|$. By Definition 8, the configuration $\langle q, a::w, i\rangle$ goes in one step to $\langle q', w, i \circ \sigma_{q'}\rangle$, which in turn reaches an accepting configuration. We can not apply the induction hypothesis yet, because $i \circ \sigma_{q'}$ is not necessarily a total relation: $\star$ may belong to $Im\sigma_{q'}$. This is easily rectified by looking at the tuple $\langle q, w, i'\rangle$ where $i' = i[^z/_\star] \circ \sigma_{q'}$, for $z \notin Supp\,w \cup Im\,i$, reaching the same accepting configuration (see Lemma 1 below). The choice of z is crucial for the proof: it is both outside of $Im\,i$, to match with Definition 13, and of $Supp\,w$, so that the configuration still accepts the word (notice that $Im(i \circ \sigma_{q'}) \subseteq Im\,i$). By the induction hypothesis we know that $\langle q', i' \circ \mathtt{G}q'\rangle$ accepts w in $\langle A, f, K\rangle$. We just need to exhibit a transition $\langle q, i \circ \mathtt{G}q\rangle \xrightarrow{a} \langle q', i' \circ \mathtt{G}q'\rangle$ in the latter automaton. By looking at Definition 13, there is a transition

$$\langle q, i \circ \mathtt{G}_q\rangle \xrightarrow{(i[^z/_\star] \circ \sigma_{Rep\,a}))(Rep\,a)} \langle q', i[^z/_\star] \circ \sigma_{q'} \circ \mathtt{G}q'\rangle$$

By $Im\sigma_{Rep\,a} \subseteq |q|$, $i[^z/_\star] \circ \sigma_{Rep\,a}(Rep\,a) = i \circ \sigma_{Rep\,a}(Rep\,a) = Norm\,a_{|_{Supp\,a}}(Rep\,a) = a$, so the label is correct. The destination state is precisely obtained by the definition of i'.

Case 2: $\sigma_{Rep\,a}(x) = \star$ for exactly one x due to injectivity of $\sigma_{Rep\,a}$. We then let $z = (Norm\,a)(x)$, which does not belong to $Im\,i$ by condition (5) in Definition 8. Therefore, substituting $(Norm\,a)(x)$ for z in the transition used in the previous case, we obtain again a transition in $\langle A, f, K\rangle$. Using conditions (3) and (4) of Definition 8, we prove that it is the desired transition.

□

Proof. (Theorem 1) The proof is based on the fact that the transitions outgoing from the configurations of A are in one-to-one correspondence with those of $\mathcal{H}[A]$. More precisely,

$$((q, \mathbf{u}), a, (q', \mathbf{u}')) \in \mu^c \iff \langle q', \varepsilon, \mathcal{R}'_{\mathbf{u}'}\rangle \in step(\langle q, a, \mathcal{R}_{\mathbf{u}}\rangle), \text{ for a given } \mathcal{R}'_{\mathbf{u}'}$$

where $\mathcal{R}_{\mathbf{u}} : x_i \mapsto u_i$ provided that $u_i \neq \#$.

In fact, $((q,\mathbf{u}),a,(q',\mathbf{u}')) \in \mu^c$ either because (†) there is $i \in \{1,\ldots,r\}$ such that $u_i = a$ and $\langle q,i,q'\rangle \in \mu$ or (‡) for no $i \in \{1,\ldots,r\}$ $u_i = a$, $\rho\, q$ is defined, $\mathbf{u}' = \mathbf{u}\left[{}^{a}/_{\rho\ q}\right]$, and $\langle q,\rho\ q,q'\rangle \in \mu$. Then we proceed by case analysis (taking into account the construction of $\mathcal{H}[A]$);

(†) there is $\langle q',1,id_{\{x_1,\ldots,x_r\}},1 \mapsto x_i\rangle \in tr_A\, q$ and by (1) and (2) in Definition 8 we obtain the thesis by taking $\mathcal{R}'_{\mathbf{u}'} = \mathcal{R}_{\mathbf{u}} \circ id_{\{x_1,\ldots,x_r\}} = \mathcal{R}_{\mathbf{u}}$;

(‡) there is $\langle q',1,id_{\{x_1,\ldots,x_r\}}\left[{}^{\star}/_{\rho\, q}\right],1 \mapsto \star\rangle \in tr_A\, q$ and the thesis follows by observing that (3), (4), and (5) in Definition 8 hold if we take $\mathcal{R}'_{\mathbf{u}'} = \mathcal{R}_{\mathbf{u}}\left[{}^{a}/_{x_{\rho q}}\right]$. In fact, since $Norm\, a = 1 \mapsto a$,

- (3) holds because $Norm\, a = \mathcal{R}_{\mathbf{u}}[{}^{a}/_{\star}] \circ (1 \mapsto \star)$,
- (4) holds because $\sigma_{q'} = id_{\{x_1,\ldots,x_r\}}\left[{}^{\star}/_{\rho q}\right]$ and, by definition of $\mathcal{R}'_{\mathbf{u}'}$, $\mathcal{R}_{\mathbf{u}}[{}^{a}/_{\star}] \circ \sigma_{q'} = \mathcal{R}'_{\mathbf{u}'}$, and
- (5) holds by hypothesis since $(Norm\, a)\, 1 = a \notin \{u_1,\ldots,u_r\}$.

□

Lemma 1. *If the configuration $\langle q,w,\mathcal{R}\rangle$ reaches by step an accepting configuration C, then also $\langle q,w,\hat{\mathcal{R}}\rangle$ reaches C, where $\hat{\mathcal{R}} : |q| \to \mathbb{N}$ is a total injective function obtained by extending $\mathcal{R}$ in such a way that $\hat{\mathcal{R}}(|q| \setminus dom(\mathcal{R})) \cap (Supp\, w \cup Im\mathcal{R}) = \emptyset$.*

Proof. (sketch) Just observe that a symbol a can be accepted by the HDN with $\mathcal{R}$ only if either its names are in $Im\mathcal{R}$, by condition (1) in Definition 8, or one of them is not mapped, and it is accepted due to a name of the label of one transition mapped to $\star$. The latter can only happen if such name does not belong to $Im\mathcal{R}$ by condition (5) in Definition 8. If the symbol is accepted, its names are in $Im\,\mathcal{R}$ and they have corresponding names in $dom(\mathcal{R})$; therefore these names are not affected by the extension $\hat{\mathcal{R}}$. If the symbol is not accepted, then it can't be accepted using $\hat{\mathcal{R}}$ either: if mapping the names in $|q| \setminus dom(\mathcal{R})$ outside of $Supp\, w$ satisfies the above conditions, then also $\mathcal{R}$ does. □