The Languages of History-Dependent Automata

Vincenzo Ciancia[1], Emilio Tuosto[2], and Nikos Tzevelekos[3]

[1] Institute for Logic, Language and Computation, University of Amsterdam
[2] Department of Computer Science, University of Leicester
[3] Department of Computer Science, University of Oxford

Abstract. History-Dependent Automata have been seminal in the study and automation of mobile process calculi, and have contributed substantial insights in the specification of computation with names. However successful, these machines have not been studied as language acceptors. We initiate just such a study here. We introduce the notion of language-accepting history-dependent automata over infinite alphabets. Moreover, we relate them to existing formalisms for languages over infinite alphabets, in particular to the recently introduced Fresh-Register Automata. Finally, we examine their closure properties and their inherent minimisation capabilities, the latter due to the incorporated notion of symmetries.

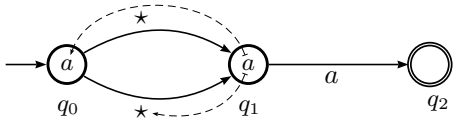
1 Introduction

Automata and languages over infinite alphabets have seen a remarkably increasing interest over the last few years. The motivation has stemmed from a range of seemingly distant fields where infinite alphabets naturally arise: from process calculi and the study of mobility; to programming languages and the semantics of generative dynamic behaviours; and to XML-based languages and the ensuing formal languages over data words. Such formalisms include *register automata* [14], *pebble automata* [21] and *data monoids* [1], to mention but a few. A particularly successful formalism for mobile process calculi which has not been studied yet in relation to language acceptance is that of *History-Dependent Automata (HDAs)*. We initiate just such a study here.

The initial motivation behind HDAs was the introduction of an algorithmic, syntax-free description of mobile computation [18, 22] emerging from agents which interact in a dynamically evolving communication environment. The dynamics of the environment is captured by the notion of *name* which can be created fresh and exchanged by agents. Names are pervasive in mobility and can be used to represent communication channels, threads, sessions, cryptographic keys, etc. For instance, using names to represent channel identifiers one implements, through fresh name creation and dissemination (*extrusion*), the dynamic creation of new private communication channels. This is done e.g. in the π-calculus [17], the paradigmatic language for mobile computation.

A language-accepting HDA consists of a set of states, each equipped with a finite set of names. Apart from the initial state, the specific choice of names for each state is immaterial. Such names have internal scope: they act as name-representatives, or place-holders, for the actual names which populate the state of the automaton during computation. On the other hand, the initial state contains actual names upon which the automaton will base its computation. Each transition of the automaton contains a label involving names from the source state together with an injective map which matches

the names of the target state to those of the source, with the provision that some names in the label or in the target may not correspond to any name in the source state but rather to "fresh" names. Following these maps from a given state in the computation back to the initial state and adding also fresh names on the way, the representative names of each state are mapped to actual names. The inputs to be accepted at each stage in the computation involve precisely these latter names. Consider, for example, the following automaton $\mathcal{H}$ with states q_0, q_1, and q_2



where both q_1 and q_2 contain a single name a while q_2 contains no names. The symbol $\star$ represents a fresh name. Thus, following the uppermost transition from q_0 to q_1, $\mathcal{H}$ will accept a fresh name, say b, and move to q_1; the mapping of the transition stipulates that the internal name a in q_1 represents, in fact, the name a. Then $\mathcal{H}$ will accept the symbol a and terminate in q_2. On the other hand, if $\mathcal{H}$ follows the lowermost transition from q_0 to q_1 at the beginning, it will accept a fresh b and move to q_1 but, this time, the name a in q_1 will represent the fresh name b. Thus, in the next step the automaton will accept b and terminate. Note how the same state q_1 is instantiated with two different names in each of the traversals of $\mathcal{H}$. Hence, the language accepted comprises all words ba and bb where a is fixed and $b \neq a$.

In this paper we introduce the notion of language-accepting HDA (§3) and establish the closure properties and expressiveness of these automata (§4). In doing so, we compare them to *Register Automata (RAs)* and in particular we single out classes of HDAs which precisely correspond to register automata [14] and fresh-register automata [25] respectively. As a result, language equivalence is undecidable for non-deterministic HDAs operating on an alphabet of letters of unary size, but we can decide emptiness and bisimilarity. We then turn our attention to canonical representations of our machines (§5) through minimal models. When computing with names and freshness, as observed in [22] and recently rediscovered in [2], in order to compute minimal representatives it is necessary to use some information about the *symmetry* of internal names of states, specifying which ones have equivalent roles. HDAs are constructed inside the theory of *named sets* [22] which naturally features name symmetries. For a class of HDAs called *symmetrised-deterministic* HDAs, we show that minimal models can be computed by use of a partition refinement algorithm.

Related work The research on *Register Automata (RAs)* [14, 21], is the most relevant to our work. HDAs and RAs are similar machines but they hinge on different principles and have been independently developed by communities focussed on different goals. For example, HDAs yield an automata-theoretic model of process calculi and have been extensively examined under the lens of effective representation and minimisation [9], which has led to practical implementations [10]. On the more technical level, HDAs are higher-level machines with a built-in notion of symmetry, which permit operations on automata (e.g. minimisation, composition) to be implemented in a direct, relabelling-free fashion. We envisage these two approaches as complementary.

Another very relevant recent work is [2], which studies notions of automata over structures generalising nominal sets [11] by use of canonical categorical constructions. The approach provides a generalised framework for nominal automata. Our approach, on the other hand, is at a more concrete level: it provides specific automata constructions and establishes their intrinsic properties. Moreover, the automata we consider take into account local and global freshness. We find these research directions closely connected and expect to relate them in future work.

Other related recent works on nominal languages are [12, 15]. In [12] algebraic descriptions of nominal languages (in particular, *nominal Kleene algebras*) are studied together with representation properties of their languages. On the other hand, the languages examined in [15] involve words with binders and yield automata with stack structures, much different from the ones examined herein.

2 Background

Notational conventions. Let $f : A \to B$ be a function. The domain and the image of f are respectively denoted as $dom(f)$ and $Im(f)$; if f is partial (written $f : A \rightharpoonup B$), $dom(f) \stackrel{\text{def}}{=} \{x \in A \mid \exists y \in B. f(x) = y\}$. Also, $f_{|_S}$ is the restriction of f on $S \subseteq A$ and id_A is the identity on a set A. The update at a with b of f is the function such that $f[a \mapsto b](a) \stackrel{\text{def}}{=} b$, and $f[a \mapsto b](x) \stackrel{\text{def}}{=} f(x)$ if $x \neq a$.

We give a brief account of *nominal sets* [11] and *named sets* [22].

Through the paper we fix $\mathbb{A}$ to be a countably infinite set of *atoms* or *names*. We write $Perm$ for the group of finite permutations of $\mathbb{A}$, i.e. those bijections $\pi : \mathbb{A} \to \mathbb{A}$ such that the set $\{a \mid \pi(a) \neq a\}$ is finite. The transposition of $a, b \in \mathbb{A}$ is denoted by $(a\ b)$ and, for each finite $S \subseteq \mathbb{A}$, $Grp(S)$ is the collection of all groups (wrt composition) of permutations on S.

Definition 1. *A **nominal set** is a set X along with an action for $Perm$, that is, a function $\cdot : Perm \times X \to X$ such that, for all $x \in X$ and $\pi, \pi' \in Perm$: $id_{\mathbb{A}} \cdot x = x$ and $(\pi \circ \pi') \cdot x = \pi \cdot (\pi' \cdot x)$. Also, each $x \in X$ has **finite support**, namely, there exists a finite $S \subseteq \mathbb{A}$ such that, for all $\pi \in Perm$, $\pi_{|_S} = id_S \implies \pi \cdot x = x$.*

We shall write each nominal set $(X, \cdot)$ simply as X. For any $x \in X$, there is a unique minimal set $Supp(x) \subseteq \mathbb{A}$ supporting it, called *the support of* x. Given two permutations π and π', whenever $\pi_{|_{Supp(x)}} = \pi'_{|_{Supp(x)}}$, we have $\pi \cdot x = \pi' \cdot x$. Therefore, an injective function ρ into $\mathbb{A}$ whose domain includes $Supp(x)$ can also be "applied" to x by a choice of a permutation that agrees with ρ on $Supp(x)$. We may make no distinction between the two cases and write $\rho \cdot x$ when ρ is defined on $Supp(x)$.

See Appendix A for a few examples of nominal sets.

Definition 2. *A **named set** is a pair $(Q, \mathtt{G}_Q)$ where Q is a set and $\mathtt{G}_Q : Q \to \bigcup_S Grp(S)$ associates to each element its **symmetry**. For each $q \in Q$, with $\mathtt{G}_Q(q) \in Grp(S)$, the set of **internal names** of q is $|q|_Q \stackrel{def}{=} S$.*

The elements of a named set are equipped with a group of permutations over a finite set of internal names. These permutations establish how the internal names of an element may be exchanged preserving its intended semantics. Note that in general neither a named set $(Q, \mathtt{G}_Q)$ nor its underlying set Q are nominal sets. We often omit subscripts from $\mathtt{G}_Q$ and $|\_|_Q$ and denote a named set $(Q, \mathtt{G}_Q)$ simply by Q.

3 History-dependent automata as acceptors

Hereafter, we fix a nominal set Σ as our alphabet of *letters*. Thus, letters may feature arbitrarily many names, symmetries, etc. (see Appendix A). In particular, our letters include distinguished constants $\star$ and $\circledast$ which will be used to represent *name-freshness*.

Definition 3. *Let Q be a named set and $q \in Q$. A **transition** from q is a tuple $t = (q', \ell, \sigma, f)$, written $q \xrightarrow[\sigma]{\ell, f} q'$, where:*

- *$q' \in Q$ is the destination state and $\ell \in \Sigma$ is the label of t,*
- *$\sigma : |q'| \to |q| \cup Im(f)$ is an injective function,*
- *the freshness annotation $f : \{\star, \circledast\} \rightharpoonup Supp(\ell)$ is an injective partial map such that $|q| \cap Im(f) = \emptyset$ and $Supp(\ell) \subseteq |q| \cup Im(f)$.*

The role of σ is to relate the local names of the destination state q' of the transition and the local names of the source state q. Transitions have a *name allocation* capability, expressed by the freshness annotation f. A name $a \in Im(f)$ is symbolically treated as a place-holder which will be used to accept any *fresh* name. In particular, there are two levels of freshness: if $a = f(\star)$ then a represents any *locally fresh* name, i.e. any b which does not occur in the current computation step; if $a = f(\circledast)$ then a represents any *globally fresh* name, i.e. any b which has not occurred in the whole computation.

Definition 4. *A **history-dependent automaton** (HDA) over Σ is a quadruple $\mathcal{H} = (Q, tr, q_0, F)$ where:*

- *Q is a finite named set of states, $q_0 \in Q$ the initial state, and $F \subseteq Q$ the final ones;*
- *tr is the transition function mapping each $q \in Q$ to a finite set of transitions from q.*

Additionally, for each $q \in Q$ and $\pi_q \in \mathtt{G}(q)$, the following closure property must hold

$$(q', \ell, \sigma, f) \in tr(q) \implies (q', \pi[f] \cdot \ell, \pi[f] \circ \sigma, f) \in tr(q) \tag{1}$$

where $\pi[f] \stackrel{def}{=} \pi_q \cup \{(f(u), f(u)) \mid u \in \{\star, \circledast\} \cap dom(f)\}$.

The closure property (1) makes HDAs compressible: the set of transitions is *saturated* using the symmetry of the source state, so that a canonical choice of a transition may be used to represent all the generated ones. The states of an HDA form a named set. In effect, names in states are internal and have a place-holding utility. The accepted words are built on the actual names instantiating them. Thus, the semantics of a state q is defined by use of an assignment $\rho : |q| \to \mathbb{A}$ to actual names. Moreover, a history H of occurred names is needed in order to implement global name-freshness. These constructs are packed into *configurations*, which are the semantic counterparts of states.

Fix an HDA $\mathcal{H} = (Q, tr, q_0, F)$. The set $\mathcal{C}$ of ***configurations*** on Q consists of all triples (q, ρ, H) such that $\rho : |q| \to \mathbb{A}$ is an injective function and $H \subseteq \mathbb{A}$ is finite. Acceptance for $\mathcal{H}$ relies on the notion of *configuration graph*, defining which symbols of the alphabet are accepted in a configuration (q, ρ, H). Let $\Sigma^\star$ be the *words over* Σ (namely the finite sequences of letters of Σ), ϵ denote the empty word, and let ℓw denote the concatenation of $\ell \in \Sigma$ and $w \in \Sigma^\star$. The permutation action over Σ induces the permutation action over $\Sigma^\star$ defined by $\pi \cdot \epsilon \stackrel{\text{def}}{=} \epsilon$ and $\pi \cdot (\ell w') \stackrel{\text{def}}{=} (\pi \cdot \ell)(\pi \cdot w')$.

Definition 5. *Let $\rightarrow \subseteq \mathcal{C} \times \Sigma \times \mathcal{C}$ be the smallest relation induced by the rule:*

$$\frac{q \xrightarrow[\sigma]{\ell, f} q'}{(q, \rho, H) \xrightarrow{\rho' \cdot \ell} (q', \rho' \circ \sigma, H \cup Supp(\rho' \cdot \ell))} \quad \rho' = \begin{cases} \rho & \text{if } dom(f) = \emptyset \\ \rho[f(\star) \mapsto a] & \text{if } dom(f) = \{\star\} \\ \rho[f(\circledast) \mapsto b] & \text{if } dom(f) = \{\circledast\} \\ \rho[f(\star) \mapsto a][f(\circledast) \mapsto b] & \text{if } dom(f) = \{\star, \circledast\} \end{cases}$$

for all $a \in \mathbb{A} \setminus Im(\rho)$ and $b \in \mathbb{A} \setminus (H \cup |q_0| \cup \{a\})$.
*The **configuration graph** of $\mathcal{H}$ is the graph corresponding to the LTS of $\rightarrow$. We write $\twoheadrightarrow$ for the extension of $\rightarrow$ to finite words.*[1] *The language accepted by (q, ρ, H) is:*

$$\mathcal{L}(q, \rho, H) = \{w \in \Sigma^\star \mid (q, \rho, H) \xrightarrow{w} \twoheadrightarrow (q', \rho', H') \ \wedge \ q' \in F\}.$$

*The **language accepted by** $\mathcal{H}$ is $\mathcal{L}(q_0, id_{|q_0|}, \emptyset)$.*

The above definition of reduction between configurations is essentially split into four cases according to the domain of the freshness annotation f. The first case specifies the semantics of transitions that do not allocate names. Notice the role of σ which relates the internal names of the destination state q' to those of q, which are mapped into actual names by ρ. The composition $\rho \circ \sigma$ therefore updates the map ρ so that it correctly operates on the names of q'. Since the names of each label ℓ outgoing from q are also internal to q itself, ρ is applied to ℓ in turn. The history H is enlarged to keep track of the names used in each label. The next two rules are similar, respectively dealing with local and global freshness; the former requires a new name to be fresh w.r.t. the names assigned to q by ρ. The latter avoids *all names of the current computation*, namely all names which have appeared in the past, therefore they are in H, and all names in the initial state.[2] The last rule deals with the simultaneous allocation of locally and globally fresh names; this requires to choose two names that differ from any names in the state (like a) or any name in the computation (like b). Notice that, in the target configuration, ρ' has to be pre-composed with σ so to guarantee that the names of q' are correctly interpreted. Also, the configuration graph of $\mathcal{H}$ may be infinite, while $\mathcal{H}$ is not.

The following proposition establishes *equivariance* [11] of the map associating to each configuration its language, and clarifies the role of symmetries.

Proposition 6. *If $(q, \rho, H) \in \mathcal{C}$ and $\rho' : \mathbb{A} \to \mathbb{A}$ is an injective map, then $\mathcal{L}(q, \rho' \circ \rho, \rho'(H)) = \rho' \cdot \mathcal{L}(q, \rho, H)$. Also, for all $\pi \in \mathtt{G}(q)$, $\mathcal{L}(q, \pi, H) = \mathcal{L}(q, id_{|q|}, H)$.*

Remark 7. Global freshness makes automata more expressive, as studied in [25], and requires the history to be carried along during computation. Note that, although the whole history of the computation is kept in each configuration, the automata can use it only to establish global freshness and have otherwise no access to it, e.g. they are not in position to check whether a specific name has appeared before another one in the history. On the other hand, the sub-class of HDAs where the global freshness symbol $\circledast$ does not appear enjoys a memory-bound acceptance process, as the H component of configurations can be safely removed.

[1] Formally, we set: $(q, \rho, H) \xrightarrow{\epsilon} \twoheadrightarrow (q, \rho, H)$ and $(q, \rho, H) \xrightarrow{\ell w} \twoheadrightarrow (q', \rho', H')$ if there is (q'', ρ'', H'') such that $(q, \rho, H) \xrightarrow{\ell} (q'', \rho'', H'') \xrightarrow{w} \twoheadrightarrow (q', \rho', H')$.

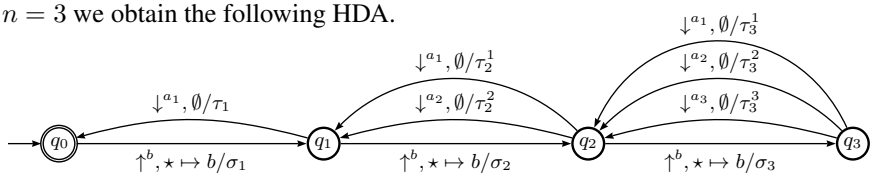
[2] This is because of our design choice to include an empty history in initial configurations.

Example 8. Consider an idealised system where an unbounded number of processes can access, in a mutually exclusive way, a fixed number of shared identical resources. Each process can use at most one resource at a time, by acquiring and eventually releasing it. We construct an HDA whose language characterises the correct executions of the system, that is, those in which each process acquires at most one resource at a time and eventually releases it. Such a language contains words of arbitrary length made of letters drawn from an infinite alphabet.

Suppose there are n shared resources. We identify processes by atoms in $\mathbb{A}$ and take our alphabet to consist of letters $\uparrow^a$ or $\downarrow^a$ denoting the acquisition or release of a resource by process a respectively. The permutation action is defined by: $\pi\cdot\uparrow^a \stackrel{\text{def}}{=} \uparrow^{\pi(a)}$ and $\pi\cdot\downarrow^a \stackrel{\text{def}}{=} \downarrow^{\pi(a)}$. Let $a_1,\ldots,a_n \in \mathbb{A}$ be n distinct names. We define an HDA $\mathcal{H}_n$ as follows. The named set of states is $\{q_0,\ldots,q_n\}$, $|q_i| \stackrel{\text{def}}{=} \{a_1,\ldots,a_i\}$ and $\mathtt{G}q_i \stackrel{\text{def}}{=} \{id_{\{a_1,\ldots,a_i\}}\}$.[3] The initial state is q_0, which is also the only accepting state. For all $i \in \{0,\ldots,n-1\}$,

- q_i has one $q_i \xrightarrow[\sigma]{\uparrow^b,\star\mapsto b} q_{i+1}$, where $\sigma \stackrel{\text{def}}{=} id_{|q_i|}[a_{i+1}\mapsto b]$ and $b \in \mathbb{A}\setminus\{a_1,\ldots,a_n\}$;
- q_{i+1} has $i+1$ transitions $q_{i+1} \xrightarrow[\sigma]{\downarrow^a,\emptyset} q_i$, where $a \in \{a_1,\ldots,a_{i+1}\}$ and σ is some canonical injection from $\{a_1,\ldots,a_i\}$ to $\{a_1,\ldots,a_{i+1}\}\setminus\{a\}$.

The first type of transitions are *acquire* transitions, the other *release* ones. For example, for $n=3$ we obtain the following HDA.



$\sigma_1 = \{a_1\mapsto b\}$, $\sigma_2 = \{a_1\mapsto a_1, a_2\mapsto b\}$, $\sigma_3 = \{a_1\mapsto a_1, a_2\mapsto a_2, a_3\mapsto b\}$, $\tau_1 = \emptyset$, $\tau_2^1 = \{a_1\mapsto a_2\}$, $\tau_2^2 = \{a_1\mapsto a_1\}$, $\tau_3^1 = \{a_1\mapsto a_2, a_2\mapsto a_3\}$, $\tau_3^2 = \{a_1\mapsto a_1, a_2\mapsto a_3\}$, $\tau_3^3 = \{a_1\mapsto a_1, a_2\mapsto a_2\}$.

Given a word $w = \ell_1\ldots\ell_m$, let $\uparrow w$ denote the size of the set $\{i \mid \exists a\in\mathbb{A} : \ell_i = \uparrow^a\}$ and similarly for $\downarrow w$. The language accepted by $\mathcal{H}_n$ is:

$$\begin{aligned}
\mathcal{L} = \{\, \ell_1\ldots\ell_m \mid\ & \forall i.\ \ell_i = \downarrow^a \implies \exists j<i.\ \ell_j = \uparrow^a && (2)\\
& \wedge \forall i.\ \ell_i = \uparrow^a \implies \exists i<j.\ \ell_j = \downarrow^a && (3)\\
& \wedge \forall i<j.\ \ell_i = \ell_j = \uparrow^a \implies \exists i<h<j.\ \ell_h = \downarrow^a && (4)\\
& \wedge \forall i<j.\ \ell_i = \ell_j = \downarrow^a \implies \exists i<h<j.\ \ell_h = \uparrow^a && (5)\\
& \wedge \forall i.\ \uparrow(\ell_1\cdots\ell_i) - \downarrow(\ell_1\cdots\ell_i) \le n \,\} && (6)
\end{aligned}$$

Conditions (2) and (3) state that every release (resp. acquire) transition of process a is preceded (resp. followed) by an acquire (resp. a release) one of a. By (4), a process a cannot acquire a resource, say r, while holding another one and, by (5), a cannot release r without first acquiring it. Finally, (6) states that the total number of currently allocated resources does not exceed n.

[3] Note that the dynamically allocated entities are processes, not the resources they use.

We now define bisimilarity for HDAs in terms of configuration graphs. Two states q and q' in isolation cannot be compared, due to locality of names, but it is necessary to establish a partial correspondence between $|q|$ and $|q'|$. Therefore, we consider bisimilarity of configurations instead of states. The names of two configurations (q_1, ρ_1, H_1) and (q_2, ρ_2, H_2) are partially related by the injections ρ_1 and ρ_2.

Definition 9. *A relation $\_\mathcal{R}\_ \subseteq \mathcal{C} \times \mathcal{C}$ is a **simulation** if $(q_1, \rho_1, H_1)\mathcal{R}(q_2, \rho_2, H_2)$ implies that if q_1 is final then q_2 is final, and for each $(q_1, \rho_1, H_1) \xrightarrow{a} (q_1', \rho_1', H_1')$ there is $(q_2, \rho_2, H_2) \xrightarrow{a} (q_2', \rho_2', H_2')$ such that $(q_1', \rho_1', H_1')\mathcal{R}(q_2', \rho_2', H_2')$. If both $\mathcal{R}$ and $\mathcal{R}^{-1}$ are simulations, $\mathcal{R}$ is a **bisimulation**. Two HDAs $\mathcal{H}_1$ and $\mathcal{H}_2$ are **bisimilar**, written $\mathcal{H}_1 \sim \mathcal{H}_2$, if there is a bisimulation $\mathcal{R}$ such that $(q_{01}, id_{|q_{01}|}, \emptyset)\mathcal{R}(q_{02}, id_{|q_{02}|}, \emptyset)$, where q_{0i} is the initial state of $\mathcal{H}_i$.*

4 Expressiveness

We study the expressiveness of our machines by relating them to other classes of automata over infinite alphabets, and in particular to register automata. Since such classes of automata are typically presented with *unary* transitions (i.e., transitions labelled either with a register name or with a symbol from a finite alphabet), we restrict our attention to a (unary) alphabet Σ_{u} such that finitely many elements have empty support while the support of infinitely may others is a singleton atom. Formally, $\Sigma_{\mathsf{u}} \stackrel{\text{def}}{=} \mathbb{A} \cup \mathbb{C}$, where $\mathbb{C}$ is a finite set with $\mathbb{C} \cap \mathbb{A} = \emptyset$.

4.1 Unary HDAs and FRAs

The restriction to Σ_{u} allows to significantly simplify the definition of our automata by embedding freshness annotations within labels. Formally, we consider labels

$$\ell \in \Sigma_{\mathsf{u}} \cup \{\star, \circledast\}$$

and a transition from q is now written $q \xrightarrow[\sigma]{\ell} q'$, where $\sigma : |q'| \to |q| \cup (\{\ell\} \cap \{\star, \circledast\})$ is an injective map.

Definition 10. *A **unary history-dependent automaton (UHDA)** $\mathcal{H}$ is an HDAs on Σ_{u} with transitions defined as above. The configuration graph of a UHDA $\mathcal{H} = (Q, tr, q_0, F)$ is produced as follows. Given $(q, \rho, H) \in \mathcal{C}$ and $q \xrightarrow[\sigma]{\ell} q'$,*

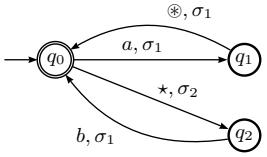
- *if $\ell \in \mathbb{C}$ then $(q, \rho, H) \xrightarrow{\ell} (q', \rho \circ \sigma, H)$;*
- *if $\ell \in |q|$ then $(q, \rho, H) \xrightarrow{\rho(\ell)} (q', \rho \circ \sigma, H \cup \{\rho(\ell)\})$;*
- *if $\ell = \star$ and $a \in \mathbb{A} \setminus Im(\rho)$ then $(q, \rho, H) \xrightarrow{a} (q', \rho[\star \mapsto a] \circ \sigma, H \cup \{a\})$;*
- *if $\ell = \circledast$ and $b \in \mathbb{A} \setminus (H \cup |q_0|)$ then $(q, \rho, H) \xrightarrow{b} (q', \rho[\circledast \mapsto b] \circ \sigma, H \cup \{b\})$.*

Example 11. Consider the program (7) below written in a higher-order CBV language with ML-style integer references [20] where the reference type is populated by names.

$$x : \mathsf{ref} \ \vdash \ \lambda y^{\mathsf{ref}}.\, \mathsf{if}\ (x == y)\ \mathsf{then}\ \mathsf{newref}\ \mathsf{else}\ y \ : \ \mathsf{ref} \to \mathsf{ref} \tag{7}$$

(7) returns a function F of type $\mathsf{ref} \to \mathsf{ref}$ which compares the name x (of reference type and known both to the program and its environment) with its input name y (of reference

type). When x equals y, F returns newref, which evaluates to a fresh name (different function calls will return different fresh names). Otherwise, it returns back y. Thus, the behaviour of (7) can be described by the following FHDA $\mathcal{H}$.



$|q_0| = |q_1| = \{a\}, \;\; |q_2| = \{a, b\}$
$\sigma_1 = (a \mapsto a), \;\; \sigma_2 = (a \mapsto a, b \mapsto \star)$

The accepted language $\mathcal{L}$ is given as follows. Setting $\mathcal{L}'(H) = \{\, ac \mid c \notin H \cup \{a\} \,\}$ and $\mathcal{L}'' = \{\, cc \mid c \neq a \,\}$, we have that $\mathcal{L} = \bigcup_i \mathcal{L}_i$ where $\mathcal{L}_0 = \{\, \epsilon \,\}$ and $\mathcal{L}_{i+1} = \{\, ww' \mid w \in \mathcal{L}_i \wedge w' \in \mathcal{L}'(Supp(w)) \cup \mathcal{L}'' \,\}$. It is shown in [20] that $\mathcal{L}$ (and hence $\mathcal{H}$) is a precise representation of the observable behaviour of our program.

We now proceed to *Fresh-Register Automata*; the following definitions are adapted from [25]. For each natural number n, let $[n] \stackrel{\text{def}}{=} \{1, \ldots, n\}$ and define

$$\mathsf{Reg}_n = \{\, \theta : [n] \rightharpoonup \mathbb{A} \mid \forall i, j \in dom(\theta).\, \theta(i) \neq \theta(j) \,\}$$

to be the set of *register assignments* of size n. Whenever $a \notin Im(\theta)$ and $i \in [n]$, we write $\theta[i \mapsto a] = \{\, (i, a) \,\} \cup \{\, (j, \theta(j)) \mid j \in [n] \setminus \{i\} \,\}$ for the update of θ at i.

Definition 12. *A **fresh-register automaton (FRA)** of n registers is a quintuple $\mathcal{A} = (Q, q_0, \theta_0, \delta, F)$ where:*

- *Q is a finite set of states, $q_0 \in Q$ is the initial one, and $F \subseteq Q$ are the final ones;*
- *$\theta_0 \in \mathsf{Reg}_n$ is the initial register assignment;*
- *$\delta \subseteq Q \times (\mathbb{C} \cup \{\, i, i^\star, i^\circledast \mid i \in [n] \,\}) \times Q$ is the transition relation.*

$\mathcal{A}$ is called a register automaton (RA) if there are no q, q', i such that $(q, i^\circledast, q') \in \delta$.

RAs are also known in the literature as Finite-Memory Automata (FMA) [14]. A *configuration* for $\mathcal{A}$ is a triple (q, θ, H) where $q \in Q$, $\theta \in \mathsf{Reg}_n$ and H is a finite subset of $\mathbb{A}$. We write $\mathcal{C}_{\mathsf{fra}}$ for the set of all configurations of this form. Given an FRA $\mathcal{A}$ as above, we define a transition relation on configurations $\rightarrow \subseteq \mathcal{C}_{\mathsf{fra}} \times \Sigma_{\mathsf{u}} \times \mathcal{C}_{\mathsf{fra}}$ as follows. For all $(q, \theta, H) \in \mathcal{C}_{\mathsf{fra}}$ and $(q, \ell, q') \in \delta$:

- if $\ell \in \mathbb{C}$ then $(q, \theta, H) \xrightarrow{\ell} (q', \theta, H)$;
- if $\ell = i \in dom(\theta)$ then $(q, \theta, H) \xrightarrow{\theta(i)} (q', \theta, H \cup \{\theta(i)\})$;
- if $\ell = i^\star$ and $a \notin Im(\theta)$ then $(q, \theta, H) \xrightarrow{a} (q', \theta[i \mapsto a], H \cup \{a\})$;
- if $\ell = i^\circledast$ and $a \notin H \cup Im(\theta_0)$ then $(q, \theta, H) \xrightarrow{a} (q', \theta[i \mapsto a], H \cup \{a\})$.

The language *accepted* by $\mathcal{A}$ is:

$$\mathcal{L}(\mathcal{A}) = \{\, w \in \Sigma_{\mathsf{u}}^* \mid (q_0, \theta_0, \emptyset) \stackrel{w}{\longrightarrow\!\!\!\!\!\rightarrow} (q, \theta, H) \wedge q \in F \,\}\,.$$

Two automata are *equivalent* if they accept the same language. Bisimilarity is defined in Definition 9, by considering bisimulations in the induced transition graphs. For RAs the constructions above are simplified by completely ignoring the H-components.

Fact 13 *RAs are closed under union, intersection, concatenation and Kleene star; they are not closed under complementation [14]. Moreover, FRAs are closed under union*

and intersection; and not closed under concatenation, Kleene star and complementation [25]. Finally, for both classes of automata, emptiness and bisimilarity are decidable [14, 25]; while universality, equivalence and containment are undecidable [21].

4.2 Correspondence

We next show that UHDAs precisely correspond to FRAs and therefore these classes of automata are equi-expressive. More precisely, given a UHDA $\mathcal{H}$ we effectively construct a bisimilar FRA $\mathcal{A}$, and viceversa. Also, these correspondences project onto UHDAs which do not use the $\circledast$ label and RAs. We thus obtain the results of Corollary 15.

The direction from FRAs to UHDAs is relatively simple: FRAs carry registers where UHDAs carry internal names. In practice the translation becomes a bit more elaborate because FRAs may have empty registers, whereas our automata are garbage-free and use all the names in their supports. The construction is relatively simple and delegated to the Appendix.

Let now $\mathcal{H} = (Q, q_0, tr, F)$ be a UHDA such that its states have at most n internal names. Moreover, let us assume a fixed ordering of $\mathbb{A}$, that is, some injective χ from $\mathbb{A}$ to natural numbers. For each finite $S \subseteq \mathbb{A}$ we write $\mathsf{ord}(S)$ for the ordering of S according to χ, i.e. for the sequence $a_1 \dots a_n \in \mathbb{A}^*$ such that $Supp\,(a_1 \dots a_n) = S$, all the a_i are distinct and, for each $i < j$, $\chi(a_i) < \chi(a_j)$. Note that the specific choice of χ is immaterial and the following results do not depend on it.

We define an FRA $\mathcal{A}$ with $n+1$ registers which (bi)simulates the behaviour of $\mathcal{H}$ as follows. During computation in $\mathcal{H}$, the internal names of its states may be re-ordered or deleted. For example, consider the transition τ_3^1 of Example 8 where no name of q_3 is mapped to the internal name a_1 of q_2, and the internal names are re-ordered. Since the registers of FRAs are fixed, and cannot be re-ordered nor deleted, in order to simulate this behaviour an external component needs to be added to the states. This will be a function $f : [n] \to [n+1]$ which will represent re-orderings by mapping elements of $[n]$ to elements of $[n]$, and will simulate deletions by mapping elements of $[n]$ to $n+1$. This means that the "deleted" names will remain in the registers, but the information in f will allow us to know that these names should not be there. Therefore, if we want to simulate a locally fresh transition and know that the i-th register, say, contains a deleted name then we will explicitly add a transition for this register; otherwise, $\mathcal{A}'$ would not be able to match the deleted name. Finally, note that the reason we need $n+1$ registers is that a state q may contain n local names already and $\mathcal{H}$ may still have a transition, say, $q \xrightarrow[id_{|q|}]{\star} q$. In $\mathcal{A}$ the fresh name will be put in the $n+1$-th register.

Concretely, we define the FRA $\mathcal{A} = (Q', q_0', \theta_0', \delta', F')$ with $n+1$ registers, where

$$Q' = Q \times \{\, f : [n] \to [n+1] \mid \forall i, j.\, f(i) = f(j) \neq n+1 \implies i = j \,\},$$

Q is the set underlying the named set $(Q, \mathtt{G}_Q)$ and, assuming $\mathsf{ord}(|q_0|) = a_{01} \dots a_{0m_0}$:

- $q_0' = (q_0, f_0)$, where $f_0 = \{\, (i,i) \mid 1 \leq i \leq m_0 \,\} \cup \{(i, n+1) \mid m_0 < i \leq n+1 \,\}$.
- θ_0' maps each $1 \leq i \leq m_0$ to a_{0i} and each $i > m_0$ to $\#$.
- $F' = \{(q, f) \in Q' \mid q \in F\}$.
- δ' is given as follows. Assume below that $\mathsf{ord}(|q|) = a_1 \dots a_m$ and $\mathsf{ord}(|q'|) = a_1' \dots a_{m'}'$. If $(\ell, \sigma, q') \in tr(q)$ then $((q, f), \ell', (q', f')) \in \delta'$, with ℓ', f' given by:

- $f' = (\bar{f} \circ \sigma')[i_{\mathsf{fr}} \mapsto j_{\mathsf{fr}}]$, where $\sigma' : [n] \to [n+1]$ is defined as: $\sigma'(i) = n+1$ (if $i > m'$ or $\sigma(a'_i) \in \{\star, \circledast\}$) and $\sigma'(i) = j$ (if $\sigma(a'_i) = a_j$), and $\bar{f} = f[n+1 \mapsto n+1]$. Moreover, i_{fr} is the index such that $\sigma(a'_{i_{\mathsf{fr}}}) \in \{\star, \circledast\}$, and j_{fr} is the least index such that $j_{\mathsf{fr}} \notin (f \circ \sigma')([n+1])$; in case no such i_{fr} exists, i.e. $\sigma(|q'|) \cap \{\star, \circledast\} = \emptyset$, the notation means $f' = \bar{f} \circ \sigma'$.
- If $\ell \in \mathbb{C}$ then $\ell' = \ell$.
- If $\ell \in |q|$ then $\ell' = f(i)$, where $\ell = a_i$.
- If $\ell = \star/\circledast$ then $\ell' = j^{\star}/j^{\circledast}$, where $j = n+1$ if $\ell \notin \sigma(|q'|)$, and $j = f'(i)$ if $\sigma(a'_i) = \ell$.

Moreover, if $(\star, \sigma, q') \in tr(q)$ then $((q, f), j_{\mathsf{fr}}, (q', f')) \in \delta'$, for each $j_{\mathsf{fr}} \in f^{-1}(n+1)$ and $f' = (\bar{f} \circ \sigma')[i_{\mathsf{fr}} \mapsto j_{\mathsf{fr}}]$, with $i_{\mathsf{fr}}, \bar{f}, \sigma'$ given as above.

We can now establish the following result (proof in the Appendix).

Proposition 14. *For $\mathcal{H}$ and $\mathcal{A}$ as above, $\mathcal{H} \sim \mathcal{A}$.*

We have thus shown the equivalence between UHDAs and FRAs. By examining the precise constructions it is easy to see that UHDAs which do not use globally fresh transitions are equivalent to RAs in the same sense. We therefore obtain the following.

Corollary 15.
- *UHDAs are closed under union and intersection; they are not closed under concatenation, Kleene star and complementation.*
- *UHDAs which do not use the $\circledast$ label are closed under union, intersection, concatenation and Kleene star; they are not closed under complementation.*

For both classes of automata, emptiness and bisimilarity are decidable; universality, equivalence and containment are undecidable.

A consequence of the above is that global freshness adds expressive power to our automata. For instance, the language $\mathcal{L}$ of the Example 11 cannot be accepted without $\circledast$-transitions, as it is not RA-accepted [14, Prop. 3].

5 Symmetries and minimal models

A primary question when dealing with models is the existence of computable *canonical* ones. In language theory, a procedure for computing a unique canonical representative of each class of language-equivalent automata allows one to decide language equivalence. We discuss here how to compute canonical models for subclasses of HDAs. In fact, Corollary 15 implies that language equivalence is not decidable for HDAs. Therefore, minimal models can be computed for proper subclasses of HDAs, in particular, we use a minimisation procedure for HDAs. Such procedure is based on a notion of partition refinement for HDAs which hinges on symmetries [22, 10]. This is the main reason for the inclusion of symmetries in Definition 2 (see Appendix D).

Recall the definition of HDA bisimilarity (Definition 9). It is easy to see that bisimilarity implies language equivalence. A more interesting question is to find classes of HDAs where language equivalence implies bisimilarity. One such class is easily identified in the *deterministic HDAs*, defined as the ones whose configuration graph is deterministic. A larger class is given in the following definition.

Definition 16. *Call an HDA $\mathcal{H} = (Q, tr, q_0, F)$* ***symmetrised-deterministic*** *(sdHDA) iff. for each $q \in Q$, at least a final state $q' \in F$ is reachable from q, and for any two*

transitions $q \xrightarrow[\sigma_1]{\ell_1, f_1} q_1$ *and* $q \xrightarrow[\sigma_2]{\ell_2, f_2} q_2$ *having the same source, whenever there is a permutation* ι *such that* $\iota \cdot \ell_1 = \ell_2$*, and* ι *is the identity on* $|q|$*, then* $q_1 = q_2$*, and there is* $\pi \in \mathtt{G}(q_1)$ *such that* $\iota \circ \sigma_1 \circ \pi = \sigma_2$*.*

In an sdHDA, pairs of transitions that may accept the same symbol must lead to the same state, modulo a permutation in its symmetry. Such pairs of transitions are characterised by the existence of an assignment of fresh names to fresh names ι that makes two labels equal. Lemma 17 below demonstrates the purpose of Definition 16 and allows us to show the subsequent proposition (proofs in the Appendix).

Lemma 17. *For all configurations* (q, ρ, H) *of an sdHDA, whenever there are two transitions with the same sources and labels in the configuration graph, say* $(q, \rho, H) \xrightarrow{a} (q_1, \rho_1, H')$ *and* $(q, \rho, H) \xrightarrow{a} (q_2, \rho_2, H')$*, then* $\mathcal{L}(q_1, \rho_1, H') = \mathcal{L}(q_2, \rho_2, H')$*.*

Proposition 18. *On sdHDAs, language equivalence and bisimilarity coincide.*

The *minimal automaton* of an sdHDA is its bisimilarity quotient. Minimisation is performed as in [10], with some extra care taken in order to cater for the inter-matching of globally and locally fresh transitions, as in [25]. Interestingly, the algorithm also computes the largest symmetry that preserves the accepted language. The automated computation of such groups is paramount for efficient verification (see e.g. [8]).

Example 19. Let us construct the minimal automaton (Q, tr, q_0, F) for the HDA $\mathcal{H}_n$ of Example 8. We set $Q = \{q_0, \ldots, q_n\}$ and $F = \{q_0\}$. The symmetry of each state q_i is the *symmetric* group over $\{a_1, \ldots, a_i\}$, that is, it contains all the permutations of this set. The transitions are as in the case without symmetry but, due to condition (1) of Definition 4, only one release transition from each q_{i+1} to q_i needs to be represented. The remaining transitions are obtained by closure wrt the symmetry of states.

Finally, we remark that symmetries have compact representations. This is due to a theorem of group theory: a finite permutation group G of cardinality $|G|$ can be represented using a set of *generators* [3] of size smaller than $\log_2 |G|$ (see e.g. [7], §3.3), whose closure wrt composition is G. For instance, the symmetric group over $\{a_1, \ldots, a_i\}$ used in Example 19 can be represented using a transposition of two arbitrary elements and a *round shift* of all the elements. Efficient algorithms can be defined e.g. for permutation search, which work directly on the generators [16]. These can be reused in implementations of analysis tools based on HDAs as language acceptors.

6 Concluding remarks

We have studied a class of automata over infinite alphabets, whose main feature is the presence of names and name allocation. We regard this work as a foundational starting point for a theory of resource-sensitive language models of computations taking into account compositionality and existence of canonical models. Our investigation aims to use these models for applications such as trace-analysis and runtime monitoring of systems that may have to deal with infinite sets of resources. Therefore, we plan to investigate efficient recognition procedures and language-theoretic operations (e.g. intersection of automata). Such procedures would make use of the compact representation of a symmetry by its generators, and of efficient algorithms for permutation groups [16].

A valuable application of the framework is the possibility to use process calculi such as the CCS or the π-calculus to *specify* HDAs. Combining our results with process calculi mapped on HDAs (e.g. [22, 10]) would make it possible to use typical process algebraic operators to specify languages (e.g. to specify monitors, trace-analyses and parsers) for systems with infinite alphabets.

References

1. M. Bojanczyk. Data monoids. In *Proceedings of STACS*, pp. 105-116, 2011.
2. M. Bojanczyk, B. Klin, and S. Lasota. Automata with group actions. In *Proceedings of LICS*, pp. 355-364, 2011.
3. C. C. Sims. Computational methods in the study of permutation groups. In John Leech, editor, *Computational Problems in Abstract Algebra*. Pergamon (Oxford), 1970.
4. V. Ciancia. *Accessible Functors and Final Coalgebras for Named Sets*. PhD thesis, Dip. di Informatica, Pisa, 2008.
5. V. Ciancia, A. Kurz, and U. Montanari. Families of symmetries as efficient models of resource binding. *Electr. Notes Theor. Comput. Sci.*, 264(2):63–81, 2010.
6. V. Ciancia and U. Montanari. Symmetries, local names and dynamic (de)-allocation of names. *Inf. Comput.*, 208:1349–1367, 2010.
7. J. D. Dixon and B. Mortimer. *Permutation Groups*, vol. 163 of *GTM*. Springer, 1996.
8. E. Emerson and S. Sistla. Symmetry and model checking. In *Formal Methods in System Design*, vol. 9, n. 1-2, pp. 105-131, 1994.
9. G. Ferrari, U. Montanari, and M. Pistore. Minimizing Transition Systems for Name Passing Calculi: A Co-algebraic Formulation. In *Proceedings of FOSSACS*, pp. 129–143, 2002.
10. 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.
11. M. Gabbay and A. Pitts. A new approach to abstract syntax involving binders. In *Proceedings of LICS*, pp. 214–224, 1999.
12. M. J. Gabbay and V. Ciancia. Freshness and name-restriction in sets of traces with names. In *Proceedings of FOSSACS*, vol. 6604 of *LNCS*, pp. 365–380, 2011.
13. 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.
14. N. Kaminski, M. Francez. Finite-memory automata. *TCS*, 134(2):329–363, 94.
15. A. Kurz, T. Suzuki, and E. Tuosto. Towards nominal formal languages. *CoRR*, abs/1102.3174, 2011.
16. E. M. Luks. Permutation Groups and Polynomial Time Computation. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 11:139–175, 1993.
17. 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.
18. U. Montanari and M. Pistore. History-dependent automata. Technical Report TR-98-11, Dip. di Informatica, Pisa, 1998.
19. U. Montanari and M. Pistore. π-Calculus, Structured Coalgebras, and Minimal HD-Automata. In *Proceedings of MFCS*, vol. 1983 of *LNCS*, 2000.
20. A. S. Murawski and N. Tzevelekos. Algorithmic nominal game semantics. In *ESOP*, 2011.
21. F. Neven, T. Schwentick, and V. Vianu. Finite state machines for strings over infinite alphabets. *ACM Trans. Comput. Logic*, 5(3):403–435, 2004.
22. M. Pistore. *History Dependent Automata*. PhD thesis, Dip. di Informatica, Pisa, 1999.
23. S. Staton. *Name-passing process calculi: operational models and structural operational semantics*. PhD thesis, University of Cambridge, 2007.
24. E. Tuosto. *Non-Functional Aspects of Wide Area Network Programming*. PhD thesis, Dip. di Informatica, Pisa, 2003.
25. N. Tzevelekos. Fresh-register automata. In *Proceedings of POPL*, pp. 295-306, 2011.

A Examples of nominal sets

Every set X is a nominal set with trivial action, $\pi \cdot x = x$, in which case every $x \in X$ has empty support. More interestingly:

– The set $\mathbb{A}$ is a nominal set under the "natural" action $\pi \cdot a = \pi(a)$, each $\pi \in Perm$ and $a \in \mathbb{A}$; observe that, of each $a \in \mathbb{A}$, $Supp(a) = \{a\}$.

– The set $\mathbb{A}^*$ of finite words of names is a nominal set under the action $\pi \cdot a_1 \cdots a_n = \pi(a_1) \cdots \pi(a_n)$; in this case, $Supp(a_1 \cdots a_n) = \{a_1, \ldots, a_n\}$.

– The set $\mathcal{P}_{\mathsf{fn}}(\mathbb{A})$ of finite sets of names is a nominal set under the action $\pi \cdot S = \{\pi(a) \mid a \in S\}$, in which case $Supp(S) = S$.

Moreover, disjoint unions, products, etc. of nominal sets form nominal sets by defining the permutation action componentwise. Note the intrinsic symmetry of elements in the latter case above: for example, taking $S = \{a, b\}$ we have $(a\ b) \cdot S = S$, even if $(a\ b)$ actually fixes neither a nor b. Although a and b are in $Supp(S)$, they are interchangeable inside S. They are two distinct names whose *local meaning* is the same.

B Proofs

Proof (Proposition 14). We say that an FRA-configuration is *reachable* if it can be reached from the initial configuration. We show that the following relation is a bisimulation.

$$R = \{\, ((q,\rho,H),((q,f),\theta,H)) \mid ((q,f),\theta,H) \text{ reachable } \wedge\, \mathsf{ord}(|q|) = a_1 \ldots a_m \\ \forall i.\, (1 \leq i \leq m \implies \theta(f(i)) = \rho(a_i)) \wedge (m < i \leq n \implies f(i) = n+1) \,\}$$

Suppose that $((q,\rho,H),((q,f),\theta,H)) \in R$ and $(q,\rho,H) \xrightarrow{\ell} (q',\rho',H')$. Recall that $\mathsf{ord}(|q|) = a_1 \ldots a_m$ and say that $\mathsf{ord}(|q'|) = a'_1 \ldots a'_{m'}$. We do case analysis on the underlying transition, focussing on $\ell \in \mathbb{A}$; the case of $\ell \in \mathbb{C}$ is similar and simpler.

If the transition is some $(q', a_k, \sigma) \in tr(q)$ then $\ell = \rho(a_k)$ and $((q,f), f(k), (q', f')) \in \delta'$ with $f' = f \circ \sigma'$. Thus, $((q,f),\theta,H) \xrightarrow{\theta(f(k))} ((q',f'),\theta,H')$ where $\theta(f(k)) = \rho(a_k) = \ell$ by hypothesis. We still need to show that $((q',\rho',H'),((q',f'),\theta,H')) \in R$, that is, that the stated conditions are satisfied. The configuration is clearly reachable. Moreover, for each $1 \leq i \leq m'$ we have $\rho'(a'_i) = \rho(\sigma(a'_i)) = \rho(a_j) = \theta(f(j)) = \theta(f(\sigma'(i))) = \theta(f'(i))$, where $\sigma(a'_i) = a_j$ (in particular, $j = \sigma'(i) \leq n$). Finally, for each $m' < i \leq n$ we have $f'(i) = \bar{f} \circ \sigma'(i) = \bar{f}(n+1) = n+1$, as required.

If the transition is some $(q', \star, \sigma) \in tr(q)$ and $\ell \notin Im(\theta)$ then $((q,f), k^\star, (q', f')) \in \delta'$, and thus $((q,f),\theta,H) \xrightarrow{\ell} ((q',f'),\theta',H')$ with $\theta' = \theta[k \mapsto \ell]$. If $\star \notin \sigma(|q'|)$ then $k = n+1$ and $f' = \bar{f} \circ \sigma'$. Moreover, for each $1 \leq i \leq m'$ we have $\theta'(f'(i)) = \rho'(a'_i)$ as above, noting that $\sigma(a'_i) = a_j$ with $j \leq n$, and for each $m' < i \leq n$ we have $f'(i) = n+1$. On the other hand, if $\sigma(a'_{i_{\mathsf{fr}}}) = \star$ then $k = f'(i_{\mathsf{fr}})$, i.e. $k = j_{\mathsf{fr}}$, the minimum index such that $j_{\mathsf{fr}} \notin \bar{f} \circ \sigma'([n])$. Note that σ' maps i_{fr} to $n+1$, thus $j_{\mathsf{fr}} < n+1$. We have that $\theta'(f'(i_{\mathsf{fr}})) = \theta'(j_{\mathsf{fr}}) = \ell$ and $\rho'(a'_{i_{\mathsf{fr}}}) = \rho[\star \mapsto \ell] \circ \sigma(a_{i_{\mathsf{fr}}}) = \ell$. The rest of the conditions are shown as above. Thus, in both cases we get $((q',\rho',H'),((q',f'),\theta',H')) \in R$.

If the transition is some $(q', \circledast, \sigma) \in tr(q)$ then work as above, using also the fact that $((q,f),\theta,H)$ is reachable.

If the transition is some $(q', \star, \sigma) \in tr(q)$ and $\ell = \theta(k)$ then, since $\ell \notin \rho|q|$, $k > m$ and by hypothesis $f(k) = n+1$. Thus, $((q,f), j_{\mathsf{fr}}, (q', f')) \in \delta'$ with $j_{\mathsf{fr}} = k$ and

therefore $((q,f),\theta,H) \xrightarrow{\ell} ((q',f',\theta,H')$. If $\star \notin \sigma(|q'|)$ then work as above. Otherwise, if $\star = \sigma(a'_{i_{\mathsf{fr}}})$ then $f' = (\bar{f} \circ \sigma')[i_{\mathsf{fr}} \mapsto j_{\mathsf{fr}}]$ and $\theta(f'(i_{\mathsf{fr}})) = \theta(j_{\mathsf{fr}}) = \ell$ while $\rho'(a'_{i_{\mathsf{fr}}}) = (\rho[\star \mapsto \ell] \circ \sigma)(a'_{i_{\mathsf{fr}}}) = \ell$. The rest of this case is as above, and thus $((q',\rho',H'),((q',f'),\theta,H')) \in R$.

The above argument shows that R is a simulation. Now suppose $((q,f),\theta,H) \xrightarrow{\ell} ((q',f'),\theta',H')$ with $\ell \in \mathbb{A}$ and $\mathsf{ord}(|q'|) = a'_1 \dots a'_{m'}$. We do a case analysis on the underlying transition.

If the transition is some $((q,f),j,(q',f')) \in \delta'$ and $\ell = \theta(j) = \rho(a_i)$ then, by hypothesis, $j = f(i)$ and thus $j \notin f^{-1}(n+1)$. Thus, $((q,f),j,(q',f')) \in \delta'$ is due to some $(a_i,\sigma,q') \in tr(q)$ and thus $(q,\rho,H) \xrightarrow{\ell} (q',\rho',H')$. We can now check as above that the conditions are satisfied and therefore $((q',\rho',H'),((q',f'),\theta',H')) \in R$.

If the transition is some $((q,f),j,(q',f')) \in \delta'$ and $\ell = \theta(j) \notin \rho(|q|)$ then $f(j) = n+1$, and $((q,f),j,(q',f')) \in \delta'$ is due to some $(\star,\sigma,q') \in tr(q)$. We therefore obtain $(q,\rho,H) \xrightarrow{\ell} (q',\rho',H')$ and we can check that $((q',\rho',H'),((q',f'),\theta',H')) \in R$.

The other two cases involve the transitions containing labels of the form $j^\star/j^{\circledast}$, which have one corresponding $\mathcal{A}'$-transition each. These cases are resolved in a similar manner to the above. □

Proof (Lemma 17). By injectivity of ρ and definition of configuration graph, under the hypothesis of Lemma 17, it is not difficult to see that the symbol a is accepted by two transitions in the form $q \xrightarrow[\sigma_1]{\ell_1,f_1} q_1$, and $q \xrightarrow[\sigma_2]{\ell_2,f_2} q_2$ such that there is ι as in Definition 16 satisfying $\iota(\ell_1) = \ell_2$. By definition of sdHDA, there is $\pi \in \mathtt{G}q_1$ such that $\iota \circ \sigma_1 \circ \pi = \sigma_2$. By Prop. 6, $\mathcal{L}(q_1,\pi,H) = \mathcal{L}(q_1,id,H)$. Then by how ρ_1 and ρ_2 are obtained from σ_1 and σ_2 (see Definition 5), and Prop. 6 again we get the thesis. □

Proof (Proposition 18). We show that language equivalence is a bisimulation. Consider two configurations c_1 and c_2 accepting the same language. Consider a transition $c_1 \xrightarrow{a} c'_1$. Observe that by Lemma 17, for all the transitions $c_1 \xrightarrow{a} c''_1$ we have $\mathcal{L}(c'_1) = \mathcal{L}(c''_1)$. We have to prove that there is a transition $c_2 \xrightarrow{a} c'_2$ such that $\mathcal{L}(c'_1) = \mathcal{L}(c'_2)$. By language equivalence of c_1 and c_2, and the first condition in Definition 16, there certainly is at least a transition $c_2 \xrightarrow{a} c'_2$, and by Lemma 17 all such transitions go into configurations that accept the same language. Reasoning on the possible derivatives, we have that $\mathcal{L}(c'_1) = \{w \mid aw \in \mathcal{L}(c_1)\}$ and similarly $\mathcal{L}(c'_2) = \{w \mid aw \in \mathcal{L}(c_2)\}$. By language equivalence again, we have the thesis. □

C From FRAs to UHDAs

Let us assume $\mathcal{A} = (Q,q_0,\theta_0,\delta,F)$ is an FRA of n registers, and let us fix $b_1,\dots,b_n$ to be distinct names such that $dom(\theta_0) \subseteq \{b_1,\dots,b_n\}$. Following [25], we effectively construct a bisimilar FRA $\overline{\mathcal{A}}$ of n registers where each state is decorated with a subset of $[n]$ specifying its non-empty registers. The automaton has the same transitions as $\mathcal{A}$, but for *blocking* transitions, that is, transitions of the form $((q,S),i,(q',S'))$ where $i \notin S$: such transitions are not allowed in $\overline{\mathcal{A}}$. Concretely, $\overline{\mathcal{A}} = (\overline{Q},\overline{q_0},\theta_0,\overline{\delta},\overline{F})$ where

$\overline{Q} = Q \times \mathcal{P}([n])$, $\overline{q_0} = (q_0, dom(\theta_0))$, $\overline{F} = \{\, (q,S) \mid q \in F \,\}$ and:

$$\begin{aligned}\overline{\delta} = \{\, ((q,S), \ell, (q',S)) \mid (q,\ell,q') \in \delta \wedge (\ell \in \mathbb{C} \vee \ell \in S) \,\} \\ \cup \{\, ((q,S), i^\star/i^\circledast, (q',S')) \mid (q, i^\star/i^\circledast, q') \in \delta \wedge S' = S \cup \{i\} \,\}\end{aligned}$$

As in [25, Section 3], we can show that $\mathcal{A} \sim \overline{\mathcal{A}}$. We now construct the FHDA $\mathcal{H} = (Q', tr', q_0', F')$ as follows.

- $Q' = \{\, (\overline{Q}, \mathtt{G}) \mid \mathtt{G}(q, \{i_1, \ldots, i_m\}) = id_{\{b_{i_1}, \ldots, b_{i_m}\}} \,\}$. In particular, each (q,S) has local names $|(q,S)|_{Q'} = \{b_i \mid i \in S\}$.
- $q_0' = (q_0, dom(\theta_0))$, $F' = \{\, (q,S) \mid q \in F \,\}$, and tr' is given by:
  - if $((q,S), \ell, (q',S)) \in \overline{\delta}$ and $\ell \in \mathbb{C}$ then $((q',S), \ell, id_{|(q,S)|}) \in tr'(q,S)$;
  - if $((q,S), i, (q',S)) \in \overline{\delta}$ then $((q',S), b_i, id_{|(q,S)|}) \in tr'(q,S)$;
  - if $((q,S), i^\star/i^\circledast, (q',S')) \in \overline{\delta}$ then $((q',S'), \star/\circledast, id_{|(q,S)|}[i \mapsto \star/\circledast]) \in tr'(q,S)$.

In effect, $\mathcal{H}$ works exactly as $\overline{\mathcal{A}}$, only using b_i where $\overline{\mathcal{A}}$ would use i, and $[i \mapsto \star/\circledast]$ where $\overline{\mathcal{A}}$ would use $i^{\star/\circledast}$. It is thus straightforward to show the following.

Proposition 20. *Let $\mathcal{A}$, $\overline{\mathcal{A}}$, and $\mathcal{H}$ be as above. The following relation is a bisimulation and hence $\mathcal{A} \sim \overline{\mathcal{A}} \sim \mathcal{H}$.*

$$\mathcal{R} = \{\, (((q,S), \theta, H), ((q,S), \rho, H)) \mid dom(\theta) = S \wedge \forall i \in S.\, \theta(i) = \rho(b_i) \,\}$$

D Symmetries and canonical models in HDAs

The following example is adapted from [22], § 6.6.2 and shows that symmetries are needed for canonical models. Consider two HDAs

$$\mathcal{H}_1 = (\{q\}, tr_1, q, \{q\}) \qquad \text{and} \qquad \mathcal{H}_2 = (\{q\}, tr_2, q, \{q\})$$

such that $|q| = \{a, b\}$, the symmetry of q is the singleton $\{id_{|q|}\}$ for both $\mathcal{H}_1$ and $\mathcal{H}_2$, and

$$tr_1(q) = \{(q, a, id_{|q|}, \emptyset), (q, b, id_{|q|}, \emptyset)\}, \quad tr_2(q) = \{(q, a, (a\ b), \emptyset), (q, b, (a\ b), \emptyset)\}$$

Note that $\mathcal{H}_1$ and $\mathcal{H}_2$ differ only for the symmetries of the name mappings on their transitions; in fact, they accept the same language.

The canonical representative of $\mathcal{H}_1$ and $\mathcal{H}_2$ is the automaton $\mathcal{H} = (\{q'\}, tr', q', \{q'\})$, with $\mathtt{G}(q') = \{id_{|q|}, (a\ b)\}$, and $tr'(q')$ containing the same transitions as $tr_1(q) \cup tr_2(q)$, with q' in place of q.