Abstract

What is a basic automata-theoretic model of computation with names and fresh-name generation? We introduce Fresh-Register Automata (FRA), a new class of automata which operate on an infinite alphabet of names and use a finite number of registers to store fresh names, and to compare incoming names with previously stored ones. These finite machines extend Kaminski and Francez’s Finite-Memory Automata by being able to recognise globally fresh inputs, that is, names fresh in the whole current run. We examine the expressivity of FRA’s both from the aspect of accepted languages and of bisimulation equivalence. We establish primary properties and connections between automata of this kind, and answer key decidability questions. As a demonstrating example, we express the theory of the π-calculus in FRA’s and characterise bisimulation equivalence by an appropriate, and decidable in the finitary case, notion in these automata.

Categories and Subject Descriptors F.1.1 [Computation by Abstract Devices]: Models of Computation; D.3.1 [Programming Languages]: Formal Definitions and Theory—Semantics

General Terms Theory, Languages, Verification

1. Introduction

One of the most common and useful abstractions in programming is the assumption that entities of specific kinds can be created at will and, moreover, in such a manner that newly created entities are always fresh — distinct from any other such created thus far. This is, for example, the case with mutable reference cells, exceptions user-declared datatypes, etc. in languages like Standard ML [15]. Following a long tradition in computer science [20], we call these entities names and specify them as follows.

Names can be created fresh dynamically and locally, compared for equality and communicated between agents or subroutines.

Apart from the uses mentioned above, names form the basis of calculi of mobile processes (e.g. the π-calculus [14]); appear in network protocols and secure transactions; and are generally essential in programming for identifying variables, channels, threads, objects, codes, and many other sorts of name in disguise. To our knowledge, there has been in the literature a proposal of a basic automata-theoretic model of names, providing abstract machines underlying all these paradigms. We propose just such a model here.

Our model is based on the successful paradigm of Finite-Memory Automata (FMA), introduced by Kaminski and Francez in the early 90’s [11]. Motivated by real-world problems (where codes, addresses, identifiers, etc. may have unbounded domains), those automata address a demand for a “natural” finite-state machine model over infinite alphabets. An FMA is an automaton attached with a finite number of name-storing registers. Its structure looks identical to that of an ordinary finite-state automaton over a finite set of labels generated by indices in the range 1, . . . , n, where n is the number of registers. However, A truly operates on the infinite set of inputs ℱ (the set of names), with indices i referring to the names stored in the i-th register of A. This simple idea lifts the automaton from finite to infinite alphabet.

There are two ways in which an FMA can access its registers: either by comparing an input name to a stored one, or by storing an input name in one of its registers but only in case it is locally fresh, that is, it does not already appear in any of them. Thus, FMA’s are history-free: their computational steps rely solely on their current registers. Here we introduce Fresh-Register Automata (FRA), a finite-register automaton model which extends FMA’s by global freshness recognition: an automaton can now accept (and store) an input name just in case it is fresh in the whole run. For example, a transition

\[ q \xrightarrow{a} q' \]

means that if \( A \) is at state \( q \) and the set of names that have appeared in its registers so far is \( H \), then \( A \) can accept any name \( a \notin H \), store it in its i-th register and proceed to \( q' \). This history-sensitive feature precisely captures fresh-name creation.\(^1\) Thus, e.g. the following language (not recognised by FMA’s [11]) is recognised by a single-state FRA with one register.

\[ L_1 = \{ a_1 \cdots a_k \in L \mid \forall i \neq j, a_i \neq a_j \} \]

An intuitive way to view \( L_1 \) is as the trace of a fresh-name generator; one which returns reference cells in SML, objects in Java, memory addresses in C, etc. Research in FMA’s and their formal languages has been extensive [2, 6, 11, 21, 25, 27]. It has been shown [11, 21] that FMA-recognisable languages are closed under union, intersection, concatenation and Kleene star; they are not closed under complement and, emptiness of FMA’s is decidable; and universality is undecidable. Our first contribution is to answer this series of questions for FRA’s. We show that for emptiness and universality the situation remains the same as in FMA’s. On the other hand, FRA-recognisable languages are still closed under union and intersection, but history-sensitiveness prohibits this for concatenation and Kleene star. Moreover, they are not closed under complement and, in fact, there is an FMA-recognisable language whose complement is not recognised by FMA’s.

\(^1\)Note that, although history-sensitive, the automaton does not have full access to the history \( H \). In automata-theoretic jargon, the situation can be described as consulting an oracle who can decide the freshness of names.


Fresh-Register Automata

Nikos Tzevelekos
Oxford University Computing Laboratory
nikt@comlab.ox.ac.uk
Our main vehicle for studying equivalence between FRA’s is bisimulation equivalence (also called bisimilarity). The notion is very relevant from the point of view of programming, and process calculi in particular, and in the case of FRA’s it implies language equivalence. More importantly, we show that by examining FRA’s at the symbolic level, i.e. as ordinary finite-state automata on the set of index-generated labels, it is possible to capture bisimilarity by an appropriate symbolic notion: we thus prove that FRA-bisimilarity is decidable. A symbolic bisimulation relates states of two automata in specific environments, the latter specifying how are the names which appear in their registers related.

As a demonstrating example, we express the \( \pi \)-calculus in the context of fresh-register automata. We introduce the \( \pi \)-calculus system: a presentation of the \( \pi \)-calculus with early transition semantics [14, 26], in which processes are states of an infinite FRA. Transitions are given by FRA-transitions and the system is finitely branching. More specifically, bound outputs are modelled by globally fresh transitions, while each input is decomposed into finitely many cases: either the incoming name is locally fresh or it already appears in the registers. This clean treatment of fresh and bound names is the main advantage of the \( \pi \)-calculus and allows for the finite representation, as ordinary FRA’s, of finitary processes. Moreover, we characterise strong bisimilarity by an appropriate symbolic notion in \( \pi \). This gives an alternative proof of decidability of bisimilarity for finitary processes.

Motivation and related work

**Programming languages** The idea of studying names in higher-order languages and in isolation of other effects was first pursued by Pitts and Stark [24]. They introduced the \( \nu \)-calculus, an extension of the simply-typed \( \lambda \)-calculus with references of unit type. Investigations on the \( \nu \)-calculus were meticulously carried on by Stark in his PhD thesis [28], which exposed a rather unexpected complexity hidden behind names. It became evident that better models for languages with names were needed. To address this, new directions in denotational [1, 12, 13, 18] and operational [3, 10] models were explored, significantly advancing our understanding of computation with names but, at the same time, leaving basic questions unanswered. In particular, those works examined computation at the higher level, that of programs and program equivalence, leaving open the question of a basic, lower-level model.

Interestingly, in their initial paper on FMA’s [11], Kaminski and Francez motivate their construction (also) by briefly presenting an idealised procedural language with names. There, names cannot be freshly created, but they can be read from the environment as inputs and stored in a finite memory. Moreover, stored names can flow inside the memory from one register to another and can also be compared for equality and thus trigger goto’s. The authors explain that FMA’s operate like acceptors for that simple imperative language with names. By analogy, FRA’s describe the extension of names, and \( \pi \)-calculus in the setting of FRA’s.

**Process calculi** For mobile systems like the \( \pi \)-calculus [14], where processes can create locally, receive or send names, the use of ordinary labelled transition systems for its semantics is in many ways unsatisfactory: for example, infinite branching arises even in the case of very simple processes that receive a (locally fresh) name, or output a locally created (globally fresh) one. Such shortcomings naturally led to solutions involving representations of processes by formalisms which incorporate name-reasoning of some sort [4, 5, 16]. The most notable paradigm in this direction is that of History-Dependent Automata (HD-Automata) [16, 22], which are structures defined in a universe of named sets and named functions. HD-automata can succinctly represent the \( \pi \)-calculus, as HD-transitions match ‘on-the-fly’ names between the source, target, and label of \( \pi \)-calculus transitions, allowing thus for the use of representatives of processes and transitions, rather than all possible ones under e.g. permutation of fresh names. The stream of research on HD-automata has focussed both on foundational issues [17, 22] and on pragmatic applications [7]. The work presented here shares objectives with HD-automata, and to some extent can be viewed as a complementary attempt to the same question, albeit based on basic machines of ‘first principles’.

Outline

In the next section we give the basic definitions on FRA’s. Section 3 provides some useful bisimilar constructions. In Section 4 we recall FMA’s and establish their connection to FRA’s. We examine WFRA’s, a weaker notion of FRA’s focussing on global freshness, in Section 5. In Section 6 we prove some technical results regarding closure properties for FRA’s, and in Section 7 we show that emptiness and bisimilarity are decidable using symbolic methods. Section 8 examines the \( \pi \)-calculus in the setting of FRA’s.

2. Definitions

We distinguish between two sets of input symbols:

- an infinite set of names, \( \mathcal{A} \), and
- a finite set of constants, \( \mathbb{C} \).

Constants have an auxiliary role and are non-storable.\(^3\) We let \( a, b, \ldots \) range over names. We write \( \mathcal{A}^{*} \) for the set of finite strings of names, and \( \mathcal{A}^{\omega} \) for its restriction to those containing pairwise distinct names. Strings \( a_{1} \cdots a_{n} \) will be typically represented by vectors \( \bar{a} \), in which case \( \text{img}(\bar{a}) = \{a_{1}, \ldots, a_{n}\} \).

For each \( n \in \omega \), we write \( [n] \) for the set \( \{1, \ldots, n\} \), and let \( \mathbb{L}_{n} = \mathbb{C} \cup \{i, \bar{a}^{*}, \bar{a}^{\omega} \mid i \in [n]\} \), be the set of labels generated by \([n]\). Moreover, we define

\[
\text{Reg}_{n} = \{ \sigma : [n] \rightarrow \mathcal{A}^{\omega} \mid \forall i \neq j. \sigma(i) = \sigma(j) \implies \sigma(i) = \emptyset \}
\]

\( \text{Reg}_{\infty} \) to be the set of register assignments of size \( n \). We write \( \text{img}(\sigma) \) for the name-range of \( \sigma \), i.e. \( \text{img}(\sigma) = \{a \in \mathcal{A} \mid \exists i. \sigma(i) = a \} \), and let \( \text{dom}(\sigma) = \{i \in [n] \mid \sigma(i) \in \mathcal{A} \} \). Whenever \( a \notin \text{img}(\sigma) \), \( \sigma[i \mapsto a] = \{ (i, a) \cup \{(j, \sigma(j)) \mid j \in [n] \setminus \{i\}\} \}
\)

is an update of \( \sigma \), for any \( i \in [n] \).

**Definition 1.** A fresh-register automaton (FRA) of \( n \) registers is a quintuple \( \mathcal{A} = (Q, q_{0}, \sigma_{0}, \delta, F) \), where:

- \( Q \) is a finite set of states,
- \( q_{0} \) is the initial state,
- \( \sigma_{0} \in \text{Reg}_{n} \) is the initial register assignment,
- \( \delta \subseteq Q \times \mathbb{L}_{n} \times Q \) is the transition relation,
- \( F \subseteq Q \) is the set of final states.

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

Transitions containing labels of the form \( i \) are called known transitions; those of the form \( \bar{a}^{*} \) are locally fresh ones; and globally fresh transitions involve \( \bar{a}^{\omega} \). Thus, an RA is a FRA with no globally fresh transitions.\(^4\)

Here is an informal reading of \( \delta \). Suppose \( \mathcal{A} \) is at state \( q_{1} \) with current register assignment \( \sigma \). If input \( i \in \mathbb{C} \cup \mathcal{A} \) arrives then:\(^5\)

\(^3\) In other presentations [11, 21] there is no such distinction, but symbols that appear in the initial register assignment can play the role of constants.

\(^4\) This yields the same notion of register automaton as that of [21].

\(^5\) Note that the same symbol, \( \bar{a} \), is later used to range over elements of \( \mathbb{L}_{n} \).
• If $\ell \in C$ and $(q_1, \ell, q_2) \in \delta$ then $A$ accepts $\ell$ and moves to $q_2$.
• If $\ell \in A$ and $(q_1, i, q_2) \in \delta$ and $\sigma(i) = \ell$ then $A$ accepts $\ell$ and moves to $q_2$.
• If $\ell \in A$ and $(q_1, i^*, q_2) \in \delta$ and $\ell$ is not stored in $\sigma$ then $A$ accepts $\ell$, it sets $\sigma(i) = \ell$ and moves to $q_2$.
• If $\ell \in A$ and $(q_1, i^*, q_2) \in \delta$ and $\ell \not\in \text{img}(\sigma_0)$ and $\ell$ has not appeared in the current run then $A$ accepts $\ell$, it sets $\sigma(i) = \ell$ and moves to $q_2$.

The above is formally defined by means of configurations representing the intended current state of the automaton, which apart from states contains information on the current register assignment and the set of names having appeared thus far (the history). The latter component is necessary for global fresh transitions.

**Definition 2.** A configuration of $A$ is a triple $(q, \sigma, H) \in \mathcal{Q}$, with

$$\mathcal{Q} = Q \times \text{Reg}_n \times \mathcal{P}_b(A)$$

and $\mathcal{P}_b(A)$ being the set of finite subsets of $A$. From $\delta$ define a transition relation on configurations

$$\mathcal{Q} \rightarrow \mathcal{Q} \times (C \cup A) \times \mathcal{Q}$$

as follows. For all $(q, \sigma, H) \in \mathcal{Q}$ and $(q', \sigma', H') \in \mathcal{Q}$:

- If $\ell \in C$ then $(q, \sigma, H) \xrightarrow{\ell} (q', \sigma', H')$.
- If $\ell = i$ and $\sigma(i) = a$ then $(q, \sigma, H) \xrightarrow{i, a} (q', \sigma', H \cup \{a\})$.
- If $\ell = i^*$ and $\sigma \not\in \text{img}(\sigma_0)$ then $(q, \sigma, H) \xrightarrow{\sigma} (q', \sigma', H')$ with $\sigma' = \sigma[i \mapsto a]$ and $H' = H \cup \{\ell\}$.
- If $\ell = \emptyset$ and $\sigma \not\in \text{img}(\sigma)$ then $(q, \sigma, H) \xrightarrow{\sigma} (q', \sigma', H')$ with $\sigma' = \sigma[i \mapsto a]$ and $H' = H \cup \{\ell\}$.

We write $\mathcal{Q} \rightarrow_s$ for the reflexive transitive closure of $\mathcal{Q} \rightarrow$.

We say that configuration $q$ is reachable if $(q_0, \sigma_0, \emptyset) \xrightarrow{\ell} q$ for some $\ell \in (A \cup C)$. We call $A$ a closed FRA if, for all reachable configurations $(q, \sigma, H)$ and all $(q', i, q') \in \delta$, we have that $\sigma(i) \not\in \emptyset$. Finally, the set of strings accepted by $A$ is

$$\mathcal{L}(A) = \{ \ell \in (A \cup C)^* \mid (q_0, \sigma_0, \emptyset) \xrightarrow{\ell} (q, \sigma, H) \land q \in \mathcal{F} \}$$

and is called the language recognised by $A$. Two automata are equivalent if they recognise the same language.

**Remark 3.** There is an equivalent definition of FRA’s in which histories include $\text{img}(\sigma_0)$ by default, and in which reachable configurations are the ones reached from $(q_0, \sigma_0, \text{img}(\sigma_0))$. Here instead we have decided to separate the history of the run from its initial names, which appears to give a cleaner presentation but it is by no means a substantial point of difference. Note also that reachable configurations contain names that have appeared before one way or another: if $(q, \sigma, H)$ is reachable then $\text{img}(\sigma) \subseteq \text{img}(\sigma_0) \cup H$.

**Example 4.** The reader can check that the language $\mathcal{L}_1 (= A^{\emptyset})$ of the Introduction is recognised by the following FRA.

$$A_0 = \{ (q_0), (q_0, \{1, 2\}), \{ (q_0, 1^+, q_0), \{q_0\} \} \}$$

Note that the FRA $B = \{ (q_0), (q_0), \{1, 2\}, \{ (q_0, 1^+, q_0), \{q_0\} \} \}$ recognises the language

$$\mathcal{L}_2 = \{ a_1 \ldots a_k \in A^* \mid k \in \omega \land \forall i. a_i \neq a_{i+1} \}$$

and is therefore not equivalent to $A$. A more elaborate example is the following. Let $A$ be the FRA:

![Diagram of an FRA]

with initial assignment $\{ (1, \sharp) \}$. The automaton works as follows. It receives a name $a$ and then keeps receiving $a$ until some $b \neq a$ arrives; then it keeps receiving $b$ until a globally fresh $c$ arrives; it then repeats from start. Thus, members of $\mathcal{L}(A)$ are of the form

$$a_0 b_0 c_0 a_1^1 b_1^1 c_1 a_2^2 b_2 c_2 \ldots a_n^mn_c$$

where, for all $i$, we have $j_i, k_i > 0$, $a_i \neq b_i$, and $c_i$ differs from all symbols preceding it. Formally, setting

$$\mathcal{L}'(H) = \{ n_i = 1 \mid n_i > 0 \land a \neq b \land c \neq b \in H \cup \{a, b\} \}$$

we have that $\mathcal{L}(A) = \bigcup_{H \subseteq A} \mathcal{L}'(H)$, where we set $\mathcal{L}_0 = \mathcal{L}'(\emptyset)$ and $\mathcal{L}_{i+1} = \{ a \in \mathcal{L}_i \mid a \in \mathcal{L}_i \land b \in \mathcal{L}'(\text{img}(b)) \}$.

**Some basic results.** The languages of FMA’s [11] are regular once constrained to a finite number of symbols. Moreover, the language accepted by an FMA is impervious to name-permutations that do not affect its initial register. These properties carry over to FRA’s, and are proved as in [11].

**Proposition 5.** Let $A = (\{ q_0, q_1, \sigma_0, \Delta \}, F)$ be an FRA of $n$ registers and $S \subseteq A$ be finite. Then, $\mathcal{L}(A) \cap S^n$ is a regular language.

**Proposition 6.** For $A$ as above, if $\bar{a} \in \mathcal{L}(A)$ and $\pi : \mathcal{L} \rightarrow A$ is such that $\pi(a) = a$ for all $a \in \text{img}(\sigma_0)$ and $\pi(\emptyset) \in \mathcal{L}(A)$.

**Bisimulation** Bisimulation equivalence turns out to be a great tool for relating automata, even from different paradigms. It implies language equivalence and, in all our cases of interest, it is not too strict in this aspect. We choose it here as our main vehicle of study.

**Definition 7.** Let $A_i = (Q_i, q_{i0}, \sigma_{i0}, \Delta_i, F_i)$ be FRAs with $n_i$ registers, for $i = 1, 2$. A relation $R \subseteq Q_1 \times Q_2$ is called a simulation on $A_1$ and $A_2$ if, for all $(q_1, q_2) \in R$,

- if $\pi_1(q_1) \in F_1$ then $\pi_1(q_2) \in F_2$,
- if $q_1 \xrightarrow{a} q_1'$ then $q_2 \xrightarrow{\sigma_{i0}(a)} q_2'$ for some $(q_1', q_2') \in R$.

$R$ is called a bisimulation if both $R$ and $R^{-1}$ are simulations. We say that $A_1$ and $A_2$ are bisimilar, written $A_1 \sim A_2$, if there is a bisimulation $R$ such that $((q_{i0}, \sigma_{i0}, \emptyset), (q_{i0}, \sigma_{i0}, \emptyset)) \in R$.

**Lemma 8.** If $A_1 \sim A_2$ then $\mathcal{L}(A_1) = \mathcal{L}(A_2)$.

The above is proved using standard methods. Bisimilarity is also called bisimulation equivalence. For instance, the automaton $A_0$ of example 4 is bisimilar to

$$B = \{ (q_0, q_1), (q_0, (1, \sharp)), \{ (q_0, 1^+, q_1), (q_1, 1^+, q_0) \}, \{q_0, q_1\} \}$$

with a bisimulation witnessing this being the following,

$$\{(q_0, \sigma_0, \emptyset), (q_0, \sigma_0, \emptyset)\} \cup\{(q_0, \sigma_1, H_1), (q_1, \sigma_2, H_2)\} | H_1 = H_2\}$$

where $\sigma_0 = \{ (1, \sharp) \}$.

**3. Bisimilar constructions.**

In this section we demonstrate some bisimilar constructions which will be useful in the sequel. Starting from a fresh-register automaton $A = (\{ q_0, q_1, \sigma_0, \Delta \}, F)$ of $n$ registers, we effectively construct the following bisimilar automata:

- The closed FRA $\overline{A}$, called the closure of $A$.
- For any $\bar{a} \in A^{\emptyset}$ with $\text{img}(\sigma_{i0}) \cap \text{img}(\bar{a}) = \emptyset$, the FRA $A \upharpoonright \bar{a}$.

This is called the extension of $A$ by $\bar{a}$, and its initial assignment is $\sigma_{i0} + \bar{a} = \sigma_{i0} \cup \{ (i + n, a_i) \mid 1 \leq i \leq |\bar{a}| \}$.

Our presentation will focus on constructing the bisimilar automata and explaining the candidate bisimulation relation $R$, omitting the actual proof that $R$ is a bisimulation, as these proofs are not difficult (but tedious) and follow directly from the constructions.
Closures For \( A \) as above with \( n \) registers we define its closure to be the \( n \)-register FRA \( \overline{A} = (Q', q_0, \sigma_0, \delta', F') \) given as follows. We set \( Q' = \mathcal{P}(\{n\}) \), \( q_0 = (q_0, \text{dom}(\sigma_0)) \), \( \sigma_0 = \sigma_0 \) and \( F' = \{ (q, S) | q \in F \} \). Recall we want to construct an automaton which is closed, that is, whenever a configuration with state \( q \) and assignment \( \sigma \) is reached and \( (i, q') \) is a transition, then \( \sigma(i) \in A \) and therefore the transition is allowed. The extra component added in \( Q' \) monitors the registers that have been assigned a name (note that once a register has been assigned a name it cannot return to the \( \sharp \) state). Consequently, \( \delta' \) will be designed in such a way so that this monitoring carries through and, moreover, the known transitions included in \( \delta' \) are always allowed:

\[
\delta' = \{(q, \ell, (q', S)) | (q, \ell, q') \in \delta \land \ell \in \mathbb{C}\} \\
\cup \{(q, i, (q', S)) | (q, i, q') \in \delta \land i \in S\} \\
\cup \{(q, S), i \in (q', S')) | (q, i \in \delta \land S' = S \cup \{i\}\}
\]

Now, we can check that the following relation is a bisimulation

\[
R = \{(q, \sigma, H), ((q, S), \sigma, H) | \text{dom}(\sigma) = S\}
\]

and therefore that \( A \sim \overline{A} \). Moreover, the reachable configurations of \( A \) are of the form \( ((q, S), \sigma, H) \) with \( \text{dom}(\sigma) = S \) and therefore the automaton is closed.

**Remark 9.** If \( \langle Q, q_0, \sigma_0, \delta, F \rangle \) is a closed FRA then each path \( q_0 \xrightarrow{\tau_1} q_1 \xrightarrow{\tau_2} \cdots \xrightarrow{\tau_m} q_m \) in \( A \) (where arrow notation represents \( \delta \)) is a path in \( A \) (where arrow notation represents \( \delta \)) according to the definition of \( \delta' \). Finally, if \( \ell_{j+1} = \sigma_i(i) \) then \( \ell_{j+1} = \delta', \sigma_{j+1} = \sigma_{j+1} = \sigma_j \) and \( H_{j+1} = H_j \cup \{\sigma_j(i)\} \). In this case, closedness of \( A \) guarantees that \( \sigma_j(i) \neq \emptyset \).

**Name extension** For \( A \) as above with \( n \) registers and \( \bar{a} \in \mathbb{A}^n \) a sequence of length \( m \) such that \( \text{im}(\sigma) \land \text{im}(\bar{a}) = \emptyset \), we define the extension \( \bar{A} \leftrightarrow \bar{a} \) as the FRA with \( n+m \) registers and description \( \langle Q', q_0', \sigma_0', \delta', F' \rangle \) given as follows. We set

\[
Q' = \mathcal{P}(\{n\}) \times \mathcal{P}(\{n+1, \ldots, n+m\})
\]

and \( q_0' = (q_0, i, (n+1, \ldots, n+m)) \), with \( i \) the inclusion function, \( F' = \{ (q, f, S) | q \in Q \} \) and \( \sigma_0 = \sigma_0 \cup \bar{a} \). Finally:

\[
\delta' = \{((q, f, S), f(\ell), (q', f, S')) | \ell \in \mathbb{C} \land q', \ell \in \delta\}
\cup \{(q, S), j \in (q', f, S')) | (q, j, q') \in \delta \land j \notin \text{im}(f)\}
\cup \{(q, S), j \in (q', f, S')) | q', q' \in \delta \land j \in S\}
\]

where \( f(\bar{a}) = \bar{f}(\bar{a}) \), \( f(\bar{a}) = \bar{f}(\bar{a}) \), \( f(\ell) = \ell \) for \( \ell \in \mathbb{C} \), \( f = f[i \in J] \) and \( S' = S \setminus J \). The transition relation in \( \bar{A} \cup \bar{a} \) proceeds as in \( A \) with the exception of locally/globally fresh transitions, where some extra care is needed. Since the registers of the new automaton contain more names than those of the initial one, fresh transitions in \( \bar{A} \cup \bar{a} \) can now capture fewer names. For example, if \( a \) is one of the added names then an \( \tau_{i+1} \) transition from the initial configuration could capture it before, but this is no more the case as \( a \) appears in \( \sigma_0 \); instead, we need an explicit \( j \) transition for this purpose. This is what the second clause of the definition of \( \delta' \) addresses. For this to work we need to introduce the component \( f \) to keep track of the correspondences between old and new registers that arise in the way just described. For globally fresh transitions a similar situation arises, only that this time we need only remember which of the names in the initial \( \bar{a} \) have not appeared in the history thus far, which is what the component \( S \) achieves. Thus, the following is a bisimulation

\[
R = \{(q, \sigma, H), ((q, f, S), \sigma', H') | \sigma = \sigma' \land \text{im}(\bar{a}) \subseteq H \land \bar{a}(\sigma(S))\}
\]

and therefore \( \bar{A} \sim \overline{\bar{A}} \).

---

4. **Finite-memory automata**

We now present FMA’s and examine their properties in relation to FRA’s and RA’s. In fact, RA’s are equivalent to FMA’s and in the literature they have been used as synonyms (e.g. compare [11] with [21]). The precise correspondence is stated in proposition 11, which is a folklore result.

Let us recall the original definition from [11]. A **finite-memory automaton (FMA)** of \( n \) registers is a sextuple \( \mathcal{A} = (Q, q_0, \sigma_0, \rho, \delta, F) \) where:

- \( Q \) is a finite set of states, with \( q_0 \in Q \) initial, and \( F \subseteq Q \) final.
- \( \sigma_0 \in \text{Reg}_n \) is the initial register assignment.
- \( \rho: Q \rightarrow [n] \) is the reassignment (partial) function.
- \( \delta \subseteq Q \times [n] \times Q \) is the transition relation.

The intuitive reading of \( \delta \) is the following. Suppose \( \mathcal{A} \) is at state \( q_1 \) with register assignment \( \sigma \) and let \( (q_1, i, q_2) \in \delta \). If \( \text{input } a \in A \) arrives then:

- If \( \sigma(i) = a \) then \( \mathcal{A} \) accepts \( a \) and moves to state \( q_2 \).
- If \( a \notin \text{im}(\sigma) \) and \( \rho(q_1) = i \) then \( \mathcal{A} \) accepts \( a \), it sets \( \sigma(i) = a \) and moves to state \( q_2 \).

Formally, a configuration is now a pair \((q, \sigma) \in Q \), where

\[
\hat{Q} = \hat{Q} \times \text{Reg}_n,
\]

and the transition relation \( \delta \subseteq \hat{Q} \times A \times \hat{Q} \) is defined as follows. For all \((q, \sigma) \in Q \) and \((q, i, q') \in \delta \):

- If \( \sigma(i) = a \) then \( (q, \sigma, a) \rightarrow (q', \sigma) \).
- If \( \rho(q) = i \) then, for all \( a \notin \text{im}(\sigma) \), \( (q, \sigma, a) \rightarrow (q', \sigma[i \rightarrow a]) \).

The notions of reachable configurations and accepted strings and languages are defined just as in the case of FRA’s.

Example 10. Recall the language \( L_2 \) of example 4:

\[
L_2 = \{ a_1 \cdots a_h \in \mathbb{A}^* | \forall i, a_i \neq a_{i+1} \}
\]

which is RA-recognisable. \( L_2 \) is recognised by the FMA:

\[
B = (Q, q_0, \sigma_0, \{(q_0, 1), (q_1, 2)\}, \{(q_0, 1, q_1), (q_1, 2, q_0)\}), Q) \]

where \( Q = (q_0, q_1) \) and \( \sigma_0 = \{1, 2, 2, 2\} \). Comparing this to \( B \) of example 4, the reader can observe how the differences between RA’s and FMA’s in reassignment have been addressed here by use of the extra register.

The main properties of FMA’s and FMA-recognisable languages have been established as follows.

(a). Emptiness is decidable for FMA’s [11] (i.e. \( L(\mathcal{A}) = \emptyset \) ?), and in particular it is NP-complete [25].

(b). The languages accepted by FMA’s are closed under union, intersection, concatenation and Kleene star; they are not closed under complement [11].

(c). Universality is undecidable [21] (i.e. \( L(\mathcal{A}) = \mathbb{A}^* \) ?). Hence, the equivalence and containment problems are undecidable too (i.e. \( L(\mathcal{A}) = L(\mathcal{B}) \) ?).

We shall see that the emptiness problem is also decidable for FRA’s (proposition 24). Clearly, FRA’s being extensions of FMA’s implies that universality of the former is undecidable, and hence the same holds for equivalence and containment. In section 6 we will examine closure properties of FRA’s and show that closure under concatenation and Kleene star are lost, closure under complement still fails, but closure under union and intersection prevail.

We now relate FMA’s to the kind of automata we have introduced previously: in essence, FMA’s are the same as RA’s. The
notions of simulation and bisimulation straightforwardly extend to FMA’s. In fact, definition 7 applies to all machines operating on the infinite alphabet \(\mathbb{C} \cup \mathbb{A}\) which have configuration graphs containing initial and final configurations. It therefore makes sense to extend these notions to RA-FMA pairs (and FRA-WFRA pairs later on).

**Proposition 11.** For any FMA \(A\) of \(n\) registers there is an effectively constructible RA \(B\) of \(n\) registers such that \(A \sim B\). Conversely, for any RA \(B\) of \(n\) registers there is an effectively constructible FMA \(A\) of \(n + 1\) registers such that \(A \sim B\).

**Proof.** Going from FMA’s to FRA’s is simple: we use the same set of states; we match each transition \((q_1, i, q_2)\) with \((q_1, i, q_2)\); and, additionally, for each transition \((q_1, i, q_2)\) where \(\rho(q_1) = i\) we add \((q_1, i^*, q_2)\). The other direction is more elaborate but apparently the construction is already known [21], so we omit it.

**Corollary 12.** The universality, equivalence and containment problems are undecidable for RA’s and FRA’s.

### 5. Weak fresh-register automata

In this section we examine a weaker version of FRA’s by concentrating on the aspect of global freshness while relaxing that of local freshness. Even though this restriction leads us to machines that do not extend FMA’s, we show that universality remains undecidable (proposition 17).

The machines we introduce operate on sets of labels

\[ L^* = \mathbb{C} \cup \{ i, i^*, i^\circ \mid i \in [n] \} \]

where \(i^\circ\) stands for "accept any name" transitions. Moreover, their registers are now taken from the sets \(\text{Reg}^*_n = [n] \rightarrow \mathbb{A} \cup \{ \emptyset \} \).

**Definition 13.** A weak fresh-register automaton (WFRA) of \(n\) registers is a quintuple \(A = (Q, q_0, \sigma_0, \delta, F)\) where:

- \(Q\) is a finite set of states, with \(q_0 \in Q\) initial, and \(F \subseteq Q\) final.
- \(\sigma_0 \in \text{Reg}^*_n\) is the initial register assignment.
- \(\delta \subseteq Q \times L^*_n \times Q\) is the transition relation.

The transition relation has the same intuitive meaning as in the case of FRA’s, with the exception that in transitions of the form \((q_1, i^?, q_2)\) \(\in \delta\) the automaton accepts any name \(a\) stores it at its \(i^\text{th}\) cell and moves to state \(q_2\). Formally, a configuration is now given as a triple \((q, \sigma, H)\) \(\in Q\), where

\[ \hat{Q} = Q \times ([n] \rightarrow (\mathbb{A} \cup \{ \emptyset \})) \times \text{Reg}^*_n(\mathbb{A}) \]

and the transition relation \(\longrightarrow \subseteq \hat{Q} \times (\mathbb{C} \cup \mathbb{A}) \times \hat{Q}\) on configurations is defined as follows. For all \((q, \sigma, H) \in \hat{Q}\) and \((q', \ell', q'')\) \(\in \hat{Q}\),

- if \(\ell \in C\) then \((q, \sigma, H) \xrightarrow{\omega} (q', \sigma, H)\);
- if \(\ell = i\) and \(\sigma(i) = a\) then \((q, \sigma, H) \xrightarrow{a} (q', \sigma', H')\);
- if \(\ell = i^?\) then \((q, \sigma, H) \xrightarrow{\sigma'(i)} (q', \sigma', H')\);
- if \(\ell = i^\circ\) and \(a \notin H \cup \text{img}(\sigma_0)\) then \((q, \sigma, H) \xrightarrow{a} (q', \sigma', H')\);

with \(\sigma' = \sigma(\cdot)\) \(a\) and \(H' = H \cup \{a\}\). Reachable configurations and accepted strings/languages are defined exactly as in FRA’s.

**Example 14.** Consider the following language,

\[ L_3 = \{ a_1 \cdots a_k b_1 \cdots b_l \in A^* \mid i \neq j, a_i \neq a_j \land b_i \neq b_j \} \]

which is in fact the concatenation of \(A^*\) with itself, and the WFRA:

```
      17

d θ

1 2

1 17

17
```

with 2 registers, both of them initially empty. Call the above \(A\). We claim that \(L(A) = A^* \cup L_3\), that is, \(s \in L(A) \iff s \notin L_3\) for all \(s \in A^*\). The forward implication is clear: if \(s \in L(A)\) then either the same name \(a\) appears three times in \(s\) (via the path \(q_0 q_1 q_2 q_3\)), or names \(a_1\) and \(a_2\) appear each twice in \(s\) without interleafing (via the path \(q_0 q_1 q_2 q_3 q_4\)). In both cases, \(s \notin L_3\).

For the opposite direction, let \(s \notin L_3\) and feed it to \(A\). Since \(s \notin A^*\), we can write \(s = s_1 a_1, s_2 b_1, \ldots, s_l a_l\) with \(s_1, s_2, \ldots, s_l \in A^*\). In \(A\), the automaton accepts \(s_1 a_1, s_2 b_1, \ldots, s_l a_l\) and \(s_2 b_2, \ldots, s_l a_l\). For simplicity, we shall simulate any-transitions (i.e. of the form \(s \xrightarrow{\delta} s'\) and \(s \xrightarrow{\delta} s''\) of \(A^*\) and known-transitions (\(j\), for all \(j\)). In the end, defining the new transition relation gets a bit involved as one has to bear reorderings in mind, which need to be accounted for before making a transition and updated afterwards.

**Lemma 15.** For any WFRA \(A\) of \(n\) registers there is an effectively constructible FRA \(B\) of \(n\) registers such that \(A \sim B\).

**Proof.** Let \(A = (Q, q_0, \sigma_0, \delta, F);\) construct \(B = (Q', q_0', \sigma_0', \delta', F')\) as follows. We set \(Q' = Q \times [n] \rightarrow [n]\) and write elements of \(Q'\) as \((q, i)\). Simulation of non-linear memory \(\sigma\) by linear memory \(\sigma'\) and reordering \(f\) is defined in the obvious manner: \(\sigma = \sigma' \circ f\). Moreover, for each \(i \in [n]\), the multiplicity of \(\sigma(i)\), i.e., the number of times it appears in \(\sigma\), is given by the size of \(f^{-1}(f(i))\); we denote this by \(\mu(i)\). We let \((\sigma_0', f_0)\) be a simulation of \(\sigma_0\) such that \(\sigma_0'\) contains no more names than \(\sigma_0\), and set \(f'_0 = (q_0, f_0)\) and \(F' = \{ (f, q) \mid q \in F \}\). We now define \(\delta'\):

\[ \delta' = \{(q, i, \ell, q', f') \mid (q, \ell, q') \in \delta \land \ell \in \mathbb{C} \} \]

\[ \cup \{(q, f), (i, q', f') \mid (q, i, q') \in \delta \land \mu(i) > 1 \} \]

\[ \cup \{(q, f), (i, q', f') \mid (q, i, q') \in \delta \land \mu(i) = 1 \} \]

\[ \cup \{(q, f), (i, q', f') \mid (q, i, q') \in \delta \land \mu(i) = 1 \} \]

\[ \cup \{(q, f), j, (i, f') \mid (q, i, f') \in \delta \land \mu(i) > 1 \} \]

\[ \cup \{(q, f), j, (i, f') \mid (q, i, f') \in \delta \land \mu(i) > 1 \} \]

where \(f' = f[i \mapsto j]\). The first line is straightforward. The second line says that receiving the name of the \(i\)-th register in \(A\) is simulated by receiving the \(f(i)\)-th name in \(B\). The same rationale is repeated in the third line, only that now we have to do a memory update and therefore we need to be careful with reorderings. In particular, storing the new name, say \(a\), in the \(f(i)\)-th register should not be allowed when \(\mu(i) > 1\); if this is the case and we set \(\sigma'(f(i)) = a\) then \(a\) still appears in \(\sigma\) but no longer appears in \(\sigma'\), breaking thus the simulation. Nonetheless, if \(\mu(i) > 1\) then there must be some \(j\) which is free in \(\sigma'\) (i.e. \(j \notin \text{img}(f)\)) and we can safely store the new name in there, updating the reordering function
accordingly. The last three lines of \( \delta' \) implement the idea that receiving any name can be matched by receiving either a locally fresh name or one of the stored ones. Thus,

\[
R = \{(q, \sigma, H), (\langle q, f \rangle, \sigma', H) \mid \sigma = \sigma' \circ f\}
\]

is a bisimulation and therefore \( A \sim B \).

We next show that the absence of locally fresh transitions in WFRAs renders them incapable of recognising FMA-recognisable languages. Combining this with the previous result we obtain that WFRAs are indeed strictly weaker than FRA's.

Lemma 16. The language \( L_2 = \{a_1 \cdots a_k \mid \forall i, a_i \neq a_{i+1} \} \) of examples 4 and 10 is not WFRA-recognisable.

Proof. Suppose \( L_2 = L(A) \), for a WFR A with \( n \) registers. Then, for any \( s \in \mathbb{N}^n \) of length \( m > 1 \), we have \( ss \in L(A) \). Let following be the transition path in A accepting it,

\[
q_0 \rightarrow q_1 \cdots \rightarrow q_m
\]

with the subpath from \( q_0 \) to \( q_m \) accepting the second copy of \( s \). Then, none of the \( a_i \)'s can be of the form \( i+1 \) as their names have appeared before. Moreover, if \( a_i = j \), then \( a_i \) can also accept the preceding symbol, contradicting the fact that \( L(A) = L_2 \). Hence, all \( a_i \)'s are in \([n]\). Choosing \( m > n \) we arrive to a contradiction.

Emptiness is decidable for WFRAs, by inheritance. More interesting, the universality problem remains undecidable, and hence the same happens for equivalence and containment.

Proposition 17. Universality is undecidable for WFRAs.

Proof. The proof is by reduction from the Post Correspondence Problem, and follows the track of the analogous proof in [21]. In particular, we show that the locally fresh transitions of the RA's constructed in that proof can be replaced by WFRA-transitions. Unlike [21], here it is necessary to use the set \( C \).

6. Closure properties

In order to establish closure properties of FRA's, and following the approach on FMA's in [11], it is useful to introduce a version of FRA's with \( m \) with \( m \)-\( \lambda \)-assignment, that is, automata that can store an input name at several of their registers at one step. In particular, assignments will now be taken from the sets \( \text{Reg}_n \). The set of labels shall we use is the following:

\[
L' = C \cup (\mathcal{P}(\mathbb{N}) \times \mathcal{P}(\mathbb{N})) \cup \{(\bot) \cup \mathcal{P}(\mathbb{N})\}
\]

Labels of the form \((S, T, \bot)\) are written simply \((S, T)\), and when we write \((S, T, A)\) we assume \( A \neq \bot \). If we want to allow for \( \bot \), we write \((S, T, \bot)\).

Definition 18. An MFR A of \( n \) registers is a quintuple \( A = (Q, q_0, \sigma_0, \delta, F) \) where:

- \( Q \) is a finite set of states, \( q_0 \in Q \) is initial and \( F \subseteq Q \) are final.
- \( \sigma_0 \in \text{Reg}_n \) is the initial register assignment.
- \( \delta \subseteq Q \times L' \times Q \) is the transition relation.

The intuitive reading of \( \delta \) is the following. If \( A \) is at state \( q_1 \) with register assignment \( \sigma \) and input \( \ell \in C \cup A \) arrives then:

- if \( \ell \in C \) and \((q_1, \ell, q_2) \in \delta \) then A accepts \( \ell \) and moves to \( q_2 \).
- if \( \ell \in \mathcal{B} \) and \((q_1, (S, T), q_2) \in \delta \) and \( |\sigma(S \rightarrow \ell)|^{-1}(\ell) = T \), i.e. \( \ell \) appears exactly in the registers in \( T \) after it is assigned to all registers \( S \), then \( A \) accepts \( \ell \), it sets \( \sigma(S) = \{\ell\} \) and moves to state \( q_2 \).

- if \( \ell \in \mathcal{A} \) and \((q_1, (S, T), q_2) \in \delta \) and \( |\sigma(S \rightarrow \ell)|^{-1}(\ell) = T \) and \( \ell \) has not appeared in the history nor does it appear in \( \sigma_0(A) \) then \( A \) accepts \( \ell \), it sets \( \sigma(S) = \{\ell\} \) and moves to state \( q_2 \).

Formally, let \( Q = Q \times \text{Reg}_n \times \mathbb{N}(A) \) be the set of configurations and define \( \delta_s \subseteq Q \times \mathcal{C} \cup A \times \mathcal{C} \) as follows. For all \((q, q, H) \in Q \):

- if \((q, q, q') \in \delta \) then \((q, q, H) \xrightarrow{n-s} (q', q, H) \).
- if \((q, (S, T), q') \in \delta \) and \( q' = \sigma[S \rightarrow a] \) and \( \sigma^{-1}(a) = T \) then \((q, q, H) \xrightarrow{n-s} (q, q', H \cup \{a\}) \).
- if \((q, (S, T), q') \in \delta \) and \( q' = \sigma[S \rightarrow a] \) and \( \sigma^{-1}(a) = T \) and \( a \notin H \cup \sigma_0(A) \) then \((q, q, H) \xrightarrow{n-s} (q', q', H \cup \{a\}) \).

Reachability and acceptance are defined as before. Note that plausible transition labels \((S, T, \bot)\) satisfy \( S \subseteq T \). Moreover, if \( S \neq T \) and \( A \neq \bot \) then the transition can only be instantiated by a name \( a \in \sigma_0(\mathbb{N}) \setminus A \) that has not yet appeared in the history but is still in some register.

Lemma 19. For any FRA A of \( n \) registers there is an effectively constructible MFR B of \( n+1 \) registers such that \( A \sim B \).

The other direction is a bit more elaborate and we achieve it in two steps. Let us say that an FRA A is pure if, for all transitions \((q_1, (S, T), q_2) \rightarrow (q_3, S, T, A)\), \( S = T \) and \( A = \mathbb{N} \).

Lemma 20. For any MFR A of \( n \) registers there is an effectively constructible pure MFR B of \( 2n \) registers such that \( A \sim B \).

Lemma 21. For any pure MFR A of \( n \) registers there is an effectively constructible FRM B of \( n \) registers such that \( A \sim B \).

We can now establish the following closure properties. Closure under union and intersection is answered positively, while closure under concatenation, Kleene star or complement fails.

Proposition 22. For FRA's A and B, the languages \( L(A) \cup L(B) \) and \( L(A) \cap L(B) \) are FRA-recognisable.

Proof. Assume MFR A's \( A' = (Q_1, q_{01}, \sigma_{01}, \delta_1, F_1) \sim A \) and \( B' = (Q_2, q_{02}, \sigma_{02}, \delta_2, F_2) \sim B \) of \( n \) and \( m \) registers respectively. For the union, construct an MFR C = \((Q, q_{0}, \sigma_0, \delta, F)\) of \( n + m \) registers, where

\[
Q = \{q_0\} \cup Q_1 \cup Q_2, \quad \sigma_0 = \sigma_{01}+\sigma_{02}, \quad F = F_1 \cup F_2 \cup \phi(F_1 \cup F_2)
\]

with \( \phi : Q_1 \cup Q_2 \rightarrow Q \) mapping \( q_{01} \) and \( q_{02} \) to \( q_0 \) and being elsewhere the identity. Finally:

\[
\delta = \{(q', q', q) \mid q \in C \cup \{q_0\} \}
\]

\[
\cup \{(q', (S \cup [m])^n, T \cup [m]^n, \sigma_1, q') \mid (q, (S, T, A), q') \in \delta_1 \}
\]

\[
\cup \{(q', ([n] \cup S^+)^n, T \cup S^+ \cup A^2, \sigma_1, q') \mid (q, (S, T, A), q') \in \delta_2 \}
\]

where \( \delta'' \subseteq \{q, \phi(q)\} \) and \( S^+ = \{i+n \mid i \in S\} \), for each \( S \subseteq \omega \) and \( \omega^+ = \omega \cup \mathbb{N} \) such that \( C = L(A) \cup L(B) \).

For the intersection, construct an MFR C = \((Q, q_{0}, \sigma_0, \delta, F)\) of \( n + m \) registers where \( Q = Q_1 \times Q_2, \quad q_0 = (q_{01}, q_{02}), \quad \sigma_0 = \sigma_{01}+\sigma_{02}, \quad F = F_1 \times F_2 \) and, assuming \( \bot \cup A = A_L \),

\[
\delta = \{(q, q, q') \mid q \in C \cap \mathbb{N} \}
\]

\[
\cup \{(q, (S_1 \cup S^+ \cup T_1 \cup T^+ \cup A_1 \cup A^2 \cup 1+q'), q') \mid q \in \mathbb{N} \}
\]

\[
\forall i \in \mathbb{N}, \quad (\pi_i(q), (S_i, T_i, A_i), \pi_i(q')) \in \delta \}
\]
It follows that $L(C) = L(A) \cap L(B)$. □

Proposition 23. There are FRA's $A$ and $B$ such that the language $L(A) \cup L(B)$ is not FRA-recognisable. Moreover, there is an FRA $A$ such that the language $L(A^*)$ is not FRA-recognisable. Finally, there is an RA $B$ such that the language $h^* \setminus L(B)$ is not FRA-recognisable.

Proof. For the first part we show that the language $L' = L_1 \cup L_2$ is not FRA-recognisable, where $L_1 = \{ \varepsilon \}$. Suppose $L'$ were recognised by an FRA $C$ of $n$ registers, so $s \in L(C)$ with $s$ being a string of $m$ distinct names. Let the following be the transition path in $C$ accepting it,

$$q_0 \longrightarrow \cdots \longrightarrow q_i \longrightarrow q_{i+1} \longrightarrow \cdots \longrightarrow q_m$$

with the subpath from $q_0$ to $q'_i$, call it $p$, accepting the second copy of $s$. As all the symbols of $s$ have already appeared before, none of the $a_i$'s is of the form $\varepsilon$. Moreover, as all the symbols in $s$ are distinct, there cannot be $i \in [n]$ and $j < j'$ such that $\alpha_j \in \{1, \ldots, n\}$ and $\alpha_{j'} = j$. Thus, one can rename all symbols present in the subpath $p$. Moreover, there cannot be $i, i' \in [n]$ and $j < j' < j''$ such that $\alpha_j \in \{i, i'\}$ and $\alpha_{j''} = j$. For this purpose we did the same, and suppose all $a_i$'s between $j$ and $j'$ are not in $[1, \ldots, n]$. Then, $s = s_1s_2s_3s_4s_5s_6$ with $a_1, a_2, a_3$ corresponding to $\alpha_{j'}$, $\alpha_j$, $\alpha_j$. But $s'$ is also allowed to accept $a_1$, hence there is $s_1'$ such that $s = s_1s_2s_3s_4s_5s_6 \in L(C)$, contradicting $L(C) = L'$. But now taking $m > n + 1$ we obtain a contradiction.

The second part is shown in a similar manner, taking as $A$ the automaton accepting the language

$$L_{2,n,a} = \{ a_0a_1 \ldots a_k \in \mathbb{A} \mid \forall i \neq j, a_i \neq a_j \}$$

for some chosen $a_0$. A similar argument to the above applies, that is, we assume $L_{2,n,a_0} = L'$ for some FRA $C$ and select $a_{0,8a_08} \in L_{2,n,a_0}$ of size big enough to yield a contradiction. Finally, it suffices to show $L' = h^* \setminus L(B)$ for an RA $B$. By example 14 we have that $L' = h^* \setminus L(A)$ for a WFA $A$ with no fresh transitions. From that, we obtain $B$ by applying lemma 15. □

7. Symbolic methods

The automata we have introduced can be viewed in two different manners: either as ordinary finite-state automata operating on constant symbols and the symbols $1, \ldots, n$ (for machines with $n$ registers), or as machines which recognise languages from an alphabet comprising a finite set of names and an infinite set of names. We use the term semantic level for the latter interpretation, and symbolic level for the former one. The semantic is of course the intended interpretation but, on the other hand, viewing our automata as operating on the finite alphabet $L_n$ is much more convenient. In this section we examine methods from the symbolic level which characterise semantic notions. More specifically, we start by giving a simple proof of decidability of FRA-emptiness by reducing the problem to FSA-emptiness. Then we proceed to our main point of focus, which is the definition of an appropriate notion of symbolic bisimilarity that is equivalent to the notion of bisimilarity we have been using thus far. As a corollary we prove that bisimilarity is decidable for FRA's.

Proposition 24. The emptiness problem is decidable for FRA's.

Proof. Given an FRA $A$ of $n$ registers, construct its closure $\overline{A}$, and take $A'$ to be the ordinary FSA with the same set of states, initial state, transition relation and final states as $\overline{A}$, and operating on the set of labels $\mathbb{L}_n$. We claim that $L(\overline{A}) = \emptyset \iff L(A') = \emptyset$.

Indeed, if $A$ accepts a string $s \in \mathbb{C} \cup \mathbb{A}^*$ then, the accepting path in $A'$ yields a string $s' \in L_n$, and $s' \in L(A')$. Conversely, if $A'$ accepts a string $s'$ then the accepting path in $A'$ is also a path in $A$ ending in an accepting state. From remark 9, we have that the latter yields a string $s \in L(\overline{A})$. □

In order to define a symbolic notion of bisimulation equivalence which captures its semantical analogue, we introduce auxiliary structures which record the way in which two register assignments are related. In particular, they record the domains of the assignments and those indices on which the two assignments coincide. A symbolic bisimulation between two automata relates states of the automata in specific record environments. At each bisimulation step the records are updated according to the specific symbolic transitions taking place. This symbolic description is shown to accurately capture what happens at the semantic level.

We adapt Stark's notion of span [28]. We call a typed span on $(n_1, n_2)$ if:

- $(i, j), (i', j') \in \rho$ implies that $i = i' \iff j = j'$.
- $\| \rho \| \subseteq \{ i \mid \exists j, (i, j) \in \rho \}$.
- $\text{dom}(\rho) \subseteq \{ j \mid \exists i, (i, j) \in \rho \}$.

We write $[n_1] = [n_2]$ for the set of typed spans on $(n_1, n_2)$. A perhaps more intuitive way to view a typed span $(S_1, \rho, S_2)$ is as a triple of relations:

$$(S_1, \rho, S_2) = (S_1, \rho, S_2) \iff \rho = \{ j \in \rho \mid (i, j) \in \rho \}$$

By abuse of notation, we write $\rho$ for the whole of $(S_1, \rho, S_2)$, in which case we also use the notation $S_1(\rho) = S_1$ and $S_2(\rho) = S_2$. Suppose $\rho : [n_1] = [n_2]$ and $(i, j) \in [n_1] \times [n_2]$ then $\rho [i \mapsto j] : [n_1] = [n_2]$ is the typed span:

$$(S_i(\rho) \cup \{ i \}) \rho \setminus \{ (i', j') \mid i = i' \wedge j' \in \rho \cap \{ (i, j) \} \}$$

This is called a span $(S_1, \rho, S_2)$ which relates register assignments $\sigma_1$ and $\sigma_2$ just in case $\rho$ is a bijection between the parts of $[n_1]$ and $[n_2]$ that have common images under $\sigma_1$ and $\sigma_2$, while $S_i$ keeps track of (the indices of) all names in $\sigma_i$. Formally, $\rho = \sigma_1 \leftrightarrow \sigma_2$ if:

$$(\sigma_1) = (S_1(\rho)) \cup \text{dom}(\sigma_2) = (S_2(\rho) \setminus \rho = \{ i \mid \sigma_1(i) = \sigma_2(j) \})$$

In this case, $\| \rho \| = |S_1(\rho)| + |S_2(\rho)| - |\text{dom}(\rho)|$ gives the total number of names in $\sigma_1$ and $\sigma_2$. Suppose, for example, that we have related state $q_i$ of automata $A_1$ to state $q_j$ of $A_2$ with respect to $\rho$. If $(q_1, i, q_j, \sigma_1)$ is a transition in $A_1$ and $i \in \text{dom}(\rho)$ then the name in register $i$ of $A_1$ (in the semantical scenario captured by the symbolic description) resides in register $\rho(i)$ of $A_2$. Consequently, $A_2$ can only simulate the transition by some $(q_2, j, \sigma_2)$. On the other hand, if $(q_1, i, q_j, \sigma_1)$ is a transition in $A_1$ then there are several factors to consider:

- Any private name of $A_1$ can be captured by $\sigma_1$. Hence, $A_2$ needs a simulating transition $(q_2, q, \sigma_2)$ for every $j \in S_2(\rho) \setminus \text{im}(\rho)$.
- Moreover, $A_2$ needs a transition for all names locally fresh to both $A_1$ and $A_2$. This can be some $(q_2, j, q_j, \sigma_2)$ but, under circumstances, it may also be some $(q_2, j, q_j)$.

In order for $(q_2, j, q_j)$ to capture all names locally fresh to $A_1$ and $A_2$, it must be the case that all names in history are present in the registers of $A_1$ and $A_2$ (so that global freshness coincide with mutual local freshness). If $A_1$ has $n_1$ registers and $A_2$ has $n_2$, and assuming that the initial register assignments for $A_1$ and $A_2$ contain the same names, the latter can only happen in case less than $n_1 + n_2$ names appear in the history.
Corollary 28. Bisimilarity is decidable for FRA's.

Proof. Let \( A_i = (Q_i, q_0, \sigma_i, \delta_i, F_i) \) be FRA's of \( n_i \) registers, for \( i = 1, 2 \). Choose \( d_1, d_2 \in \mathbb{N}^5 \) such that \( \text{img}(d_1) = \text{img}(\sigma_0) \setminus \text{img}(\sigma_1) \) and form \( A'_1 = A_1 \cup d_2 \) and \( A'_2 = A_2 \cup d_1 \). Now close these and obtain closed FRA's \( \mathcal{A}_1 \). We have \( A_1 \sim \mathcal{A}_1 \). Moreover, by the previous propositions, \( \mathcal{A}_1 \sim \mathcal{A}_2 \) and hence \( A'_1 \sim A'_2 \). As the symbolic bisimulations between \( \mathcal{A}_1 \) and \( \mathcal{A}_2 \) live in a space bounded relatively to \( |Q_1|, |Q_2|, n_1, n_2 \), we can search it exhaustively for such relations. Hence, FRA-bisimilarity is decidable.

8. Automata for the \( \pi \)-calculus

We briefly recall the definition of the \( \pi \)-calculus with early semantics and strong bisimulation [14, 26]. We use the fixed set \( \mathcal{A} \) of names for channel names, and let \( p \) range over process constants.

The set \( \Pi \) of \( \pi \)-calculus processes is given as follows,

\[
P, Q ::= 0 | \bar{a}b.P | a(b).P | [a = b]P | v.a.P | P + Q | P \pi p(d)\]

where \( a, b \in A \) and \( d \in \mathbb{A} \). Name binding is defined as usual (\( b \) is bound in \( a(b), P \) and \( v.b.P \), and processes are equated up to \( \alpha \)-equivalence. We write \( fn(P) \) for the set of names appearing free in \( P \). Process constants are accompanied by definitions of the form \( p(d) = P \), where \( d \in \mathbb{A} \) and \( fn(P) = \text{img}(\bar{d}) \).

The semantics of the calculus is early and is given via a labelled transition relation with labels:

\[
\alpha ::= \bar{a}b | \bar{a}(b) | \bar{a} | \tau
\]

Labels have free and bound occurrences of names, but they are not equated up to \( \alpha \)-equivalence.

\[
fn(ab) = \{ a, b \} \quad fn(\bar{a}(b)) = \{ a \} \quad fn(\bar{a}) = \emptyset
\]

We write \( n(\alpha) \) for \( fn(a) \cup fn(\alpha) \). The transition relation is given by the following rules (plus symmetric counterparts).

\[
\begin{align*}
&\text{OUT} & \bar{a}b.P & \xrightarrow{a} P' & \text{MATCH} & P \xrightarrow{\alpha} P' \\
&\text{INF} & a(b).P & \xrightarrow{a} P(c/b) & \text{REC} & P(\bar{a}/\bar{b}) \xrightarrow{\alpha} P' \\
&\text{OPEN} & v.a.P & \xrightarrow{a} v.a.P' & \text{OPEN} & P \xrightarrow{\alpha} P' \\
&\text{SUM} & P + Q & \xrightarrow{\alpha} P' & \text{SUM} & P \xrightarrow{\alpha} Q \xrightarrow{\alpha} Q' \\
&\text{PAR} & P \pi P' & \xrightarrow{\alpha} P' \pi Q & \text{PAR} & P \pi (\bar{a}/\bar{b}) \xrightarrow{\alpha} P' \\
&\text{CLOSE} & P | Q & \xrightarrow{\alpha} v.b(P' | Q') & \text{CLOSE} & (P | Q) \xrightarrow{\alpha} Q
\end{align*}
\]

Note how the side-conditions impose global freshness on names created using the \( \nu \) constructor. We say that process \( Q \) is a descendant of \( P \) if there is a series of transitions from \( P \) to \( Q \).

Bisimilarity is the standard notion of equivalence in the \( \pi \)-calculus; here we shall consider strong bisimulation. A relation \( R \subseteq \Pi \times \Pi \) is called a simulation if, for all \( P, P \in R \) and all \( \alpha \) with \( \text{bn}(\alpha) \cap \text{fn}(P_1, P_2) = \emptyset \), \( P_1 \xrightarrow{\alpha} P_1' \) then \( P_2 \xrightarrow{\alpha} P_2' \) for some \( P'_1, P'_2 \in R \). \( R \) is called a bisimulation if both \( R \) and \( R^{-1} \) are simulations. We say that \( P \) and \( Q \) are \( \pi \)-bisimilar, written \( P \sim Q \), if there is a bisimulation \( R \) containing \( (P, Q) \).

We now define a version of the \( \pi \)-calculus with extended syntax that is directly representable by FRA's. Since transitions are multi-symbol, and our automata can recognize one symbol at a time, they will be decomposed to atomic ones. We add sets of input and output processes which can interact for the medium of sending [receiving] a name on a chosen channel.

Definition 29. The \( \pi \)-calculus syntax is given by the sets \( \Pi_{\text{out}} \) and \( \Pi_{\text{inp}} \), with elements:

\[
P, Q ::= 0 | \bar{a}b.P | a(b).P | [a = b]P | v.a.P | P + Q | P \pi p(d) \\
P_{\text{out}} ::= b.P \mid v.a.P_{\text{out}} \mid P_{\text{out}} \mid P_{\text{inp}} \mid P_{\text{inp}}
\]

\[
P_{\text{inp}} ::= (b).P \mid v.a.P_{\text{inp}} \mid P_{\text{inp}} | P_{\text{out}} | P_{\text{out}}
\]

We write \( P_{\text{out}} \) and \( P_{\text{inp}} \) for the sets of processes with output and input, respectively.
where $a, b \in \mathcal{A}$ and $\bar{a} \in \mathcal{A}^*$. We write $\Pi$ for $\Pi \cup \Pi_{out} \cup \Pi_{in}$, and let $\bar{P}, \bar{Q}, \ldots$ range over its elements, which we equate up to $\alpha$-equivalence. Name binding is defined as expected: $b$ is bound in $\nu b. P, a(b). P$ and $(b). P$.

It is handy to introduce here some very basic notions from the theory of nominal sets [8, 23]. We call nominal structure any structure which may contain names (i.e. elements of $\mathcal{A}$), and we denote by $\text{Perm}(\mathcal{A})$ the set of finite permutations on $\mathcal{A}$ (i.e. bijections $\pi : \mathcal{A} \rightarrow \mathcal{A}$ such that $\pi(a) \neq a$ for finitely many $a \in \mathcal{A}$). For example, $\text{id} = \{ (a, a) \mid a \in \mathcal{A} \} \in \text{Perm}(\mathcal{A})$. We shall define for each set $X$ of nominal structures of interest a function

$$\cdot : \text{Perm}(\mathcal{A}) \times X \rightarrow X$$

such that $\pi, \pi' : (\pi \circ \pi') \cdot x := \pi \cdot \pi' \cdot x$ and $\text{id} \cdot x = x$, for all $x \in X$ and $\pi, \pi' \in \text{Perm}(\mathcal{A})$. $X$ will be called a nominal set if all its elements involve finitely many names, that is, for all $x \in X$ there is a finite set $\mathcal{S} \subseteq \mathcal{A}$ such that $\pi \cdot x = x$ whenever $\forall a \in \mathcal{S}. \pi(a) = a$. For example, $\text{id}$ is a nominal set with action $\pi \cdot a = \pi(a)$, and so is $\pi_b(\mathcal{A})$ with action $\pi \cdot S = \{ \pi(a) \mid a \in S \}$. Any set of non-nominal structures is a nominal set with trivial action $\pi \cdot x = x$. More interestingly, if $X$ is a nominal set then so is $X^\star$ with action $\pi \cdot x_1 \ldots x_n = (\pi \cdot x_1) \ldots (\pi \cdot x_n)$. Also, if $X$ is a nominal set then so is the set $\bigcup_{n \in \omega} \{ [n] \rightarrow X \}$ with action $\pi \cdot f = \{ (i, \pi \cdot x) \mid (i, x) \in f \}$.

Thus, $\Pi, \Pi_{out}, \Pi_{in}, \Pi$ are all nominal sets. For example,

$$\pi, a(b). \bar{c} \cdot \bar{0} = \alpha'(b'). \bar{c}' . 0$$

where $\alpha' = \pi(a), \beta' = \pi(b), c' = \pi(c)$ (note that permutations equally affect bound and free name occurrences). Similarly to $X^\star$, we have that $X \times Y$ is a nominal set whenever $X$ and $Y$ are. Note that if $X$ is a nominal set and $X' \subseteq X$ is such that $\pi \cdot x \in X'$, for all $x \in X$ and $\pi \in \text{Perm}(\mathcal{A})$, then $X'$ is also a nominal set with the inherited action. Hence, the following set is a nominal set.

$$\bar{K} = \{ (\pi, \bar{P}) \mid \pi \in \bigcup_{n \in \omega} \text{Reg}_n \wedge \bar{P} \in \Pi \wedge \text{fn}(\bar{P}) \subseteq \text{img}(\pi) \}$$

We write $K$ for the restriction of $\bar{K}$ to elements $(\pi, \bar{P})$ with $\bar{P} \in \Pi$. Finally, from a nominal set $X$ we can derive its set of orbits:

$$O(X) = \{ O(x) \mid x \in X \} \quad \text{where} \quad O(x) = \{ \pi \cdot x \mid \pi \in \text{Perm}(\mathcal{A}) \}.$$  

Note that each $O(x)$ is a nominal subset of $X$.

The technology of the previous paragraph is used for defining the transition system of the extended calculus. In contrast to the ordinary $\pi$-calculus, the transition relation we define is finitely branching, and this is achieved by considering processes-in-context and specifying channels by their context indices instead of their names. More specifically, we let $O(\bar{K})$ be the set of processes-in-context. Each such $O(\bar{K})$, $O(P)$ is written $\sigma \rightarrow \bar{P}$.

Since $\sigma \rightarrow \bar{P} = \sigma \circ \pi \rightarrow \pi \circ \bar{P}$, for any permutation $\pi$, what matters in $\sigma \rightarrow \bar{P}$ is not the specific names occurring in $\sigma$ or $P$, but only their index in $\sigma$. For example,

$$\{ (1, a), (2, c) \} \rightarrow a(b). b. \bar{c} \cdot 0 = \{ (1, a'), (2, c') \} \rightarrow a'(b'). b'. \bar{c}' . 0$$

and in essence both of these are specified by an expression e.g. like $\{ (1, a), (2, c) \}, (1, b), (2, 0)$. Borrowing notation from FRA’s, we build up on the indices idea and use transition labels of the form $i^* j^*$ for fresh inputs/outputs.

**Definition 30.** The semantics of the $\pi$-calculus is given via a labelled transition system set with states of $O(\bar{K})$ and labels:

$$\alpha ::= i^* | i^* j^* | i^* \tau | i^* j | i^* j^* | i j | j^*$$

where $i, j \in \omega$. The transition relation is given by the rules in Table 1.

Note that $\sigma \rightarrow \bar{P} \rightarrow \rightarrow \bar{P}'$ implies $|\sigma| = |\sigma'|$. Some further remarks on reduction:

- Transitions restricted to $\Pi$ use only $\tau$ and double labels, i.e. from $\{ i j \tau, i^* j, i j^* | i, j \in \omega \}$.
- Inputs are decomposed as known inputs (INP2a) and locally fresh ones (INP2b), and are therefore finitely branching. The side-conditions impose that, whenever $\sigma \rightarrow \text{P}_{\text{in}} \rightarrow \sigma' \rightarrow \bar{P}$, then $\sigma' = \sigma[\alpha \rightarrow \bar{a}], \bar{a} \notin \text{img}(\sigma)$ and $i$ is the least index such that $\sigma(i) \notin \text{fn}(\bar{P})$. Similar finiteness and minimisation apply to bound outputs (OPEN).
- Note that the CLOSE rule involves bound outputs, hence globally fresh transitions on the output side. On the input side, it is then necessary to have a matching locally fresh transition: global freshness implies local freshness.

**Example 31.** For each $a \in \mathcal{A}$, let $\sigma_a = \{ \{ (a, 1) \} \}$ and $P_a = \nu b. p(ab)$ with definition $p(ab) = \bar{a} b. \nu c. p(bc)$.

In the $\pi$-calculus, $P_a$ induces an infinitely-branching, infinite-path transition graph:

$$\bar{a}(b') \rightarrow \bar{b}(c') \rightarrow \bar{c}(d') \rightarrow \bar{d}(e') \ldots$$

In the extended calculus, $P_a$ induces the following transition graph, $\sigma_a \rightarrow P_a \rightarrow \rightarrow \sigma_b \rightarrow P_b \rightarrow \rightarrow \ldots$ which is economic by branching once at each step. In fact, setting $P_{\text{out}} = \nu b. b. \nu c. p(bc)$, and since $\sigma_a \rightarrow P_a = \sigma_b \rightarrow P_b$ and $\sigma_a \rightarrow P_{\text{out}} = \sigma_b \rightarrow P_{\text{out}}$ for all $a, b \in \mathcal{A}$, the graph above contains just two nodes:

$$\sigma_a \rightarrow P_a \rightarrow \rightarrow \sigma_b \rightarrow P_{\text{out}}$$

and using double labels we get simply $\sigma_a \rightarrow P_a \rightarrow \rightarrow \sigma_b \rightarrow P_{\text{out}}$.

The way in which the two transition relations are related is given by the following lemma, which verifies the intuitions of Table 1.

**Lemma 32.** Let $\sigma, \sigma'$ be registers, and $\alpha, \beta$ be labels of $\pi$ and $\pi'$ respectively. For all $P, P' \in \Pi$ with $\text{fn}(P) \subseteq \text{img}(\sigma)$:

- if $\sigma \rightarrow P \rightarrow \alpha \circ \sigma' \rightarrow P'$ then $P \rightarrow \alpha \circ P'$,
- if $P \rightarrow \alpha \circ P'$ then $\sigma \rightarrow P \rightarrow \alpha \circ \sigma' \rightarrow P'$,

where either $\alpha = \pi = \sigma$ and $\sigma = \sigma'$, or $\alpha = i j / i j$, $\alpha = \bar{a} b / \bar{a} b$, $\sigma(i) = a, \sigma(j) = b$ and $\sigma' = \sigma'$, or $\alpha = i j / i j^*$, $\alpha = \bar{a} b / \bar{a} b$, $\sigma(i) = a, \sigma' = \sigma[j \rightarrow \bar{b}]$ and $j = \min \{ j \mid \sigma(j) \notin \text{fn}(P') \}$.

There is a straightforward passage from the $\pi$-calculus to FRA’s: states are taken from $O(\bar{K})$, states from $O(\bar{K})$ are final, and the transition relation is the one given in Table 1 (omitting double transitions).7 However, the usual (symbolic) notion of bisimulation between FRA’s is not appropriate because it is defined for single-step transitions and, moreover, does not take into account the distinction between inputs and outputs. We therefore define the following notion.

**Definition 33.** An $n$-simulation is a relation

$$R \subseteq O(\bar{K}) \times \{ [n] \rightarrow [n] \} \times O(\bar{K})$$

6 Although not essential, minimisation saves us from unnecessary branching.

7 Note that this translation typically yields infinite FRA’s—but we shall examine classes of processes where the resulting FRA’s are finite in the end of this section.
\[\sigma \vdash \alpha(a). \overline{P} \xrightarrow{a} \sigma \vdash (b). \overline{P}\]

Table 1. The transition relation for the \(\pi\)-calculus (symmetric counterparts of \text{SUM}, \text{PAR}, \text{COMM}, \text{CLOSE} omitted\).

The set of reducts of a given process-in-context is in general

\[
\sigma \vdash P \xrightarrow{\alpha} \sigma \vdash \hat{P}
\]

finite, even if the process is \(n\)-contained. The following result provides sufficient conditions for excluding such infinite behaviours. We say that a process has \textit{finite control} if no parallel compositions appear in its recursive definitions. A process is \textit{\(\nu\)-strict} if all its subprocesses of the form \(\nu.a.P\) satisfy \(a \in \text{fn}(P)\).

\textbf{Proposition 35.} If \(P_0 \in \Pi\) has finite control and all its descendants are \(\nu\)-strict, then there are some \(M \in \omega, \sigma_0 \in \text{Reg}_{\Delta}\) and a finite \(S \subseteq O(K)\) such that \(P_0\) is \(M\)-contained, \((\sigma_0 \vdash P_0) \in S\) and for all \((\sigma \vdash P) \in S\) if \(\sigma \not\rightarrow \sigma' \not\rightarrow \sigma''\) then \((\sigma'' \rightarrow \sigma') \in S\).

\textbf{Proof.} Suppose (WLOG) that \(P_0\) invokes definitions \(p_i(\overline{a}_i) = P_i, i \in [N]\) for some \(N\), and take \(M = |P_0| \times \max\{|P_i| \mid i \in [N]\}\) for the size function which counts a process’ occurrences of \(O\)'s, \(P\)'s and names, free or bound (but not binding): e.g. \(|ab.P| = 2 + |P|, |a(b).P| = 1 + |P|, |\nu.a.P| = |P|, |p(\overline{a})| = 1 + |\overline{a}|\) and \(|0| = 1\). If \(Q\) is a descendant of \(P\) then \(|Q| \leq M\) as a process may only increase its size by recursion and, as \(P_0\) has finite control, recursions cannot obtain size greater than \(\max\{|P_i| \mid i \in [N]\}\). But then, because all descendants of \(P_0\) are \(\nu\)-strict, their number of \(\nu\)-abstractions is bounded by \(M\), and hence they all have length (number of symbols or constructors) bounded relatively to \(M\). They are still unboundedly many, due to different choices of free variables. But since each descendant can be matched with a context from \(\text{Reg}_{\Delta}\), the number of the resulting processes-in-context is bounded relatively to \(M\). We collect all these in \(\mathcal{S}\).

\textbf{Corollary 36.} Bismilarity is decidable in \(\Pi\) when restricted to processes with finite control.

\textbf{Proof.} For any such processes \(P_1, P_2 \in \Pi\), by the previous proposition and after equating processes up to non-strict \(\nu\)-abstractions, we obtain \(M\)-transition graphs with sizes bounded relatively to \(P_1\) and \(P_2\). Clearly, \(P_1 \sim P_2\) iff there is an \(M\)-bismimulation between

\[\text{note: } \sigma + v = \sigma \cup \{(\sigma + 1, v)\}, v + \sigma = \{(1, v)\} \cup \{(i+1, v') \mid (i, v') \in \sigma\}\]
those graphs. As those bisimulations live in a space bounded relatively to the sizes of $P_1$ and $P_2$, we can search it exhaustively for such relations.

Equating processes up to structural congruence [14], the above results can be further strengthened to processes with finite degree of parallelism, in a similar manner to [4].

9. Further directions

We have introduced an abstract computational paradigm and established its key properties, laying the ground for further research. The next logical step is to examine concrete applications of FRA’s to the description of computation with names, either in the direction of mobile calculi or that of programming languages, relating this approach to existing higher-level approaches. A first such advance has been recently accomplished in [19] by constructing a model of a low-order restriction of Reduced ML (a fragment of ML with ground-type integer references) representable in a variant of FRA’s where labels contain store information. This was achieved by representing the fully abstract game semantics of the language [18].

On the foundational side, the study of the $\pi$-calculus in FRA’s revealed that there is a notion of polarity inherent in computation with names. In particular, the examined FRA’s do not mix locally revealed that there is a notion of polarity inherent in computation with names, either in the direction of mobile calculi or that of programming languages, relating this approach to existing higher-level approaches. A first such advance has been recently accomplished in [19] by constructing a model of a low-order restriction of Reduced ML (a fragment of ML with ground-type integer references) representable in a variant of FRA’s where labels contain store information. This was achieved by representing the fully abstract game semantics of the language [18].

For example, we have not considered deterministic versions of restriction [9] are the subject of ongoing research.

Finally, some important questions have still not been answered. For example, we have not considered deterministic versions of FRA’s, nor examined whether FRA’s can be determinised. Assuming that in a deterministic FRA to each input string corresponds a unique path, we can see that e.g. the FRA accepting the language

$$L = \{ a_1 \cdots a_k \mid a \in \{ a_1, \ldots, a_k \} \land \forall i \neq j, a_i \neq a_j \}$$

has no deterministic equivalent. Other directions for further research concern minimisation of FRA’s (recently examined for FMA’s [2]) and the evident connections to HD-automata. Moreover, several possible extensions of FRA’s are of interest, e.g. variants with labels (data words), stores, or pushdown variants.

10. Acknowledgements

Thanks to Andrzej Murawski, Samson Abramsky, Vassilis Koutavas and Ulrich Schöpp for discussions and suggestions. Thanks also to the anonymous reviewers for their comments. Section 7 in particular is now much simpler due to a reviewer’s suggestion.

A. Proofs from section 6

Proof of Lemma 19. Let $A = (Q, q_0, \sigma_0, \delta, F)$. The construction of $B = (Q', q_0', \sigma_0', \delta', F')$ follows closely [11]. In particular, each transition of $A$ involving a name induces an assignment of that name in the extra register of $B$. If the transition were a fresh assignment then this would result in the name occurring in $B$ just once after assignment, otherwise it would occur twice. As the actual extra register of $B$ changes during this process we add an extra component in states to remember it.

We set $Q' = Q \times \{ (n + 1) \}$ and write elements of $Q'$ as $(q, \pi)$. Moreover, $q_0' = (q_0, \text{id})$, $\sigma_0' = \sigma_0 |_{n+1} \mapsto \emptyset$ and $F' = \{ (q, \pi) \mid q \in F \}$. Finally,

$$\delta' = \{ (q, \pi, \ell, (q_2, \pi) \mid \ell \in C \land (q_1, \ell, q_2) \in \delta \}$$

$$\cup \{ (q_1', \{ \pi(n+1) \}, \{ \pi(i), \pi(n+1) \}, (q_2, \pi) \mid (q_1, i, q_2) \in \delta \}$$

$$\cup \{ (q_1', \{ \pi(n+1) \}, \{ \pi(n+1) \}, (q_2, \pi') \mid (q_1, i^*, q_2) \in \delta \}$$

$$\cup \{ (q_1', \{ \pi(n+1) \}, \{ \pi(n+1) \}, [n], (q_2, \pi') \mid (q_1, i^*, q_2) \in \delta \}$$

where $q_1' = (q_1, \pi)$ and $\pi' = (\pi(i) \mapsto \pi(n+1)) \circ \pi$ (we write $(k-j)$ for the permutation that swaps $k$ and $j$). We can show that the following relation is a bisimulation and therefore that $A \sim B$.

$$R = \{ ((q, \sigma, H), ((q, \pi), \sigma', H)) \mid \forall i \in [n], \sigma(i) = \sigma'(\pi(i)) \}$$

Proof of Lemma 20. Let $A = (Q, q_0, \sigma_0, \delta, F)$ and construct $B = (Q', q_0', \sigma_0', \delta', F')$ as follows. The idea is to keep in the extra memory registers of $B$ a copy of the initial configuration $\sigma_0$ which is never touched by assignments. Thus, whenever $A$ wants to make a transition with label $(S,T,A)$, $B$ will simulate it by a transition $(S,S,\{n\})$ and transitions of the form $(S,T \cup T_a)$ where $T_a \subseteq \{n+1, \ldots, 2n\}$, $a \in \sigma_0([n] \setminus A)$ and $A$ is not in the history. In order to accomplish this we need to enrich states with information regarding whether the names in $\text{img}(\sigma_0)$ appear in the history.

Therefore, we set $Q' = Q \times \text{P}(\text{img}(\sigma_0))$, $q_0' = (q_0, \emptyset)$, $\sigma_0' = \sigma_0 + \sigma_0$, $F' = \{ (q, I) \mid q \in F \}$ and:

$$\delta' = \{ ((Q, I), (q, I), Q') \mid q \in C \land \pi, \pi' \in \delta \}$$

$$\cup \{ ((Q, I), (S,T), (q', I)) \mid (q, S, T) \in \delta \}$$

$$\cup \{ ((Q, I), (S,T \cup T_a), (q', I \cup [a])) \} \mid (q, S, T) \in \delta \}$$

$$\cup \{ ((Q, I), (S,S,\{n\}), (q', I)) \mid (q, S, S, A) \in \delta \}$$

$$\cup \{ ((Q, I), (S,T \cup T_a), (q', I \cup [a])) \} \mid (q, S, T, A) \in \delta \}$$

where $a \in \text{img}(\sigma_0)$, $T_a = \{ (n+i) \in [2n] \mid \sigma_0(i) = a \}$, $\alpha \in \sigma_0([n] \setminus A) \land I$, and $T_a \subseteq T_a$. We can check that

$$R = \{ ((q, \sigma, H), ((q, \pi), \sigma', H)) \mid I = H \cap \text{img}(\sigma_0) \land \sigma' = \sigma + \sigma_0 \}$$

is a bisimulation and therefore that $A \sim B$.

Proof of Lemma 21. Let $A = (Q, q_0, \sigma_0, \delta, F)$ and construct $B = (Q', q_0', \sigma_0', \delta', F')$ by setting $Q' = Q \times \{ n \}$ and selecting $f_0, \sigma_0$ such that $\text{img}(\sigma_0) = \text{img}(\sigma_0')$ and $\sigma_0 = \sigma_0' \circ f_0$. Moreover, set $q_0' = (q_0, f_0)$, $F' = \{ (q, f) \mid q \in F \}$ and:

$$\delta' = \{ ((Q, f), \ell, (q', f')) \mid \ell \in C \cup \{ q, q', \pi' \} \in \delta \}$$

$$\cup \{ ((Q, f), i, (q', f')) \mid (T, S) = \{ i \} \land (q, S, T) \in \delta \}$$

$$\cup \{ ((Q, f), i, (q', f')) \mid f^{-1}(i) \subseteq S \land (q, S, S) \in \delta \}$$

$$\cup \{ ((Q, f), i^*, (q', f')) \mid f^{-1}(i) \subseteq S \land (q, S, S) \in \delta \}$$

where $f' = f[S \mapsto i]$. Now, the following is a bisimulation

$$R = \{ ((q, \sigma, H), ((q, f), \sigma', H)) \mid \sigma = \sigma' \circ f \}$$

and hence $A \sim B$. 

B. Proofs from section 7

Proof of Proposition 26. It will suffice to check only non-constant transitions. So let \((q_1, \sigma_1, H), (q_2, \sigma_2, H')\) \(\in R\) due to some \((q_1, \rho, q_2) \in R\) and suppose that \((q_1, \sigma_1, H) \rightarrow_{s_1} (q_1', \sigma_1', H')\) with \(H' = H \cup \{a\}\). We do case analysis on \(a\). Below we write \(\rho[p_i \mapsto j]\) for \(\rho\)

\[ a \in image(\sigma_1) \cap image(\sigma_2), \text{say } a = \sigma_1(i) = \sigma_2(j). \]

Then it is necessary that \((q_1, i, q_1') \in \delta_1\) and \(\sigma_1' = \sigma_1\). Also, \(\rho = \sigma_1 \mapsto R\) implies \((i, j) \in \rho\), so \((q_1, \rho, q_2) \in R\). Thus, \((q_2, \sigma_2, H') \rightarrow_{s_2} (q_2', \sigma_2', H')\) and, noting that \(H' = H\) so \(h = |H'|\) so we can see that \((\langle q_1, \sigma_1, H', (q_2, \sigma_2, H')\rangle) \in R'\).

\[ a \in image(\sigma_1) \cup image(\sigma_2), \text{say } a = \sigma_1(i). \]

Then, again \((q_1, i, q_1') \in \delta_1\) and \(\sigma_1' = \sigma_1\), but \(i \in S_1(\rho) \setminus dom(\rho)\). Thus, \((q_2, j, q_2) \in R\) for some \((q_1, h, \rho, q_2) \in R\). Thus, \((q_2, \sigma_2, H') \rightarrow_{s_2} (q_2', \sigma_2', H')\), \(\sigma_2' = \sigma_2[j \mapsto a]\). Noting that \(\rho' = \sigma_1 \mapsto \sigma_2'\) and \(H' = H\), we have that \((\langle q_1, \sigma_1, H', (q_2, \sigma_2', H')\rangle) \in R'\).

\[ a \in image(\sigma_2) \setminus image(\sigma_1), \text{say } a = \sigma_2(j). \]

Since \(a \in H\) \((\langle q_1, \sigma_1, H\rangle), \) we have some \((q_1, i, q_1') \in \delta_1\), and \(\sigma_1' = \sigma_1[i \mapsto a]\). Moreover, \(j \in S_2(\rho) \setminus \rho[j]\) and therefore \((q_2, j, q_2) \in R\) for some \((q_1, h, \rho, q_2) \in R\). Thus, \((q_2, \sigma_2, H) \rightarrow_{s_2} (q_2', \sigma_2', H')\) and we can see that \((\langle q_1, \sigma_1, H', (q_2, \sigma_2', H')\rangle) \in R'\).

\[ a \in H \setminus (image(\sigma_2) \cup image(\sigma_1)), \text{so } (q_1, i, q_1') \in \delta_1\] and \(\sigma_1' = \sigma_1[i \mapsto a]\). If \(h = n\) then \(|\rho| = \{image(\sigma_1) \cup image(\sigma_2)\} \setminus \{H\} = |H'|\). Then, \((q_2, \sigma_2, H) \rightarrow_{s_2} (q_2', \sigma_2', H')\), \(\sigma_2' = \sigma_2[j \mapsto a]\). We have \(\rho' = \sigma_1 \mapsto \sigma_2'\) and \(h = |H'|\) so \((\langle q_1, \sigma_1, H', (q_2, \sigma_2', H')\rangle) \in R'\).

\[ a \notin H \text{ and say transition is due to } (q_1, i, q_1') \in \delta_1, \text{ and } \sigma_1' = \sigma_1[i \mapsto a]. \]

Then, \((q_2, j, q_2) \in R\) for some \((q_1, h, \rho, q_2) \in R\), so \((q_2, \sigma_2, H) \rightarrow_{s_2} (q_2', \sigma_2', H')\), \(\sigma_2' = \sigma_2[j \mapsto a]\). We have that \(h' = |H'|\) so \((\langle q_1, \sigma_1', H', (q_2, \sigma_2', H')\rangle) \in R'\).

Thus, \(R'\) is a simulation. If \(R\) is a symbolic bisimulation then, by symmetry, \(R'\) is a bisimulation. Finally, if \((q_0, i, q_0) \in R\) then \((\langle q_0, i, q_0, \emptyset \rangle, (q_0, q_2, \emptyset)) \in R'\).