Fresh-Register Automata

Tzevelekos, N

Fresh-Register Automata

Nikos Tzevelekos
Oxford University Computing Laboratory
nikt@comlab.ox.ac.uk

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 pi-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 not 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.

Revised version *18/11/2010*.

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 $\mathcal{A}$ 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 $1,\dots,n$, where n is the number of registers. However, $\mathcal{A}$ truly operates on the infinite set of inputs $\mathbb{A}$ (the set of names), with indices i referring to the names stored in the i-th register of $\mathcal{A}$. 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

$$q \xrightarrow{i^{\circledast}} q'$$

means that if $\mathcal{A}$ is at state q and the set of names that have appeared in its registers so far is H, then $\mathcal{A}$ 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.

$$\mathcal{L}_1 = \{\, a_1 \cdots a_k \in \mathbb{A}^* \mid \forall i \neq j.\, a_i \neq a_j \,\}$$

An intuitive way to view $\mathcal{L}_1$ 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; 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 FRA'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.

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 π-calculus in the context of fresh-register automata. We introduce the *xπ-calculus* system: a presentation of the π-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 xπ-calculus and allows for the finite representation, as ordinary FRA's, of finitary processes.[2] Moreover, we characterise strong bisimilarity by an appropriate symbolic notion in xπ. 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 ν-calculus, an extension of the simply-typed λ-calculus with references of unit type. Investigations on the ν-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 the language with fresh-name generation.

Process calculi For mobile systems like the π-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 π-calculus, as HD-transitions match 'on-the-fly' names between the source, target, and label of π-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".

[2] A process is *finitary* if its it does not grow unboundedly in parallelism.

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 π-calculus in the setting of FRA's.

2. Definitions

We distinguish between two sets of input symbols:

- an infinite set of *names*, $\mathbb{A}$, and
- a finite set of *constants*, $\mathbb{C}$.

Constants have an auxiliary role and are non-storable.[3] We let a, b, *etc.* range over names. We write $\mathbb{A}^*$ for the set of finite strings of names, and $\mathbb{A}^{\circledast}$ for its restriction to those containing pairwise distinct names. Strings $a_1 \cdots a_n$ will be typically represented by vectors $\vec{a}$, in which case $\mathsf{img}(\vec{a}) = \{a_1, \ldots, a_n\}$.

For each $n \in \omega$, we write $[n]$ for the set $\{1, \ldots, n\}$, and let

$$\mathbb{L}_n = \mathbb{C} \cup \{\, i, i^{\bullet}, i^{\circledast} \mid i \in [n] \,\} .$$

be the set of labels generated by $[n]$. Moreover, we define

$$\mathsf{Reg}_n = \{\, \sigma : [n] \to \mathbb{A} \cup \{\sharp\} \mid \forall i \neq j.\, \sigma(i) = \sigma(j) \implies \sigma(i) = \sharp \,\}$$

to be the set of *register assignments* of size n. We write $\mathsf{img}(\sigma)$ for the name-range of σ, i.e. $\mathsf{img}(\sigma) = \{\, a \in \mathbb{A} \mid \exists i.\, \sigma(i) = a \,\}$, and let $\mathsf{dom}(\sigma) = \{\, i \in [n] \mid \sigma(i) \in \mathbb{A} \,\}$. Whenever $a \notin \mathsf{img}(\sigma)$,

$$\sigma[i \mapsto a] = \{\, (i, a) \,\} \cup \{\, (j, \sigma(j)) \mid j \in [n] \setminus \{i\} \,\}$$

is an *update* of σ, for any $i \in [n]$.

Definition 1. A ***fresh-register automaton (FRA)*** of n registers is a quintuple $\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F \rangle$ where:

- Q is a finite set of states,
- q_0 is the initial state,
- $\sigma_0 \in \mathsf{Reg}_n$ 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', i such that $(q, i^{\circledast}, q') \in \delta$.

Transitions containing labels of the form i are called *known* transitions; those of the form $i^{\bullet}$ are *locally fresh* ones; and *globally fresh* transitions involve $i^{\circledast}$. Thus, an RA is an FRA with no globally fresh transitions.[4]

Here is an informal reading of δ. Suppose $\mathcal{A}$ is at state q_1 with current register assignment σ. If input $\ell \in \mathbb{C} \cup \mathbb{A}$ 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, ℓ, is later used to range over elements of $\mathbb{L}_n$.

- If $\ell \in \mathbb{C}$ and $(q_1, \ell, q_2) \in \delta$ then $\mathcal{A}$ accepts ℓ and moves to q_2.
- If $\ell \in \mathbb{A}$ and $(q_1, i, q_2) \in \delta$ and $\sigma(i) = \ell$ then $\mathcal{A}$ accepts ℓ and moves to q_2.
- If $\ell \in \mathbb{A}$ and $(q_1, i^\bullet, q_2) \in \delta$ and ℓ is not stored in σ then $\mathcal{A}$ accepts ℓ, it sets $\sigma(i) = \ell$ and moves to q_2.
- If $\ell \in \mathbb{A}$ and $(q_1, i^\circledast, q_2) \in \delta$ and $\ell \notin \mathsf{img}(\sigma_0)$ and ℓ has not appeared in the current run then $\mathcal{A}$ accepts ℓ, 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 globally fresh transitions.

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

$$\hat{Q} = Q \times \mathsf{Reg}_n \times \mathcal{P}_{\mathsf{fn}}(\mathbb{A})$$

and $\mathcal{P}_{\mathsf{fn}}(\mathbb{A})$ being the set of finite subsets of $\mathbb{A}$. From δ define a transition relation on configurations

$$\longrightarrow_\delta \subseteq \hat{Q} \times (\mathbb{C} \cup \mathbb{A}) \times \hat{Q}$$

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

- If $\ell \in \mathbb{C}$ then $(q, \sigma, H) \xrightarrow{\ell}_\delta (q', \sigma, H)$.
- If $\ell = i$ and $\sigma(i) = a$ then $(q, \sigma, H) \xrightarrow{a}_\delta (q', \sigma, H \cup \{a\})$.
- If $\ell = i^\bullet$ and $a \notin \mathsf{img}(\sigma)$ then $(q, \sigma, H) \xrightarrow{a}_\delta (q', \sigma', H')$ with $\sigma' = \sigma[i \mapsto a]$ and $H' = H \cup \{a\}$.
- If $\ell = i^\circledast$ and $a \notin H \cup \mathsf{img}(\sigma_0)$ then $(q, \sigma, H) \xrightarrow{a}_\delta (q', \sigma', H')$ with $\sigma' = \sigma[i \mapsto a]$ and $H' = H \cup \{a\}$.

We write $\twoheadrightarrow_\delta$ for the reflexive transitive closure of $\longrightarrow_\delta$.

We say that configuration $\hat{q}$ is *reachable* if $(q_0, \sigma_0, \emptyset) \overset{\vec{\ell}}{\twoheadrightarrow}_\delta \hat{q}$ for some $\vec{\ell} \in (\mathbb{A} \cup \mathbb{C})^*$. We call $\mathcal{A}$ a ***closed*** FRA if, for all reachable configurations (q, σ, H) and all $(q, i, q') \in \delta$, we have that $\sigma(i) \neq \sharp$. Finally, the set of strings *accepted* by $\mathcal{A}$ is:

$$\mathcal{L}(\mathcal{A}) = \{\, \vec{\ell} \in (\mathbb{A} \cup \mathbb{C})^* \mid (q_0, \sigma_0, \emptyset) \overset{\vec{\ell}}{\twoheadrightarrow}_\delta (q, \sigma, H) \wedge q \in F \,\}$$

and is called the *language* recognised by $\mathcal{A}$. Two automata are *equivalent* if they recognise the same language.

Remark 3. There is an equivalent definition of FRA's in which histories include $\mathsf{img}(\sigma_0)$ by default, and in which reachable configurations are the ones reached from $(q_0, \sigma_0, \mathsf{img}(\sigma_0))$. 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, σ, H) is reachable then $\mathsf{img}(\sigma) \subseteq \mathsf{img}(\sigma_0) \cup H$.

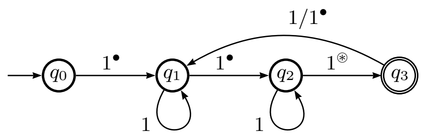
Example 4. The reader can check that the language $\mathcal{L}_1$ ($= \mathbb{A}^\circledast$) of the Introduction is recognised by the following FRA.

$$\mathcal{A}_0 = \langle \{q_0\}, q_0, \{(1, \sharp)\}, \{(q_0, 1^\circledast, q_0)\}, \{q_0\} \rangle$$

Note that the FRA $\mathcal{B} = \langle \{q_0\}, q_0, \{(1, \sharp)\}, \{(q_0, 1^\bullet, q_0)\}, \{q_0\} \rangle$ recognises the language:

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

and is therefore not equivalent to $\mathcal{A}$.
A more elaborate example is the following. Let $\mathcal{A}$ be the 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 $\mathcal{L}(\mathcal{A})$ are of the form

$$a_0^{j_0}\, b_0^{k_0}\, c_0\, a_1^{j_1}\, b_1^{k_1}\, c_1\, a_2^{j_2}\, b_2^{k_2}\, c_2 \,\ldots\, a_n^{j_n}\, b_n^{k_n}\, c_n$$

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

$$\mathcal{L}'(H) = \{\, a^{n_1} b^{n_2} c \mid n_i > 0 \wedge a \neq b \wedge c \notin H \cup \{a, b\} \,\}$$

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

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 $\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F \rangle$ be an FRA of n registers and $S \subseteq \mathbb{A}$ be finite. Then, $\mathcal{L}(\mathcal{A}) \cap S^*$ is a regular language.*

Proposition 6. *For $\mathcal{A}$ as above, if $\vec{a} \in \mathcal{L}(\mathcal{A})$ and $\pi : \mathbb{A} \xrightarrow{\cong} \mathbb{A}$ is such that $\pi(a) = a$ for all $a \in \mathsf{img}(\sigma_0)$ then $\pi(\vec{a}) \in \mathcal{L}(\mathcal{A})$.*

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 $\mathcal{A}_i = \langle Q_i, q_{0i}, \sigma_{0i}, \delta_i, F_i \rangle$ be FRA's with n_i registers, for $i = 1, 2$. A relation $R \subseteq \hat{Q}_1 \times \hat{Q}_2$ is called a ***simulation*** on $\mathcal{A}_1$ and $\mathcal{A}_2$ if, for all $(\hat{q}_1, \hat{q}_2) \in R$,

- if $\pi_1(\hat{q}_1) \in F_1$ then $\pi_1(\hat{q}_2) \in F_2$,
- if $\hat{q}_1 \xrightarrow{\ell}_{\delta_1} \hat{q}_1'$ then $\hat{q}_2 \xrightarrow{\ell}_{\delta_2} \hat{q}_2'$ for some $(\hat{q}_1', \hat{q}_2') \in R$.

R is called a ***bisimulation*** if both R and R^{-1} are simulations. We say that $\mathcal{A}_1$ and $\mathcal{A}_2$ are ***bisimilar***, written $\mathcal{A}_1 \sim \mathcal{A}_2$, if there is a bisimulation R such that $((q_{01}, \sigma_{01}, \emptyset), (q_{02}, \sigma_{02}, \emptyset)) \in R$.

Lemma 8. *If $\mathcal{A}_1 \sim \mathcal{A}_2$ then $\mathcal{L}(\mathcal{A}_1) = \mathcal{L}(\mathcal{A}_2)$.*

The above is proved using standard methods. Bisimilarity is also called *bisimulation equivalence*. For instance, the automaton $\mathcal{A}_0$ of example 4 is bisimilar to

$$\mathcal{B} = \langle \{q_0, q_1\}, q_0, \{(1, \sharp)\}, \{(q_0, 1^\bullet, q_1), (q_1, 1^\circledast, q_1)\}, \{q_0, q_1\} \rangle\,,$$

with a bisimulation witnessing this being the following,

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

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 $\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F \rangle$ of n registers, we effectively construct the following bisimilar automata.

- The closed FRA $\overline{\mathcal{A}}$, called the *closure* of $\mathcal{A}$.
- For any $\vec{a} \in \mathbb{A}^\circledast$ with $\mathsf{img}(\sigma_0) \cap \mathsf{img}(\vec{a}) = \emptyset$, the FRA $\mathcal{A} \uplus \vec{a}$. This is called the *extension* of $\mathcal{A}$ by $\vec{a}$, and its initial assignment is $\sigma_0 + \vec{a} = \sigma_0 \cup \{\, (i + n, a_i) \mid 1 \leq i \leq |\vec{a}| \,\}$.

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 $\mathcal{A}$ as above with n registers we define its *closure* to be the n-register FRA $\overline{\mathcal{A}} = \langle Q', q_0', \sigma_0', \delta', F' \rangle$ given as follows. We set $Q' = Q \times \mathcal{P}([n])$, $q_0' = (q_0, \mathsf{dom}(\sigma_0))$, $\sigma_0' = \sigma_0$ and $F' = \{ (q, S) \mid q \in F \}$. Recall we want to construct an automaton which is closed, that is, whenever a configuration with state q and assignment σ is reached and (q, i, q') is a transition, then $\sigma(i) \in \mathbb{A}$ 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, δ' will be designed in such a way so that this monitoring carries through and, moreover, the known transitions included in δ' are always allowed:

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

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

$$R = \{ ((q,\sigma,H), ((q,S),\sigma,H)) \mid \mathsf{dom}(\sigma) = S \}$$

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

Remark 9. If $\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F \rangle$ is a closed FRA then each path $q_0 \xrightarrow{\ell_1} q_1 \xrightarrow{\ell_2} \cdots \xrightarrow{\ell_m} q_m$ in $\mathcal{A}$ (where arrow notation represents δ) yields is a configuration path

$$(q_0, \sigma_0, \emptyset) \xrightarrow{\ell_1'}_\delta (q_1, \sigma_1, H_1) \xrightarrow{\ell_2'}_\delta \cdots \xrightarrow{\ell_m'}_\delta (q_m, \sigma_m, H_m)$$

according to the definition of $\longrightarrow_\delta$. For example, if $\ell_{j+1} = i$ then $\ell'_{j+1} = \sigma_j(i)$, $\sigma_{j+1} = \sigma_j$ and $H_{j+1} = H_j \cup \{\sigma_j(i)\}$. In this case, closedness of $\mathcal{A}$ guarantees that $\sigma_j(i) \neq \sharp$.

Name extension For $\mathcal{A}$ as above with n registers and $\vec{a} \in \mathbb{A}^\circledast$ a sequence of length m such that $\mathsf{img}(\sigma_0) \cap \mathsf{img}(\vec{a}) = \emptyset$, we define the extension $\mathcal{A} \uplus \vec{a}$ as the FRA with $n+m$ registers and description $\langle Q', q_0', \sigma_0', \delta', F' \rangle$ given as follows. We set

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

and $q_0' = (q_0, \iota, \{n+1, \ldots, n+m\})$, with ι the inclusion function, $F' = \{ (q, f, S) \in Q' \mid q \in F \}$ and $\sigma_0' = \sigma_0 + \vec{a}$. Finally:

$$\begin{aligned}\delta' = {} & \{ ((q,f,S), f(\ell), (q',f,S)) \mid \ell \in \mathbb{C} \wedge (q,\ell,q') \in \delta \} \\ & \cup \{ ((q,f,S), j, (q',f',S')) \mid (q, i^\bullet, q') \in \delta \wedge j \notin \mathsf{img}(f) \} \\ & \cup \{ ((q,f,S), j, (q',f',S')) \mid (q, i^\circledast, q') \in \delta \wedge j \in S \}\end{aligned}$$

where $f(i^\bullet) = f(i)^\bullet$, $f(i^\circledast) = f(i)^\circledast$, $f(\ell) = \ell$ for $\ell \in \mathbb{C}$, $f' = f[i \mapsto j]$ and $S' = S \setminus \{j\}$.
The transition relation in $\mathcal{A} \uplus \vec{a}$ proceeds as in $\mathcal{A}$ 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 $\mathcal{A} \uplus \vec{a}$ can now capture fewer names. For example, if a is one of the added names then an $i^\bullet$ transition from the initial configuration could capture it before, but this is no more the case as a appears in σ_0'; instead, we need an explicit j transition for this purpose. This is what the second clause of the definition of δ' 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 $\vec{a}$ have not appeared in the history thus far, which is what the component S achieves. Thus, the following is a bisimulation

$$R = \{ ((q,\sigma,H), ((q,f,S),\sigma',H)) \mid \sigma = \sigma' \circ f \wedge \mathsf{img}(\vec{a}) \subseteq H \uplus \sigma'(S) \}$$

and therefore $\mathcal{A} \sim \mathcal{A} \uplus \vec{a}$.

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 $\mathcal{A} = \langle Q, q_0, \sigma_0, \rho, \delta, F \rangle$ where:

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

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

- If $\sigma(i) = a$ then $\mathcal{A}$ accepts a and moves to state q_2.
- If $a \notin \mathsf{img}(\sigma)$ 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 $(q, \sigma) \in \hat{Q}$, where

$$\hat{Q} = Q \times \mathsf{Reg}_n \,,$$

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

- If $\sigma(i) = a$ then $(q, \sigma) \xrightarrow{a}_\delta (q', \sigma)$.
- If $\rho(q) = i$ then, for all $a \notin \mathsf{img}(\sigma)$, $(q, \sigma) \xrightarrow{a}_\delta (q', \sigma[i \mapsto a])$.

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 $\mathcal{L}_2$ of example 4:

$$\mathcal{L}_2 = \{ a_1 \cdots a_k \in \mathbb{A}^* \mid \forall i.\, a_i \neq a_{i+1} \}$$

which is RA-recognisable. $\mathcal{L}_2$ is recognised by the FMA:

$$\mathcal{B} = \langle Q, q_0, \sigma_0, \{(q_0, 1), (q_1, 2)\}, \{(q_0, 1, q_1), (q_1, 2, q_0)\}, Q \rangle$$

where $Q = \{q_0, q_1\}$ and $\sigma_0 = \{(1, \sharp), (2, \sharp)\}$. Comparing this to $\mathcal{B}$ 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. is $\mathcal{L}(\mathcal{A}) = \emptyset$?), 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. is $\mathcal{L}(\mathcal{A}) = \mathbb{A}^*$?). Hence, the equivalence and containment problems are undecidable too (i.e. is $\mathcal{L}(\mathcal{A}) = / \subseteq \mathcal{L}(\mathcal{B})$?).

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 $\mathcal{A}$ of n registers there is an effectively constructible RA $\mathcal{B}$ of n registers such that $\mathcal{A} \sim \mathcal{B}$. Conversely, for any RA $\mathcal{B}$ of n registers there is an effectively constructible FMA $\mathcal{A}$ of $n+1$ registers such that $\mathcal{A} \sim \mathcal{B}$.*

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 $(q_1, i^\bullet, q_2)$. The other direction is more elaborate but apparently the construction is already known [21], so we omit it. $\square$

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

$$\mathbb{L}_n^{\mathsf{w}} = \mathbb{C} \cup \{\, i, i?, i^{\circledast} \mid i \in [n] \,\},$$

where $i?$ stands for "accept any name" transitions. Moreover, their registers are now taken from the sets $\mathsf{Reg}_n^{\mathsf{w}} = [n] \to \mathbb{A} \cup \{\sharp\}$.

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

- Q is a finite set of states, with $q_0 \in Q$ initial, and $F \subseteq Q$ final.
- $\sigma_0 \in \mathsf{Reg}_n^{\mathsf{w}}$ is the initial register assignment.
- $\delta \subseteq Q \times \mathbb{L}_n^{\mathsf{w}} \times Q$ 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-th cell and moves to state q_2. Formally, a configuration is now given as a triple $(q, \sigma, H) \in \hat{Q}$, where

$$\hat{Q} = Q \times ([n] \to (\mathbb{A} \cup \sharp)) \times \mathcal{P}_{\mathsf{fn}}(\mathbb{A}),$$

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

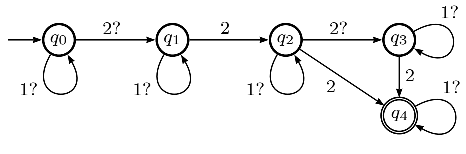
- if $\ell \in \mathbb{C}$ then $(q, \sigma, H) \xrightarrow{\ell}_\delta (q', \sigma, H)$;
- if $\ell = i$ and $\sigma(i) = a$ then $(q, \sigma, H) \xrightarrow{a}_\delta (q', \sigma, H')$;
- if $\ell = i?$ then $(q, \sigma, H) \xrightarrow{a}_\delta (q', \sigma', H')$;
- if $\ell = i^{\circledast}$ and $a \notin H \cup \mathsf{img}(\sigma_0)$ then $(q, \sigma, H) \xrightarrow{a}_\delta (q', \sigma', H')$;

with $\sigma' = \sigma[i \mapsto a]$ and $H' = H \cup \{a\}$. Reachable configurations and accepted strings/languages are defined exactly as in FRA's.

Example 14. Consider the following language,

$$\mathcal{L}_3 = \{\, a_1 \cdots a_k\, b_1 \cdots b_l \in \mathbb{A}^* \mid \forall i \neq j.\, a_i \neq a_j \wedge b_i \neq b_j \,\}$$

which is in fact the concatenation of $\mathbb{A}^{\circledast}$ with itself, and the WFRA:



with 2 registers, both of them initially empty. Call the above $\mathcal{A}$. We claim that $\mathcal{L}(\mathcal{A}) = \mathbb{A}^* \setminus \mathcal{L}_3$, that is, $s \in \mathcal{L}(\mathcal{A}) \iff s \notin \mathcal{L}_3$ for all $s \in \mathbb{A}^*$. The forward implication is clear: if $s \in \mathcal{L}(\mathcal{A})$ then either the same name a appears three times in s (via the path $q_0 q_1 q_2 q_4$), or names a_1 and a_2 appear each twice in s without interleaving (via the path $q_0 q_1 q_2 q_3 q_4$). In both cases, $s \notin \mathcal{L}'$.
For the opposite direction, let $s \notin \mathcal{L}_3$ and feed it to $\mathcal{A}$. Since $s \notin \mathbb{A}^{\circledast}$, we can write $s = s_1 a_1 s_2 a_1 s'$ with $s_1 a_1 s_2 \in \mathbb{A}^{\circledast}$. In $\mathcal{A}$, $s_1 a_1 s_2 a_1$ leads control to q_2. Now, $s \notin \mathcal{L}_3$ implies that $a_1 s' \notin \mathbb{A}^{\circledast}$ so there is some a_2 in $a_1 s'$ such that $a_1 s' = a_1 s_1' a_2 s''$, $a_1 s_1' \in \mathbb{A}^{\circledast}$ and a_2 appears in $a_1 s_1'$. If $a_2 = a_1$ then $s_1' a_2$ leads $\mathcal{A}$ directly to q_4. Otherwise, it leads to q_4 via q_3.
The reader may want to verify that changing the labels of the loops at q_0 and q_1 above to $1^{\circledast}$, and the label from q_0 to q_1 to $2^{\circledast}$, leads to a WFRA $\mathcal{A}'$ that still satisfies $\mathcal{L}(\mathcal{A}') = \mathbb{A}^* \setminus \mathcal{L}_3$.

We show that any WFRA has a bisimilar FRA of the same number of registers. The idea is to simulate the non-linear memory (i.e. a set of registers that may contain names in common) of the WFRA by a linear memory plus a *reordering function* on the FRA part. For example, here is such a simulation:

$$\{\,(1,a),(2,b),(3,b)\,\} \;\longmapsto\; \begin{cases} \{\,(1,a),(2,b),(3,c)\,\} \\ \text{plus } (1 \mapsto 1,\ 2 \mapsto 2,\ 3 \mapsto 2) \end{cases}$$

The reordering functions will be attached to the states of the FRA. Moreover, we shall simulate any-transitions (i.e. of the form $i?$) of the WFRA by means of locally-fresh-transitions ($i^\bullet$) 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 $\mathcal{A}$ of n registers there is an effectively constructible FRA $\mathcal{B}$ of n registers such that $\mathcal{A} \sim \mathcal{B}$.*

Proof. Let $\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F \rangle$; construct $\mathcal{B} = \langle Q', q_0', \sigma_0', \delta', F' \rangle$ as follows. We set $Q' = Q \times ([n] \to [n])$ and write elements of Q' as (q, f). Simulation of non-linear memory σ by linear memory σ' 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 σ, is given by the size of $f^{-1}(f(i))$; we denote this by $\mu(i)$. We let (σ_0', f_0) be a simulation of σ_0 such that σ_0' contains no more names than σ_0, and set $q_0' = (q_0, f_0)$ and $F' = \{(q, f) \mid q \in F\}$. We now define δ':

$$\begin{aligned}
\delta' = {} & \{\, ((q,f), \ell, (q',f)) \mid (q,\ell,q') \in \delta \wedge \ell \in \mathbb{C} \,\} \\
\cup {} & \{\, ((q,f), f(i), (q',f)) \mid (q,i,q') \in \delta \,\} \\
\cup {} & \{\, ((q,f), f(i)^{\circledast}, (q',f)) \mid (q,i^{\circledast},q') \in \delta \wedge \mu(i) = 1 \,\} \\
\cup {} & \{\, ((q,f), j^{\circledast}, (q',f')) \mid (q,i^{\circledast},q') \in \delta \wedge \mu(i) > 1 \wedge j \notin \mathsf{img}(f) \} \\
\cup {} & \{\, ((q,f), f(i)^{\bullet}, (q',f)) \mid (q,i?,q') \in \delta \wedge \mu(i) = 1 \,\} \\
\cup {} & \{\, ((q,f), j^{\bullet}, (q',f')) \mid (q,i?,q') \in \delta \wedge \mu(i) > 1 \wedge j \notin \mathsf{img}(f) \} \\
\cup {} & \{\, ((q,f), j, (q',f')) \mid (q,i?,q') \in \delta \,\}
\end{aligned}$$

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 $\mathcal{A}$ is simulated by receiving the $f(i)$-th name in $\mathcal{B}$. 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 σ but no longer appears in σ', breaking thus the simulation. Nonetheless, if $\mu(i) > 1$ then there must be some j which is free in σ' (i.e. $j \notin \mathsf{img}(f)$) and we can safely store the new name in there, updating the reordering function accordingly. The last three lines of δ' implement the idea that receiving any name can be matched by receiving either a locally fresh name or one of the stored ones. Thus,

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

is a bisimulation and therefore $\mathcal{A} \sim \mathcal{B}$. $\square$

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

Lemma 16. *The language $\mathcal{L}_2 = \{a_1 \cdots a_k \mid \forall i.\, a_i \neq a_{i+1}\}$ of examples 4 and 10 is not WFRA-recognisable.*

Proof. Suppose $\mathcal{L}_2 = \mathcal{L}(\mathcal{A})$, for a WFRA $\mathcal{A}$ with n registers. Then, for any $s \in \mathbb{A}^{\circledast}$ of length $m > 1$, we have $ss \in \mathcal{L}(\mathcal{A})$. Let the following be the transition path in $\mathcal{A}$ accepting it,

$$q_0 \xrightarrow{\cdots} \cdots \xrightarrow{\cdots} q_0' \xrightarrow{\alpha_1} q_1' \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_m} q_m'$$

with the subpath from q_0' to q_m' accepting the second copy of s. Then, none of the α's can be of the form $i^{\circledast}$ as their names have appeared before. Moreover, if $\alpha_i = j?$ then α_i can also accept the preceding symbol, contradicting the fact that $\mathcal{L}(\mathcal{A}) = \mathcal{L}_2$. Hence, all α's are in $[n]$. Choosing $m > n$ we arrive to a contradiction. $\square$

Emptiness is decidable for WFRA's, by inheritance. More interestingly, the universality problem remains undecidable, and hence the same happens for equivalence and containment.

Proposition 17. *Universality is undecidable for WFRA's.*

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 $\mathbb{C}$. $\square$

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 *multiple 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 $\mathsf{Reg}_n^{\mathsf{w}}$. The set of labels we shall use is the following.

$$\mathbb{L}_n' = \mathbb{C} \cup (\mathcal{P}([n]) \times \mathcal{P}([n]) \times (\{\perp\} \cup \mathcal{P}([n])))$$

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

Definition 18. An ***MFRA*** of n registers is a quintuple $\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F\rangle$ where:

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

The intuitive reading of δ is the following. If $\mathcal{A}$ is at state q_1 with register assignment σ and input $\ell \in \mathbb{C} \cup \mathbb{A}$ arrives then:

- if $\ell \in \mathbb{C}$ and $(q_1,\ell,q_2) \in \delta$ then $\mathcal{A}$ accepts ℓ and moves to q_2.
- if $\ell \in \mathbb{A}$ and $(q_1,(S,T),q_2) \in \delta$ and $(\sigma[S \mapsto \ell])^{-1}(\ell) = T$, i.e. ℓ appears exactly in the registers in T after it is assigned to all registers in S, then $\mathcal{A}$ accepts ℓ, it sets $\sigma(S) = \{\ell\}$ and moves to state q_2.
- if $\ell \in \mathbb{A}$ and $(q_1,(S,T,A),q_2) \in \delta$, $(\sigma[S \mapsto \ell])^{-1}(\ell) = T$ and ℓ has not appeared in the history nor does it appear in $\sigma_0(A)$ then $\mathcal{A}$ accepts ℓ, it sets $\sigma(S) = \{\ell\}$ and moves to state q_2.

Thus, labels of the form (S,T) work in the same way as in *M-automata* [11], and the main novelty here is the inclusion of (S,T,A): in order for the transition to be allowed, the input name a must be fresh in the history and in the part of σ_0 specified by A. This addition allows us to model globally fresh transitions and also to combine automata unifying their initial assignments.

Formally, let $\hat{Q} = Q \times \mathsf{Reg}_n^{\mathsf{w}} \times \mathcal{P}_{\mathsf{fn}}(\mathbb{A})$ be the set of configurations and define $\longrightarrow_\delta \subseteq \hat{Q} \times (\mathbb{C} \cup \mathbb{A}) \times \hat{Q}$ as follows. For all $(q,\sigma,H) \in \hat{Q}$:

- If $(q,\ell,q') \in \delta$ with $\ell \in \mathbb{C}$ then $(q,\sigma,H) \xrightarrow{\ell}_\delta (q',\sigma,H)$.
- If $(q,(S,T),q') \in \delta$, $\sigma' = \sigma[S \mapsto a]$ and $\sigma'^{-1}(a) = T$ then $(q,\sigma,H) \xrightarrow{a}_\delta (q',\sigma',H \cup \{a\})$.
- If $(q,(S,T,A),q') \in \delta$, $\sigma' = \sigma[S \mapsto a]$, $\sigma'^{-1}(a) = T$ and $a \notin H \cup \sigma_0(A)$ then $(q,\sigma,H) \xrightarrow{a}_\delta (q',\sigma',H \cup \{a\})$.

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

Lemma 19. *For any FRA $\mathcal{A}$ of n registers there is an effectively constructible MFRA $\mathcal{B}$ of $n+1$ registers such that $\mathcal{A} \sim \mathcal{B}$*

The other direction is a bit more elaborate and we achieve it in two steps. Let us say that an MFRA $\mathcal{A}$ is *pure* if, for all transitions $(q,(S,T,A),q')$ of $\mathcal{A}$, $S = T$ and $A = [n]$.

Lemma 20. *For any MFRA $\mathcal{A}$ of n registers there is an effectively constructible pure MFRA $\mathcal{B}$ of $2n$ registers such that $\mathcal{A} \sim \mathcal{B}$.*

Lemma 21. *For any pure MFRA $\mathcal{A}$ of n registers there is an effectively constructible FRA $\mathcal{B}$ of n registers such that $\mathcal{A} \sim \mathcal{B}$.*

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 $\mathcal{A}$ and $\mathcal{B}$, the languages $\mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B})$ and $\mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$ are FRA-recognisable.*

Proof. Assume MFRA's $\mathcal{A}' = \langle Q_1, q_{01}, \sigma_{01}, \delta_1, F_1\rangle \sim \mathcal{A}$ and $\mathcal{B}' = \langle Q_2, q_{02}, \sigma_{02}, \delta_2, F_2\rangle \sim \mathcal{B}$ of n, m registers respectively. For the union, construct an MFRA $\mathcal{C} = \langle Q, q_0, \sigma_0, \delta, F\rangle$ of $n+m$ registers, where

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

with $\phi : Q_1 \uplus Q_2 \to Q$ mapping q_{01} and q_{02} to q_0, and being elsewhere the identity. Finally:

$$\begin{aligned}\delta = {} & \{\, (q'',\ell,q') \mid \ell \in \mathbb{C} \wedge (q,\ell,q') \in \delta_1 \cup \delta_2 \,\}\\ & \cup \{\, (q'',(S \cup [m]^{+n}, T \cup [m]^{+n}, A_\perp),q') \mid (q,(S,T,A_\perp),q') \in \delta_1 \}\\ & \cup \{\, (q'',([n] \cup S^{+n}, [n] \cup T^{+n}, A_\perp^{+n}),q') \mid (q,(S,T,A_\perp),q') \in \delta_2 \}\end{aligned}$$

where $q'' \in \{q, \phi(q)\}$ and $S^{+n} = \{\, i+n \mid i \in S \}$, for each $S \subseteq \omega$, and $\perp^{+n} = \perp$. It follows that $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A}) \cup \mathcal{L}(\mathcal{B})$.
For the intersection, construct an MFRA $\mathcal{C} = \langle Q, q_0, \sigma_0, \delta, F\rangle$ of $n+m$ registers where $Q = Q_1 \times Q_2$, $q_0 = (q_{01}, q_{02})$, $\sigma_0 = \sigma_{01} + \sigma_{02}$, $F = F_1 \times F_2$ and, assuming $\perp \cup A_\perp = A_\perp$:

$$\begin{aligned}\delta = {} & \{\, (q,\ell,q') \mid \ell \in \mathbb{C} \wedge \forall i \in [2].\, (\pi_i(q),\ell,\pi_i(q')) \in \delta_i \,\}\\ & \cup \{\, (q,(S_1 \cup S_2^{+n}, T_1 \cup T_2^{+n}, A_{\perp 1} \cup A_{\perp 2}^{+n}),q') \mid \\ & \qquad \forall i \in [2].\, (\pi_i(q),(S_i,T_i,A_{\perp i}),\pi_i(q')) \in \delta_i \,\}\end{aligned}$$

It follows that $\mathcal{L}(\mathcal{C}) = \mathcal{L}(\mathcal{A}) \cap \mathcal{L}(\mathcal{B})$. ☐

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

Proof. For the first part we show that the language $\mathcal{L}' = \mathcal{L}_1 * \mathcal{L}_1$ is not FRA-recognisable, where $\mathcal{L}_1 = \mathbb{A}^{\circledast}$. Suppose $\mathcal{L}'$ were recognised by an FRA $\mathcal{C}$ of n registers, so $ss \in \mathcal{L}(\mathcal{C})$ with s being a string of m distinct names. Let the following be the transition path in $\mathcal{C}$ accepting it,

$$q_0 \xrightarrow{\cdots} \cdots \xrightarrow{\cdots} q_0' \xrightarrow{\alpha_1} q_1' \xrightarrow{\alpha_2} \cdots \xrightarrow{\alpha_m} q_m'$$

with the subpath from q_0' to q_m', call it p, accepting the second copy of s. As all the symbols of s have already appeared before, none of the α's is of the form $i^{\circledast}$. Moreover, as all the symbols in s are distinct, there cannot be $i \in [n]$ and $j < j'$ such that $\alpha_j \in \{i, i^\bullet\}$ and $\alpha_{j'} = i$, as $\alpha_{j'}$ would then repeat a name already present in the subpath p. Moreover, there cannot be $i, i' \in [n]$ and $j < j' < j''$ such that $\alpha_j \in \{i, i^\bullet\}$, $\alpha_{j'} = i^\bullet$ and $\alpha_{j''} = i'^\bullet$. For suppose this were the case, and suppose that all α's between j and j' are not in $\{i, i^\bullet\}$, and that all α's between j' and j'' are not in $\{i'^\bullet \mid i' \in [n]\}$. Then, $s = s_1 a_1 s_2 a_2 s_3 a_3 s_4$ with a_1, a_2, a_3 corresponding to $\alpha_j, \alpha_{j'}, \alpha_{j''}$ respectively. But $i'^\bullet$ is also allowed to accept a_1, hence there is s_4' such that $s s_1 a_1 s_2 a_2 s_3 a_1 s_4' \in \mathcal{L}(\mathcal{C})$, contradicting $\mathcal{L}(\mathcal{C}) = \mathcal{L}'$. But now taking $m > n+1$ we obtain a contradiction.
The second part is shown in a similar manner, taking as $\mathcal{A}$ the automaton accepting the language

$$\mathcal{L}_{2,a_0} = \{\, a_0 a_1 \dots a_k \in \mathbb{A}^* \mid \forall i \neq j.\, a_i \neq a_j \,\}$$

for some chosen a_0. A similar argument to the above applies, that is, we assume $\mathcal{L}_{2,a_0}^* = \mathcal{L}(\mathcal{C})$ for some FRA $\mathcal{C}$ and select $a_0 s a_0 s \in \mathcal{L}_{2,a_0}^*$ of size big enough to yield a contradiction.
Finally, it suffices to show $\mathcal{L}' = \mathbb{A}^* \setminus \mathcal{L}(\mathcal{B})$ for an RA $\mathcal{B}$. By example 14 we have that $\mathcal{L}' = \mathbb{A}^* \setminus \mathcal{L}(\mathcal{A})$ for a WFRA $\mathcal{A}$ with no fresh transitions. From that, we obtain $\mathcal{B}$ 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 $1, 1^\bullet, 1^{\circledast}, \dots, n^{\circledast}$ (for machines with n registers), or as machines which recognise languages from an alphabet comprising a finite set of constants 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 $\mathbb{L}_n$ 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. We then 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 $\mathcal{A}$ of n registers, construct its closure $\overline{\mathcal{A}}$, and take $\mathcal{A}'$ to be the ordinary FSA with the same set of states, initial state, transition relation and final states as $\overline{\mathcal{A}}$, and operating on the set of labels $\mathbb{L}_n$. We claim that $\mathcal{L}(\overline{\mathcal{A}}) = \emptyset \iff \mathcal{L}(\mathcal{A}') = \emptyset$. Indeed, if $\overline{\mathcal{A}}$ accepts a string $s \in \mathbb{C} \cup \mathbb{A}^*$ then, the accepting path in $\overline{\mathcal{A}}$ yields a string $s' \in \mathbb{L}_n^*$, and $s' \in \mathcal{L}(\mathcal{A}')$. Conversely, if $\mathcal{A}'$ accepts a string s' then the accepting path in $\mathcal{A}'$ is also a path in $\overline{\mathcal{A}}$ ending in an accepting state. From remark 9, we have that the latter yields a string $s \in \mathcal{L}(\overline{\mathcal{A}})$. ☐

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 semantical level.

We adapt Stark's notion of *span* [28]. We call

$$(S_1, \rho, S_2) \in \mathcal{P}([n_1]) \times \mathcal{P}([n_1] \times [n_2]) \times \mathcal{P}([n_2])$$

a ***typed span*** on (n_1, n_2) if:

- $(i,j), (i',j') \in \rho$ implies that $i = i' \iff j = j'$,
- $\mathsf{img}(\rho) \subseteq S_2$, where $\mathsf{img}(\rho) = \{\, i \in [n] \mid \exists j.\, (j,i) \in \rho \,\}$,
- $\mathsf{dom}(\rho) \subseteq S_1$, where $\mathsf{dom}(\rho) = \{\, j \in [n] \mid \exists i.\, (j,i) \in \rho \,\}$.

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

$$S_1 \hookrightarrow \mathsf{dom}(\rho) \xrightarrow{\simeq} \mathsf{img}(\rho) \hookleftarrow S_2$$

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

$$(S_1(\rho) \cup \{i\}, \rho \setminus \{(i',j') \mid i = i' \vee j = j'\} \cup \{(i,j)\}, S_2(\rho) \cup \{j\})$$

A typed span (S_1, ρ, S_2) relates register assignments σ_1 and σ_2 just in case ρ is a bijection between the parts of $[n_1]$ and $[n_2]$ that have common images under σ_1 and σ_2, while S_i keeps track of (the indices of) all names in σ_i. Formally, $\rho = \sigma_1 \leftrightarrow \sigma_2$ if:

$$\mathsf{dom}(\sigma_1) = S_1(\rho) \wedge \mathsf{dom}(\sigma_2) = S_2(\rho) \wedge \rho = \{(i,j) \mid \sigma_1(i) = \sigma_2(j)\}$$

In this case, $\|\rho\| = |S_1(\rho)| + |S_2(\rho)| - |\mathsf{dom}(\rho)|$ gives the total number of names in σ_1 and σ_2.

Suppose, for example, that we have related state q_1 of automaton $\mathcal{A}_1$ to state q_2 of $\mathcal{A}_2$ with respect to ρ. If (q_1, i, q_1') is a transition in $\mathcal{A}_1$ and $i \in \mathsf{dom}(\rho)$ then the name in register i of $\mathcal{A}_1$ (in the semantical scenario captured by the symbolic description) resides in register $\rho(i)$ of $\mathcal{A}_2$. Consequently, $\mathcal{A}_2$ can only simulate the transition by some $(q_2, \rho(i), q_2')$. On the other hand, if $(q_1, i^\bullet, q_1')$ is a transition in $\mathcal{A}_1$ then there are several factors to consider:

- Any private name of $\mathcal{A}_2$ can be captured by $i^\bullet$. Hence, $\mathcal{A}_2$ needs a simulating transition (q_2, j, q_2') for every $j \in S_2(\rho) \setminus \mathsf{img}(\rho)$.
- Moreover, $\mathcal{A}_2$ needs a transition for all names locally fresh to both $\mathcal{A}_1$ and $\mathcal{A}_2$. This can be some $(q_2, j^\bullet, q_2')$ but, under circumstances, it may also be some $(q_2, j^{\circledast}, q_2')$.

In order for $(q_2, j^{\circledast}, q_2')$ to capture all names locally fresh to $\mathcal{A}_1$ and $\mathcal{A}_2$, it must be the case that all names in history are present in the registers of $\mathcal{A}_1$ and $\mathcal{A}_2$ (so that global freshness coincide with mutual local freshness). If $\mathcal{A}_1$ has n_1 registers and $\mathcal{A}_2$ has n_2, and assuming that the initial register assignments for $\mathcal{A}_1$ and $\mathcal{A}_2$ contain the same names, the latter can only happen in case less than $n_1 + n_2$ names appear in the history.

We can therefore resolve the latter case by adding a component which counts the names in the history, up to $n_1 + n_2$. In the following we write n for n_1+n_2, and set $h^{++} = \lceil h+1 \rceil^n$ $(= h+1$ if $h < n$, and n otherwise).

Definition 25. Let $\mathcal{A}_i = \langle Q_i, q_{0i}, \sigma_{0i}, \delta_i, F_i \rangle$ be FRA's of n_i registers, for $i = 1, 2$, such that $\mathsf{img}(\rho_{01}) = \mathsf{img}(\rho_{02}) = H_0$. A ***symbolic simulation*** on $\mathcal{A}_1$ and $\mathcal{A}_2$ is a relation

$$R \subseteq Q_1 \times ([n] \cup \{0\}) \times ([n_1] \rightleftharpoons [n_2]) \times Q_2$$

such that, whenever $(q_1, h, \rho, q_2) \in R$, if $q_1 \in F_1$ then $q_2 \in F_2$ and if $(q_1, \ell, q_1') \in \delta_1$ then:

1. If $\ell \in \mathbb{C}$ then $(q_2, \ell, q_2') \in \delta_2$ for some $(q_1', h, \rho, q_2') \in R$.
2. If $\ell = i$ and $i \in \mathsf{dom}(\rho)$ then $(q_2, \rho(i), q_2') \in \delta_2$ for some $(q_1', h, \rho, q_2') \in R$.
3. If $\ell = i$ and $i \in S_1(\rho) \setminus \mathsf{dom}(\rho)$ then $(q_2, j^\bullet, q_2') \in \delta_2$ for some $(q_1', h, \rho[i \leftrightarrow j], q_2') \in R$.
4. If $\ell = i^\bullet$ then, for any $j \in S_2(\rho) \setminus \mathsf{img}(\rho)$, $(q_2, j, q_2') \in \delta_2$ for some $(q_1', h, \rho[i \leftrightarrow j], q_2') \in R$.
5. If $\ell = i^\bullet$ and $h = n$ or $\|\rho\| < h$ then $(q_2, j^\bullet, q_2') \in \delta_2$ for some $(q_1', h, \rho[i \leftrightarrow j], q_2') \in R$.
6. If $\ell \in \{i^\bullet, i^\circledast\}$ then $(q_2, j^\bullet, q_2') \in \delta_2$, or $(q_2, j^\circledast, q_2') \in \delta_2$, for some $(q_1', h^{++}, \rho[i \leftrightarrow j], q_2') \in R$.

Setting $(S_1, \rho, S_2)^{-1} = (S_2, \rho^{-1}, S_1)$, the inverse of R is:

$$R^{-1} = \{ (q_2, h, \rho, q_1) \mid (q_1, h, \rho^{-1}, q_2) \in R \}\,.$$

We say that R is a ***symbolic bisimulation*** if both R and R^{-1} are symbolic simulations. We say that $\mathcal{A}_1$ and $\mathcal{A}_2$ are ***symbolic bisimilar***, written $\mathcal{A}_1 \overset{s}{\sim} \mathcal{A}_2$, if there is a symbolic bisimulation R on $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $(q_{01}, h_0, \rho_0, q_{02}) \in R$ with $h_0 = |H_0|$ and $\rho_0 = \sigma_{01} \leftrightarrow \sigma_{02}$.

In the following propositions let us assume the hypotheses of Definition 25. Let us also write $\hat{H}$ for $H \cup H_0$, and n for $n_1 + n_2$.

Proposition 26. *If R is a symbolic simulation on $\mathcal{A}_1$ and $\mathcal{A}_2$ then*

$$\begin{aligned} R' = \{ ((q_1, \sigma_1, H), (q_2, \sigma_2, H)) \mid (q_1, h, \rho, q_2) \in R \\ \wedge \rho = \sigma_1 \leftrightarrow \sigma_2 \wedge h = \lceil |\hat{H}| \rceil^n \wedge \mathsf{img}(\sigma_i) \subseteq \hat{H} \,\} \end{aligned}$$

is a simulation. Moreover, if R is a symbolic bisimulation then R' is a bisimulation.

Proposition 27. *If $\mathcal{A}_1$ and $\mathcal{A}_2$ are closed FRA's and R is a simulation on $\mathcal{A}_1$ and $\mathcal{A}_2$ then*

$$\begin{aligned} R' = \{ (q_1, h, \rho, q_2) \mid ((q_1, \sigma_1, H), (q_2, \sigma_2, H)) \in R \\ \wedge \rho = \sigma_1 \leftrightarrow \sigma_2 \wedge h = \lceil |\hat{H}| \rceil^n \wedge (q_i, \sigma_i, H) \textit{ reachable} \,\} \end{aligned}$$

is a symbolic simulation. Moreover, if R is a bisimulation then R' is a symbolic bisimulation.

Corollary 28. *Bisimilarity is decidable for FRA's.*

Proof. Let $\mathcal{A}_i = \langle Q_i, q_{0i}, \sigma_{0i}, \delta_i, F_i \rangle$ be FRA's of n_i registers, for $i = 1, 2$. Choose $\vec{a}_1, \vec{a}_2 \in \mathbb{A}^\circledast$ such that $\mathsf{img}(\vec{a}_i) = \mathsf{img}(\sigma_{0i}) \setminus \mathsf{img}(\sigma_{0\bar{i}})$, and form $\mathcal{A}_1' = \mathcal{A}_1 \uplus \vec{a}_2$ and $\mathcal{A}_2' = \mathcal{A}_2 \uplus \vec{a}_1$. Now close these and obtain closed FRA's $\overline{\mathcal{A}_i'}$. We have $\mathcal{A}_i \sim \overline{\mathcal{A}_i'}$. Moreover, by the previous propositions, $\overline{\mathcal{A}_1'} \sim \overline{\mathcal{A}_2'} \iff \overline{\mathcal{A}_1'} \overset{s}{\sim} \overline{\mathcal{A}_2'}$, and hence $\mathcal{A}_1 \sim \mathcal{A}_2 \iff \overline{\mathcal{A}_1'} \overset{s}{\sim} \overline{\mathcal{A}_2'}$. As the symbolic bisimulations between $\overline{\mathcal{A}_1'}$ and $\overline{\mathcal{A}_2'}$ 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 π-calculus

We briefly recall the definition of the π-calculus with early semantics and strong bisimulation [14, 26]. We use the fixed set $\mathbb{A}$ of names for *channel names*, and let p range over *process constants*. The set Π of π-calculus processes is given as follows,

$$P, Q ::= \mathbf{0} \mid \bar{a}b.P \mid a(b).P \mid [a = b]P \mid \nu a.P \mid P{+}Q \mid P \,|\, Q \mid p(\vec{a})$$

where $a, b \in \mathbb{A}$ and $\vec{a} \in \mathbb{A}^*$. Name binding is defined as usual (b is bound in $a(b).P$ and $\nu b.P$), and processes are equated up to α-equivalence. We write $\mathsf{fn}(P)$ for the set of names appearing free in P. Process constants are accompanied by *definitions* of the form $p(\vec{a}) = P$, where $\vec{a} \in \mathbb{A}^\circledast$ and $\mathsf{fn}(P) = \mathsf{img}(\vec{a})$. Moreover, each occurrence of p must be *guarded*, i.e. it must come in one of the forms $\bar{a}b.p(\vec{a})$ or $a(b).p(\vec{a})$.

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

$$\alpha ::= \bar{a}b \mid \bar{a}(b) \mid ab \mid \tau$$

Labels have free and bound occurrences of names, but they are not equated up to α-equivalence.

$$\begin{array}{lll} \mathsf{fn}(\bar{a}b) = \mathsf{fn}(ab) = \{a, b\} & \mathsf{fn}(\bar{a}(b)) = \{a\} & \mathsf{fn}(\tau) = \emptyset \\ \mathsf{bn}(\bar{a}b) = \mathsf{bn}(ab) = \emptyset & \mathsf{bn}(\bar{a}(b)) = \{b\} & \mathsf{bn}(\tau) = \emptyset \end{array}$$

We write $\mathsf{n}(\alpha)$ for $\mathsf{fn}(\alpha) \cup \mathsf{bn}(\alpha)$. The transition relation is given by the following rules (plus symmetric counterparts).

$$\textsc{Out}\ \dfrac{}{\bar{a}b.P \xrightarrow{\bar{a}b} P} \qquad \textsc{Match}\ \dfrac{P \xrightarrow{\alpha} P'}{[a = a]P \xrightarrow{\alpha} P'}$$

$$\textsc{Inp}\ \dfrac{}{a(b).P \xrightarrow{ac} P\{c/b\}} \qquad \textsc{Rec}\ \dfrac{P\{\vec{a}/\vec{b}\} \xrightarrow{\alpha} P'}{p(\vec{a}) \xrightarrow{\alpha} P'}\ {\scriptstyle p(\vec{b})=P}$$

$$\textsc{Open}\ \dfrac{P \xrightarrow{\bar{a}b} P'}{\nu b.P \xrightarrow{\bar{a}(b)} P'}\ {\scriptstyle a \neq b} \qquad \textsc{Res}\ \dfrac{P \xrightarrow{\alpha} P'}{\nu a.P \xrightarrow{\alpha} \nu a.P'}\ {\scriptstyle a \notin \mathsf{n}(\alpha)}$$

$$\textsc{Sum}\ \dfrac{P \xrightarrow{\alpha} P'}{P{+}Q \xrightarrow{\alpha} P'} \qquad \textsc{Comm}\ \dfrac{P \xrightarrow{\bar{a}b} P' \quad Q \xrightarrow{ab} Q'}{P \,|\, Q \xrightarrow{\tau} P' \,|\, Q'}$$

$$\textsc{Par}\ \dfrac{P \xrightarrow{\alpha} P'}{P \,|\, Q \xrightarrow{\alpha} P' \,|\, Q}\ {\scriptstyle \mathsf{bn}(\alpha) \cap \mathsf{fn}(Q) = \emptyset}$$

$$\textsc{Close}\ \dfrac{P \xrightarrow{\bar{a}(b)} P' \quad Q \xrightarrow{ab} Q'}{P \,|\, Q \xrightarrow{\tau} \nu b.(P' \,|\, Q')}\ {\scriptstyle b \notin \mathsf{fn}(Q)}$$

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

Bisimulation is the standard notion of equivalence in the π-calculus; here we shall consider strong bisimulation. A relation $R \subseteq \Pi \times \Pi$ is called a ***simulation*** if, for all $(P_1, P_2) \in R$ and all α with $\mathsf{bn}(\alpha) \cap \mathsf{fn}(P_1, P_2) = \emptyset$, if $P_1 \xrightarrow{\alpha} P_1'$ 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 π***-bisimilar***, written $P \overset{\pi}{\sim} Q$, if there is a bisimulation R containing (P, Q).

We now define a version of the π-calculus with extended syntax that is directly representable by FRA's. Since transitions are multi-symbol, and our automata can recognise one symbol at a time, they will be decomposed to atomic ones. We add sets of *input* and *output* processes which cater for the intermediate stages in these decompositions. For example,

$$\bar{a}b.P \xrightarrow{\bar{a}b} P \quad \textit{decomposes to} \quad \bar{a}b.P \xrightarrow{a} b.P \xrightarrow{b} P$$

where $b.P$ is an output process. Output [resp. input] processes are in the middle of sending [receiving] a name on a chosen channel.

Definition 29. The xπ***-calculus*** syntax is given by the sets Π, Π_{out} and Π_{inp}, with elements:

$$\begin{aligned} P, Q &::= \mathbf{0} \mid \bar{a}b.P \mid a(b).P \mid [a = b]P \mid \nu a.P \mid P{+}Q \mid P \,|\, Q \mid p(\vec{a}) \\ P_{\mathsf{out}} &::= b.P \mid \nu a.P_{\mathsf{out}} \mid P \,|\, P_{\mathsf{out}} \mid P_{\mathsf{out}} \,|\, P \\ P_{\mathsf{inp}} &::= (b).P \mid \nu a.P_{\mathsf{inp}} \mid P \,|\, P_{\mathsf{inp}} \mid P_{\mathsf{inp}} \,|\, P \end{aligned}$$

where $a, b \in \mathbb{A}$ and $\vec{a} \in \mathbb{A}^*$. We write $\hat{\Pi}$ for $\Pi \cup \Pi_{\mathsf{out}} \cup \Pi_{\mathsf{inp}}$, and let $\hat{P}, \hat{Q}, \dots$ range over its elements, which we equate up to α-equivalence. Name binding is defined as expected: b is bound in $\nu b.\hat{P}, a(b).P$ 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 $\mathbb{A}$), and we denote by $\mathsf{Perm}(\mathbb{A})$ the set of finite permutations on $\mathbb{A}$ (i.e. bijections $\pi : \mathbb{A} \to \mathbb{A}$ such that $\pi(a) \neq a$ for finitely many $a \in \mathbb{A}$). For example, $\mathsf{id} = \{(a, a) \,|\, a \in \mathbb{A}\} \in \mathsf{Perm}(\mathbb{A})$. We shall define for each set X of nominal structures of interest a function

$$\_ \cdot \_ : \mathsf{Perm}(\mathbb{A}) \times X \to X$$

such that $\pi \cdot (\pi' \cdot x) = (\pi \circ \pi') \cdot x$ and $\mathsf{id} \cdot x = x$, for all $x \in X$ and $\pi, \pi' \in \mathsf{Perm}(\mathbb{A})$. 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 $S \subseteq \mathbb{A}$ such that $\pi \cdot x = x$ whenever $\forall a \in S. \pi(a) = a$. For example, $\mathbb{A}$ is a nominal set with action $\pi \cdot a = \pi(a)$, and so is $\mathcal{P}_{\mathsf{fn}}(\mathbb{A})$ with action $\pi \cdot S = \{\pi(a) \mid a \in S\}$. Also, 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^* with action $\pi \cdot x_1 \dots x_n = (\pi \cdot x_1) \dots (\pi \cdot x_n)$. Also, if X is a nominal set then so is the set $\bigcup_{n \in \omega}([n] \to X)$ with action $\pi \cdot f = \{(i, \pi \cdot x) \mid (i, x) \in f\}$.

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

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

where $a'{=}\pi(a)$, $b'{=}\pi(b)$, $c'{=}\pi(c)$ (note that permutations equally affect bound and free name occurrences). Similarly to X^*, 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 $\pi \in \mathsf{Perm}(\mathbb{A})$, then X' is also a nominal set with the inherited action. Hence, the following set is a nominal set.

$$\hat{K} = \{\, (\sigma, \hat{P}) \mid \sigma \in \bigcup_{n \in \omega} \mathsf{Reg}_n \wedge \hat{P} \in \hat{\Pi} \wedge \mathsf{fn}(\hat{P}) \subseteq \mathsf{img}(\sigma) \,\} \quad (1)$$

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

$$O(X) = \{O(x) \,|\, x \in X\} \quad \textit{where} \quad O(x) = \{\pi{\cdot}x \,|\, \pi \in \mathsf{Perm}(\mathbb{A})\}.$$

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 π-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 $O(\hat{K})$ be the set of processes-in-context. Each such $O(\sigma, \hat{P})$ is written $\sigma \vdash \hat{P}$.

Since $\sigma \vdash \hat{P} = \pi{\cdot}\sigma \vdash \pi{\cdot}\hat{P}$, for any permutation π, what matters in $\sigma \vdash \hat{P}$ is not the specific names occurring in σ or P, but only their index in σ. For example,

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

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

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

$$\alpha \ ::= \ i \mid i^{\bullet} \mid i^{\circledast} \mid \tau \mid \bar{i}j \mid \bar{i}j^{\circledast} \mid ij \mid ij^{\bullet}$$

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

Note that $\sigma \vdash \hat{P} \xrightarrow{\ell} \sigma' \vdash \hat{P}'$ implies $|\sigma| = |\sigma'|$. Some further remarks on reduction:

- Transitions restricted to Π use only τ and double labels, i.e. from $\{\, \bar{i}j, \bar{i}j^{\circledast}, ij, ij^{\bullet} \mid i, j \in \omega \,\}$.
- Inputs are decomposed as known inputs (Inp2a) and locally fresh ones (Inp2b), and are therefore finitely branching. The side-conditions impose that, whenever $\sigma \vdash P_{\mathsf{inp}} \xrightarrow{i^{\bullet}} \sigma' \vdash P$, then $\sigma' = \sigma[i \mapsto a]$, $a \notin \mathsf{img}(\sigma)$ and i is the least index such that $\sigma(i) \notin \mathsf{fn}(P)$.[6] 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 $a \in \mathbb{A}$, let $\sigma_a = \{(1, a)\}$ and

$$P_a = \nu b.\, p(ab) \quad \textit{with definition} \quad p(ab) = \bar{a}b.\, \nu c.\, p(bc)\,.$$

In the π-calculus, P_a induces an infinitely-branching, infinite-path transition graph:

$$\begin{tikzcd} & \vdots & \vdots & \vdots \\ P_a \arrow[r, "\bar{a}(b)"] \arrow[ur, "\bar{a}(b')"] & P_b \arrow[r, "\bar{b}(c)"] \arrow[ur, "\bar{b}(c')"] & P_c \arrow[r, "\bar{c}(d)"] \arrow[ur, "\bar{c}(d')"] & \cdots \end{tikzcd}$$

In the extended calculus, P_a induces the following transition graph,

$$\sigma_a \vdash P_a \xrightarrow{1} \sigma_a \vdash \nu b.\, b.\, \nu c.\, p(bc) \xrightarrow{1^{\circledast}} \sigma_b \vdash P_b \xrightarrow{1} \cdots$$

which is economic by branching once at each step. In fact, setting $P_{\mathsf{out}} = \nu b.\, b.\, \nu c.\, p(bc)$, and since $\sigma_a \vdash P_a = \sigma_b \vdash P_b$ and $\sigma_a \vdash P_{\mathsf{out}} = \sigma_b \vdash P_{\mathsf{out}}$ for all $a, b \in \mathbb{A}$, the graph above contains just two nodes:

$$\begin{tikzcd} \sigma_a \vdash P_a \arrow[r, bend left, "1"] & \sigma_a \vdash P_{\mathsf{out}} \arrow[l, bend left, "1^{\circledast}"] \end{tikzcd}$$

and using double labels we get simply $\begin{tikzcd} \sigma_a \vdash P_a \arrow[loop right, "11^{\circledast}"] \end{tikzcd}$.

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 σ, σ' be registers, and $\alpha, \hat{\alpha}$ be labels of π and xπ respectively. For all $P, P' \in \Pi$ with $\mathsf{fn}(P) \subseteq \mathsf{img}(\sigma)$:*

- *if $\sigma \vdash P \xrightarrow{\hat{\alpha}} \sigma' \vdash P'$ then $P \xrightarrow{\alpha} P'$,*
- *if $P \xrightarrow{\alpha} P'$ then $\sigma \vdash P \xrightarrow{\hat{\alpha}} \sigma' \vdash P'$;*

where either $\hat{\alpha} = \alpha = \tau$ and $\sigma = \sigma'$; or $\hat{\alpha} = \bar{i}j/ij$, $\alpha = \bar{a}b/ab$, $\sigma(i) = a$, $\sigma(j) = b$ and $\sigma' = \sigma$; or $\hat{\alpha} = \bar{i}j^{\circledast}/ij^{\bullet}$, $\alpha = \bar{a}(b)/ab$, $\sigma(i) = a$, $\sigma' = \sigma[j \mapsto b]$ and $j = \min\{j \mid \sigma(j) \notin \mathsf{fn}(P')\}$.

There is a straightforward passage from the xπ-calculus to FRA's: states are taken from $O(\hat{K})$, states from $O(K)$ 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

$$R \subseteq O(K) \times ([n] \rightleftharpoons [n]) \times O(K)$$

[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.

$$\textsc{Inp1}\,\frac{}{\sigma \vdash a(b).P \xrightarrow{i} \sigma \vdash (b).P}\,{\scriptstyle \sigma(i)=a} \qquad \textsc{Match}\,\frac{\sigma \vdash P \xrightarrow{\alpha} \sigma \vdash \hat{P}'}{\sigma \vdash [a=a]P \xrightarrow{\alpha} \sigma \vdash \hat{P}'} \qquad \textsc{Sum}\,\frac{\sigma \vdash P \xrightarrow{\alpha} \sigma \vdash \hat{P}'}{\sigma \vdash P{+}Q \xrightarrow{\alpha} \sigma \vdash \hat{P}'}$$

$$\textsc{Inp2a}\,\frac{}{\sigma \vdash (b).P \xrightarrow{i} \sigma \vdash P\{a/b\}}\,{\scriptstyle \sigma(i)=a} \qquad \textsc{Inp2b}\,\frac{}{\sigma \vdash (b).P \xrightarrow{i^\bullet} \sigma[i \mapsto b] \vdash P}\,{\scriptstyle i=\min\{i \mid \sigma(i)\notin \mathsf{fn}(P)\}}$$

$$\textsc{Out1}\,\frac{}{\sigma \vdash \bar{a}b.P \xrightarrow{i} \sigma \vdash b.P}\,{\scriptstyle \sigma(i)=a} \qquad \textsc{Out2}\,\frac{}{\sigma \vdash b.P \xrightarrow{i} \sigma \vdash P}\,{\scriptstyle \sigma(i)=b} \qquad \textsc{Rec}\,\frac{\sigma \vdash P\{\vec{a}/\vec{b}\} \xrightarrow{\alpha} \sigma \vdash \hat{P}'}{\sigma \vdash p(\vec{a}) \xrightarrow{\alpha} \sigma \vdash \hat{P}'}\,{\scriptstyle p(\vec{b})=P}$$

$$\textsc{Res}\,\frac{(\sigma+a) \vdash \hat{P} \xrightarrow{\alpha} (\sigma'+a) \vdash \hat{P}'}{\sigma \vdash \nu a.\hat{P} \xrightarrow{\alpha} \sigma' \vdash \nu a.\hat{P}'}\,{\scriptstyle \alpha\neq(|\sigma|+1)} \qquad \textsc{Open}\,\frac{\sigma[i \mapsto a] \vdash P_{\mathsf{out}} \xrightarrow{i} \sigma[i \mapsto a] \vdash P}{\sigma \vdash \nu a.P_{\mathsf{out}} \xrightarrow{i^\circledast} \sigma[i \mapsto a] \vdash P}\,{\scriptstyle i=\min\{i \mid \sigma(i)\notin \mathsf{fn}(P)\}}$$

$$\textsc{Par1}\,\frac{\sigma \vdash \hat{P} \xrightarrow{\alpha} \sigma \vdash \hat{P}'}{\sigma \vdash \hat{P} \,|\, Q \xrightarrow{\alpha} \sigma \vdash \hat{P}' \,|\, Q}\,{\scriptstyle \alpha=\, i/\tau} \qquad \textsc{Par2}\,\frac{\sigma \vdash \hat{P} \xrightarrow{i^\bullet/i^\circledast} \sigma[i \mapsto b] \vdash P'}{\sigma \vdash \hat{P} \,|\, Q \xrightarrow{j^\bullet/j^\circledast} \sigma[j \mapsto b] \vdash P' \,|\, Q}\,{\scriptstyle j=\min\{j \mid \sigma(j)\notin \mathsf{fn}(P',Q)\}}$$

$$\textsc{Comm}\,\frac{\sigma \vdash P \xrightarrow{\bar{i}j} \sigma \vdash P' \qquad \sigma \vdash Q \xrightarrow{ij} \sigma \vdash Q'}{\sigma \vdash P \,|\, Q \xrightarrow{\tau} \sigma \vdash P' \,|\, Q'} \qquad \textsc{Close}\,\frac{(\sharp+\sigma) \vdash P \xrightarrow{\bar{i}1^\circledast} (b+\sigma) \vdash P' \qquad (\sharp+\sigma) \vdash Q \xrightarrow{i1^\bullet} (b+\sigma) \vdash Q'}{\sigma \vdash P \,|\, Q \xrightarrow{\tau} \sigma \vdash \nu b.(P' \,|\, Q')}$$

$$\textsc{DblOut}\,\frac{\sigma \vdash P \xrightarrow{i} \sigma \vdash P_{\mathsf{out}} \xrightarrow{j/j^\circledast} \sigma' \vdash P'}{\sigma \vdash P \xrightarrow{\bar{i}j/\bar{i}j^\circledast} \sigma' \vdash P'} \qquad \textsc{DblInp}\,\frac{\sigma \vdash P \xrightarrow{i} \sigma \vdash P_{\mathsf{inp}} \xrightarrow{j/j^\bullet} \sigma' \vdash P'}{\sigma \vdash P \xrightarrow{ij/ij^\bullet} \sigma' \vdash P'}$$

Table 1. The transition relation for the xπ-calculus (symmetric counterparts of SUM, PAR, COMM, CLOSE omitted[8]).

such that if $(\sigma_1 \vdash P_1, \rho, \sigma_2 \vdash P_2) \in R$ then $\sigma_1, \sigma_2 \in \mathsf{Reg}_n$ and $\sigma_1 \vdash P_1 \xrightarrow{\alpha} \sigma_1' \vdash P_1'$ implies that $\sigma_2 \vdash P_2 \xrightarrow{\alpha'} \sigma_2' \vdash P_2'$ for some $(\sigma_1' \vdash P_1', \rho', \sigma_2' \vdash P_2') \in R$ such that one of the following is the case, with $i \in \mathsf{dom}(\rho)$:

- $\alpha = \alpha' = \tau$ and $\rho' = \rho$;
- $\alpha = ij$, $j \in \mathsf{dom}(\rho)$, $\alpha' = \rho(i)\rho(j)$ and $\rho' = \rho$;
- $\alpha = ij$, $j \notin \mathsf{dom}(\rho)$, $\alpha' = \rho(i)k^\bullet$ and $\rho' = \rho[j \leftrightarrow k]$;
- $\alpha = ij^\bullet$, $\alpha' = \rho(i)k^\bullet$, $\rho' = \rho[j \leftrightarrow k]$ and,

  for all $k' \in S_2(\rho) \setminus \mathsf{img}(\rho)$, $\sigma_2 \vdash P_2 \xrightarrow{\rho(i)k'} \sigma_2 \vdash P_2'$ for some $(\sigma_1' \vdash P_1', \rho[j \leftrightarrow k'], \sigma_2 \vdash P_2') \in R$;
- $\alpha = \bar{i}j$, $j \in \mathsf{dom}(\rho)$, $\alpha' = \overline{\rho(i)}\rho(j)$ and $\rho' = \rho$;
- $\alpha = \bar{i}j^\circledast$, $\alpha' = \overline{\rho(i)}k^\circledast$ and $\rho' = \rho[j \leftrightarrow k]$.

R is called an n-***bisimulation*** if both R and R^{-1} are n-simulations. P_1 and P_2 are n-***bisimilar***, written $P_1 \overset{n}{\sim} P_2$, if there is an n-bisimulation R containing $(\sigma_{01} \vdash P_1, \sigma_{01} \leftrightarrow \sigma_{02}, \sigma_{02} \vdash P_2)$, for some σ_{01}, σ_{02} with $\mathsf{img}(\sigma_{01}) = \mathsf{fn}(P_1), \mathsf{img}(\sigma_{02}) = \mathsf{fn}(P_2)$.

We say that a process is n-*contained* if all its descendants have less than n free names.

Proposition 34. *For all n-contained P, Q, $P \overset{\pi}{\sim} Q$ iff $P \overset{n}{\sim} Q$.*

Proof. The proof proceeds by showing that if R is a simulation for the π-calculus then

$$R' = \{\, (\sigma_1 \vdash P_1, \rho, \sigma_2 \vdash P_2) \mid (P_1, P_2) \in R \wedge \rho = \sigma_1 \leftrightarrow \sigma_2 \,\}$$

with P_1, P_2 n-contained and $\sigma_1, \sigma_2 \in \mathsf{Reg}_n$ is an n-simulation and, conversely, if R is an n-simulation then

$$R' = \{\, (P_1, P_2) \mid \exists \sigma_1, \sigma_2.\, (\sigma_1 \vdash P_1, \sigma_1 \leftrightarrow \sigma_2, \sigma_2 \vdash P_2) \in R \,\}$$

with P_1, P_2 n-contained is a simulation for π. ☐

[8] note: $\sigma{+}v = \sigma \cup \{(|\sigma|{+}1, v)\}$, $v{+}\sigma = \{(1, v)\} \cup \{(i{+}1, v') \mid (i, v') \in \sigma\}$.

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

Proposition 35. *If $P_0 \in \Pi$ has finite control and all its descendants are ν-strict, then there are some $M \in \omega$, $\sigma_0 \in \mathsf{Reg}_M$ 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 $\sigma \vdash P \xrightarrow{\alpha} \sigma' \vdash P'$ then $(\sigma' \vdash P') \in S$.*

Proof. Suppose (WLOG) that P_0 invokes definitions $p_i(\vec{a}_i) = P_i$, $i \in [N]$ 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 **0**'s, p's and names, free or bound (but not binding): e.g. $|\bar{a}b.P| = 2 + |P|$, $|a(b).P| = 1 + |P|$, $|\nu a.P| = |P|$, $|p(\vec{a})| = 1 + |\vec{a}|$ and $|\mathbf{0}| = 1$. 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 ν-strict, their number of ν-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 Reg_M, the number of the resulting processes-in-context is bounded relatively to M. We collect all these in S. ☐

Corollary 36. *Bisimilarity is decidable in Π when restricted to processes with finite control.*

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

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 π-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 with globally fresh transitions, and this is clearly depicted in the partition $\hat{\Pi} = \Pi_{\mathsf{inp}} \uplus \Pi_{\mathsf{out}} \uplus \Pi$. A similar observation applies to FRA's describing Reduced ML [19]. There, the states are partitioned in P-states (for Proponent/Program) and O-states (for Opponent/Environment); only P-states are allowed to perform globally fresh transitions, and only O-states can do locally fresh ones. Intuitively, the only notion of freshness that can be observed on the program's side is local freshness, whereas the environment should be assumed to have the memory needed in order to observe global freshness. These observations suggest that a notion of *polarised FRA*, where states are partitioned as above, is relevant and should be further pursued. In the polarised setting, symbolic bisimulations are simplified as there is no longer need for an h component (cf. Definitions 25 and 33).

A potential criticism towards FRA's concerns the fact that they fail to satisfy closure under concatenation and Kleene star (cf. Section 6). We find these non-closure results rather expected as FRA's are history-sensitive machines. On the other hand, FRA's seem to be closed under the *nominal versions* of concatenation and Kleene star, as recently introduced by Gabbay and Ciancia [9]. The precise connections between FRA's and regular languages with name-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

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

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

We set $Q' = Q \times ([n+1] \xrightarrow{\cong} [n+1])$ and write elements of Q' as (q, π). Moreover, $q_0' = (q_0, \mathsf{id})$, $\sigma_0' = \sigma_0[n{+}1 \mapsto \sharp]$ and $F' = \{(q,\pi) \mid q \in F\}$. Finally:

$$\begin{aligned}\delta' = {} & \{\, ((q_1,\pi), \ell, (q_2,\pi)) \mid \ell \in \mathbb{C} \wedge (q_1,\ell,q_2) \in \delta \,\}\\ & \cup \{\, (q_1', (\{\pi(n{+}1)\}, \{\pi(i), \pi(n{+}1)\}), (q_2,\pi)) \mid (q_1, i, q_2) \in \delta \,\}\\ & \cup \{\, (q_1', (\{\pi(n{+}1)\}, \{\pi(n{+}1)\}), (q_2,\pi')) \mid (q_1, i^{\bullet}, q_2) \in \delta \,\}\\ & \cup \{\, (q_1', (\{\pi(n{+}1)\}, \{\pi(n{+}1)\}, [n]), (q_2,\pi')) \mid (q_1, i^{\circledast}, q_2) \in \delta \,\}\end{aligned}$$

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

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

$\square$

Proof of Lemma 20. Let $\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F\rangle$ and construct $\mathcal{B} = \langle Q', q_0', \sigma_0', \delta', F'\rangle$ as follows. The idea is to keep in the extra memory registers of $\mathcal{B}$ a copy of the initial configuration σ_0 which is never touched by assignments. Thus, whenever $\mathcal{A}$ wants to make a transition with label (S,T,A), $\mathcal{B}$ will simulate it by a transition $(S,S,[n])$ and transitions of the form $(S, T \cup T_a)$ where $T_a \subseteq \{n{+}1, ..., 2n\}$, $a \in \sigma_0([n] \setminus A)$ and a is not in the history. In order to accomplish this we need to enrich states with information regarding whether the names in $\mathsf{img}(\sigma_0)$ appear in the history. Therefore, we set $Q' = Q \times \mathcal{P}(\mathsf{img}(\sigma_0))$, $q_0' = (q_0, \emptyset)$, $\sigma_0' = \sigma_0 + \sigma_0$, $F' = \{(q,I) \mid q \in F\}$ and:

$$\begin{aligned}\delta' = {} & \{\, ((q,I), \ell, (q',I)) \mid \ell \in \mathbb{C} \wedge (q,\ell,q') \in \delta \,\}\\ & \cup \{\, ((q,I), (S,T), (q',I)) \mid (q,(S,T),q') \in \delta \,\}\\ & \cup \{\, ((q,I), (S, T \cup T_a), (q', I \cup \{a\})) \mid (q,(S,T),q') \in \delta \,\}\\ & \cup \{\, ((q,I), (S,S,[n]), (q',I)) \mid (q,(S,S,A),q') \in \delta \,\}\\ & \cup \{\, ((q,I), (S, T \cup T_{a'}), (q', I \cup \{a'\})) \mid (q,(S,T,A),q') \in \delta \,\}\end{aligned}$$

where $a \in \mathsf{img}(\sigma_0)$, $T_a = \{(n+i) \in [2n] \mid \sigma_0(i) = a\}$, $a' \in \sigma_0([n] \setminus A) \setminus I$, and $T_{a'}$ as T_a. We can check that

$$R = \{\, ((q,\sigma,H), ((q,I),\sigma',H)) \mid I = H \cap \mathsf{img}(\sigma_0) \wedge \sigma' = \sigma + \sigma_0 \,\}$$

is a bisimulation and therefore that $\mathcal{A} \sim \mathcal{B}$. $\square$

Proof of Lemma 21. Let $\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F\rangle$ and construct $\mathcal{B} = \langle Q', q_0', \sigma_0', \delta', F'\rangle$ by setting $Q' = Q \times ([n] \to [n])$ and selecting f_0, σ_0' such that $\mathsf{img}(\sigma_0) = \mathsf{img}(\sigma_0')$ and $\sigma_0 = \sigma_0' \circ f_0$. Moreover, set $q_0' = (q_0, f_0)$, $F' = \{(q,f) \mid q \in F\}$ and:

$$\begin{aligned}\delta' = {} & \{\, ((q,f), \ell, (q',f)) \mid \ell \in \mathbb{C} \wedge (q,\ell,q') \in \delta \,\}\\ & \cup \{\, ((q,f), i, (q',f')) \mid f(T \setminus S) = \{i\} \wedge (q,(S,T),q') \in \delta \,\}\\ & \cup \{\, ((q,f), i, (q',f')) \mid f^{-1}(i) \subseteq S \wedge (q,(S,S),q') \in \delta \,\}\\ & \cup \{\, ((q,f), i^{\bullet}, (q',f')) \mid f^{-1}(i) \subseteq S \wedge (q,(S,S),q') \in \delta \,\}\\ & \cup \{\, ((q,f), i^{\circledast}, (q',f')) \mid f^{-1}(i) \subseteq S \wedge (q,(S,S,[n]),q') \in \delta \,\}\end{aligned}$$

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

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

and hence $\mathcal{A} \sim \mathcal{B}$. $\square$

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 $(q_1,h,\rho,q_2) \in R$ and suppose that $(q_1,\sigma_1,H) \xrightarrow{a}_{\delta_1} (q_1',\sigma_1',H')$ with $H' = H \cup \{a\}$. We do case analysis on a. Below we write ρ' for $\rho[i \leftrightarrow j]$.

• $a \in \mathsf{img}(\sigma_1) \cap \mathsf{img}(\sigma_2)$, 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, $\rho = \sigma_1 \leftrightarrow \sigma_2$ implies $(i,j) \in \rho$, so $(q_2,j,q_2') \in \delta_2$ for some $(q_1',h,\rho,q_2') \in R$. Thus, $(q_2,\sigma_2,H) \xrightarrow{a}_{\delta_2} (q_2',\sigma_2,H')$ and, noting that $\hat{H}' = \hat{H}$ so $h = \lceil|\hat{H}'|\rceil^n$, we can see that $((q_1',\sigma_1,H'),(q_2',\sigma_2,H')) \in R'$.

• $a \in \mathsf{img}(\sigma_1) \setminus \mathsf{img}(\sigma_2)$, say $a = \sigma_1(i)$. Then, again $(q_1,i,q_1') \in \delta_1$ and $\sigma_1' = \sigma_1$, but $i \in S_1(\rho) \setminus \mathsf{dom}(\rho)$. Thus, $(q_2,j^\bullet,q_2') \in \delta_2$ for some $(q_1',h,\rho',q_2') \in R$. Thus, $(q_2,\sigma_2,H) \xrightarrow{a}_{\delta_2} (q_2',\sigma_2',H')$, $\sigma_2' = \sigma_2[j \mapsto a]$. Noting that $\rho' = \sigma_1 \leftrightarrow \sigma_2'$ and $\hat{H}' = \hat{H}$, we have that $((q_1',\sigma_1,H'),(q_2',\sigma_2',H')) \in R'$.

• $a \in \mathsf{img}(\sigma_2) \setminus \mathsf{img}(\sigma_1)$, say $a = \sigma_2(j)$. Since $a \in \hat{H} \setminus \mathsf{img}(\sigma_1)$, we have some $(q_1,i^\bullet,q_1') \in \delta_1$, and $\sigma_1' = \sigma_1[i \mapsto a]$. Moreover, $j \in S_2(\rho) \setminus \mathsf{img}(\rho)$ and therefore $(q_2,j,q_2' \in \delta_2$ for some $(q_1',h,\rho',q_2') \in R$. Thus, $(q_2,\sigma_2,H) \xrightarrow{a}_{\delta_2} (q_2',\sigma_2,H')$ and we can see that $((q_1',\sigma_1',H'),(q_2',\sigma_2,H')) \in R'$.

• $a \in \hat{H} \setminus (\mathsf{img}(\sigma_2) \cup \mathsf{img}(\sigma_1))$, so $(q_1,i^\bullet,q_1') \in \delta_1$, and $\sigma_1' = \sigma_1[i \mapsto a]$. If $h < n$ then $\|\rho\| = |\mathsf{img}(\sigma_1) \cup \mathsf{img}(\sigma_2)| < |\hat{H}| = h$. Thus, $(q_2,j^\bullet,q_2) \in \delta_2$ for some $(q_1',h,\rho',q_2') \in R$, and so $(q_2,\sigma_2,H) \xrightarrow{a}_{\delta_2} (q_2',\sigma_2',H')$, $\sigma_2' = \sigma_2[j \mapsto a]$. We have $\rho' = \sigma_1' \leftrightarrow \sigma_2'$ and $h = \lceil|\hat{H}'|\rceil^n$, thus $((q_1',\sigma_1',H'),(q_2',\sigma_2',H')) \in R'$.

• $a \notin \hat{H}$ and say transition is due to $(q_1,i^\bullet/i^\circledast,q_1') \in \delta_1$, so $\sigma_1' = \sigma_1[i \mapsto a]$. Then, $(q_2,j^\bullet/j^\circledast,q_2') \in \delta_2$ for some $(q_1',h^{++},\rho',q_2') \in R$, so $(q_2,\sigma_2,H) \xrightarrow{a}_{\delta_2} (q_2',\sigma_2',H')$, $\sigma_2' = \sigma_2[j \mapsto a]$. We have that $h^{++} = \lceil|\hat{H}'|\rceil^n$, so $((q_1',\sigma_1',H'),(q_2',\sigma_2',H')) \in R'$.

Thus, R' is a simulation. If R is a symbolic bisimulation then, by symmetry, R' is a bisimulation. Finally, if $(q_{01},|H_0|,\sigma_{01} \leftrightarrow \sigma_{02},q_{02}) \in R$ then $((q_{01},\sigma_{01},\emptyset),(q_{02},\sigma_{02},\emptyset)) \in R'$. □

Proof of Proposition 27. We check non-constant transitions. Let $(q_1,h,\rho,q_2) \in R'$, due to some $((q_1,\sigma_1,H),(q_2,\sigma_2,H)) \in R$ and suppose that $(q_1,\ell,q_1') \in \delta_1$. We do case analysis on ℓ. Below we write H' for $H \cup \{a\}$, and ρ' for $\rho[i \leftrightarrow j]$.

• If $\ell = i$ then, by closure, $(q_1,\sigma_1,H) \xrightarrow{a}_{\delta_1} (q_1',\sigma_1,H')$ with $a = \sigma_1(i)$, and hence $(q_2,\sigma_2,H) \xrightarrow{a}_{\delta_2} (q_2',\sigma_2',H')$ for some $((q_1',\sigma_1,H'),(q_2',\sigma_2',H')) \in R$. If $i \in \mathsf{dom}(\rho)$, say $(i,j) \in \rho$, then $a \in \mathsf{img}(\sigma_2)$ and it must be $(q_2,j,q_2') \in \delta_2$, $\sigma_2' = \sigma_2$. We can see that $(q_1',h,\rho,q_2') \in R'$. If $i \in S_1(\rho) \setminus \mathsf{dom}(\rho)$ then $a \notin \mathsf{img}(\sigma_2)$ and there is some $(q_2,j^\bullet,q_2') \in \delta_2$, and $\sigma_2' = \sigma_2[j \mapsto a]$. We have that $(q_1',h,\rho',q_2') \in R'$.

• If $\ell = i^\bullet$ then, for each $a \notin \mathsf{img}(\sigma_1)$, $(q_1,\sigma_1,H) \xrightarrow{a}_{\delta_1} (q_1',\sigma_1',H')$, $\sigma_1' = \sigma_1[i \mapsto a]$, and therefore $(q_2,\sigma_2,H) \xrightarrow{a}_{\delta_2} (q_2',\sigma_2',H')$ for some $((q_1',\sigma_1',H'),(q_2',\sigma_2',H')) \in R$.

For any $j \in S_2(\rho) \setminus \mathsf{img}(\rho)$, $\sigma_2(j) \notin \mathsf{img}(\sigma_1)$, so we can take $a = \sigma_2(j)$. Then, we must have $(q_2,j,q_2') \in \delta_2$, $\sigma_2' = \sigma_2$, and we can check that $(q_1',h,\rho',q_2') \in R'$.

If $h = n$ or $\|\rho\| < h$ then we can choose $a \in \hat{H} \setminus (\mathsf{img}(\sigma_1) \cup \mathsf{img}(\sigma_2))$. Thus, we have some $(q_2,j^\bullet,q_2') \in \delta_2$, $\sigma_2' = \sigma_2[j \mapsto a]$. Noting that $\hat{H}' = \hat{H}$ and $\rho' = \sigma_1' \leftrightarrow \sigma_2'$, we get $(q_1',h,\rho',q_2') \in R'$.

Finally, if we choose $a \notin \hat{H}$ then there is some $(q_2,j^\bullet/j^\circledast,q_2') \in \delta_2$, and $\sigma_2' = \sigma_2[j \mapsto a]$. We have that $\rho' = \sigma_1' \leftrightarrow \sigma_2'$ and $\hat{H}' = \hat{H} \uplus \{a\}$, thus $h^{++} = \lceil|\hat{H}'|\rceil^n$. Hence, $(q_1',h^{++},\rho',q_2') \in R'$.

• If $\ell = i^\circledast$ then we work as in the last case above.

Thus, R' is a symbolic simulation. If R is a bisimulation then, by symmetry, R' is a symbolic bisimulation. Finally, if $((q_{01},\sigma_{01},\emptyset),(q_{02},\sigma_{02},\emptyset)) \in R$ then $(q_{01},|H_0|,\sigma_{01} \leftrightarrow \sigma_{02},q_{02}) \in R'$. □

References

[1] S. Abramsky, D. R. Ghica, A. S. Murawski, C.-H. L. Ong, and I. D. B. Stark. Nominal games and full abstraction for the nu-calculus. In *Proc. of LICS '04*, pages 150–159. IEEE Comp. Soc. Press, 2004.

[2] M. Benedikt, C. Ley, and G. Puppis. Minimal memory automata. *Alberto Mendelzon Workshop on Foundations of Databases*, 2010.

[3] N. Benton and V. Koutavas. A mechanized bisimulation for the nu-calculus. Tech. Rep. MSR-TR-2008-129, Microsoft Research, 2008.

[4] R. Bruni, F. Honsell, M. Lenisa, and M. Miculan. Modeling fresh names in the pi-calculus using abstractions. In *Proc. of CMCS '04*, volume 106, pages 25–41. Elsevier, 2004.

[5] S. Delaune, S. Kremer, and M. Ryan. Symbolic bisimulation for the Applied Pi Calculus. In *Proc. of FSTTCS '07*, volume 4855 of *LNCS*, pages 133–145, 2007.

[6] S. Demri and R. Lazic. LTL with the freeze quantifier and register automata. *ACM Trans. Comput. Log.*, 10(3), 2009.

[7] G. L. Ferrari, U. Montanari, and E. Tuosto. Model checking for nominal calculi. In *Proc. of FOSSACS '05*, volume 3441 of *LNCS*, pages 1–24, 2005.

[8] M. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Asp. Comput.*, 13(3-5):341–363, 2002.

[9] M. J. Gabbay and V. Ciancia. Freshness and name-restriction in sets of traces with names. Submitted for publication, 2010.

[10] A. Jeffrey and J. Rathke. Towards a theory of bisimulation for local names. In *LICS*, pages 56–66, 1999.

[11] M. Kaminski and N. Francez. Finite-memory automata. *Theor. Comput. Sci.*, 134(2):329–363, 1994.

[12] J. Laird. A game semantics of local names and good variables. In *Proc. of FOSSACS '04*, volume 2987 of *LNCS*, pages 289–303, 2004.

[13] J. Laird. A fully abstract trace semantics for general references. In *Proc. of ICALP '07*, volume 4596 of *LNCS*, pages 667–679, 2007.

[14] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Inf. Comput.*, 100(1):1–77, 1992.

[15] R. Milner, M. Tofte, and D. Macqueen. *The Definition of Standard ML*. MIT Press, 1997.

[16] U. Montanari and M. Pistore. An introduction to History Dependent Automata. *Electr. Notes Theor. Comput. Sci.*, 10, 1997.

[17] U. Montanari and M. Pistore. Structured coalgebras and minimal HD-automata for the pi-calculus. *Theor. Comput. Sci.*, 340(3):539–576, 2005.

[18] A. S. Murawski and N. Tzevelekos. Full abstraction for Reduced ML. In *Proc. of FOSSACS '09*, volume 5504 of *LNCS*, pages 32–47, 2009.

[19] A. S. Murawski and N. Tzevelekos. Algorithmic nominal game semantics. Submitted for publication, 2010.

[20] R. M. Needham. Names. In S. Mullender, editor, *Distributed systems*, pages 315–327. ACM Press/Addison-Wesley, 1993. 2nd edition.

[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, University of Pisa, 1999.

[23] A. M. Pitts. Nominal logic, a first order theory of names and binding. *Inf. Comput.*, 186(2):165–193, 2003.

[24] A. M. Pitts and I. Stark. Observable properties of higher order functions that dynamically create local names, or: What's *new*? In *Proc. of MFCS '93*, number 711 in LNCS, pages 122–141, 1993.

[25] H. Sakamoto and D. Ikeda. Intractability of decision problems for finite-memory automata. *Theor. Comput. Sci.*, 231(2):297–308, 2000.

[26] D. Sangiorgi and D. Walker. *The pi-calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.

[27] L. Segoufin. Automata and logics for words and trees over an infinite alphabet. In *Proc. of CSL '06*, vol. 4207 of *LNCS*, pages 41–57, 2006.

[28] I. Stark. *Names and Higher-Order Functions*. PhD thesis, University of Cambridge, 1994.