Fresh-Register Automata

Nikos Tzevelekos

University of Oxford
What this talk is about

What is a basic automata-theoretic model of computation with names and fresh-name generation?
Names in computation

let $x$=ref(3) in $f()$; assert($x==3$)
let \( x_1, x_2, x_3, x_4 = \text{ref}() \) in
\[
\lambda x. \text{case } x \text{ of } \begin{align*}
x_1 & \Rightarrow x_2 \\
x_2 & \Rightarrow x_3 \\
x_3 & \Rightarrow x_4 \\
_ & \Rightarrow x_1
\end{align*}
\]
Motivation and related work

What is a basic automata-theoretic model of computation with names and fresh-name generation?

- Programming languages
  - Operational, denotational models of higher-order computation with names
- Process calculi
  - Semantics of mobility
  - History-Dependent automata
Specifications

What is a basic automata-theoretic model of computation with names and fresh-name generation?

- Simple machines of “first principles”
- Infinite alphabet
- Freshness recognition
Finite-memory automata*

Michael Kaminski  
Department of Computer Science, The Hong Kong University of Science and Technology,  
Clear Water Bay, Kowloon, Hong Kong

Nissim Francez  
Department of Computer Science, Technion – Israel Institute of Technology, Technion-city,  
Haifa, 32000, Israel

Communicated by A.R. Meyer  
Received October 1993

Abstract


A model of computation dealing with infinite alphabets is proposed. This model is based on replacing  
the equality test by substitution. It appears to be a natural generalization of the classical Rabin–Scott  
finite-state automata and possesses many of their closure and decision properties. Also, when  
restricted to finite alphabets the model is equivalent to finite-state automata.

1. Introduction

In this paper we introduce a model of computation dealing with infinite alphabets,  
a generalization of the classical Rabin–Scott finite-state automata [6]. In doing so, we  
are aiming towards a very restrictive model, capable of recognizing only the natural  
analogue of regular languages over finite alphabets. In addition, we would like our  
model and the class of languages recognizable by it to have as many of the
Finite-memory automata*

Michael Kaminski
Department of Computer Science, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

Nissim Francez
Department of Computer Science, Technion – Israel Institute of Technology, Technion-city, Haifa, 32000, Israel

Communicated by A.R. Meyer
Received October 1993

Abstract

A model of computation dealing with infinite alphabets is proposed. This model is based on replacing the equality test by substitution. It appears to be a natural generalization of the classical Rabin–Scott finite-state automata and possesses many of their closure and decision properties. Also, when restricted to finite alphabets the model is equivalent to finite-state automata.

1. Introduction

In this paper we introduce a model of computation dealing with infinite alphabets, a generalization of the classical Rabin–Scott finite-state automata [6]. In doing so, we are aiming towards a very restrictive model, capable of recognizing only the natural analog of regular languages over finite alphabets. In addition, we would like our model and the class of languages recognizable by it to give as many of the
Fresh-Register Automata

• FMA's satisfy the specifications:
  • *Simple machines of “first principles”*
  • *Infinite alphabet*

• but not:
  • *Freshness recognition*

• Extend FMA's with transitions for fresh names.
Do names with registers

- Let $\mathbb{A}$ be an infinite set of names
- Let $\mathbb{C}$ be a finite set of constants

- Consider finite-state automata over:

$$L_n = \mathbb{C} \cup \{i, i^\bullet, i^\ast \mid 1 \leq i \leq n\}$$
Do names with registers

- Let $\mathbb{A}$ be an infinite set of names
- Let $\mathbb{C}$ be a finite set of constants

- Consider finite-state automata over:

$$L_n = \mathbb{C} \cup \{ i, i^*, i^\ast \mid 1 \leq i \leq n \}$$

- but operate in reality over:

$$\mathbb{C} \cup \mathbb{A}$$
Definition

- Recall: \( \mathbb{L}_n = \mathbb{C} \cup \{ i, \cdot, \ast \mid 1 \leq i \leq n \} \)

- Define *register assignments* of size \( n \):

\[
\text{Reg}_n = \{ \sigma : \{1, \ldots, n\} \rightarrow \mathbb{A} \mid \forall i \neq j. \sigma(i) \neq \sigma(j) \}
\]
Definition

\[ A = \langle Q, q_0, \sigma_0, \delta, F \rangle \] where:

- \( Q \) is a finite set of states,
- \( q_0 \in Q \) is the initial state,
- \( \delta \subseteq Q \times \mathbb{I}_n \times Q \) is the transition relation,
- \( F \subseteq Q \) is the set of final states.

\[ \mathbb{I}_n = \mathbb{C} \cup \{ i, i^\bullet, i^\circ | i \in [n] \} \]

\[ \text{Reg}_n = \{ \sigma : \{1, \ldots, n\} \rightarrow \mathbb{A} | \forall i \neq j. \sigma(i) \neq \sigma(j) \} \]
Definition

\[ A = \langle Q, q_0, \sigma_0, \delta, F \rangle \text{ where:} \]

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

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

\[ \text{Reg}_n = \{ \sigma : \{1, \ldots, n\} \to \mathbb{A} \mid \forall i \neq j. \sigma(i) \neq \sigma(j) \} \]
Definition

A fresh-register automaton (FRA) of $n$ registers is a quintuple $A = \langle Q, q_0, \sigma_0, \delta, F \rangle$ where:

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

\[
\mathbb{L}_n = \mathbb{C} \cup \{ i, i^\ast, i^\ominus | i \in [n] \}
\]

\[
\text{Reg}_n = \{ \sigma : \{1, \ldots, n\} \rightarrow \mathbb{A} | \forall i \neq j. \sigma(i) \neq \sigma(j) \}
\]
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,
- $q_0 \in Q$ is the initial state,
- $\sigma_0 \in \text{Reg}_n$ is the initial register assignment,
- $\delta \subseteq Q \times \mathbb{N} \times Q$ is the transition relation,
- $F \subseteq Q$ is the set of final states.
Configurations

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,
- \( q_0 \in Q \) is the initial state,
- \( \sigma_0 \in \text{Reg}_n \) is the initial register assignment,
- \( \delta \subseteq Q \times \mathbb{I}_n \times Q \) is the transition relation,
- \( F \subseteq Q \) is the set of final states.

• A configuration is a triple:

\[
(q, \sigma, H) \in Q \times \text{Reg}_n \times \mathcal{P}_{\text{fn}}(\mathcal{A})
\]

- state
- register assignment
- history
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,
- $q_0 \in Q$ is the initial state,
- $\sigma_0 \in \text{Reg}_n$ is the initial register assignment,
- $\delta \subseteq Q \times \mathbb{L}_n \times Q$ is the transition relation,
- $F \subseteq Q$ is the set of final states.

A configuration is a triple:

$$(q, \sigma, H) \in Q \times \text{Reg}_n \times \mathcal{P}_{\text{fn}}(\mathcal{A})$$

- state
- register assignment
- history
Configurations

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

- A configuration is a triple:

\[
(q, \sigma, H) \in Q \times \text{Reg}_n \times \mathcal{P}_\text{fn}(\mathcal{A})
\]

- Semantics: Transition relation on configurations.
Demo: known transitions

$q \xrightarrow{i} q'$
Demo: known transitions

$q$ \rightarrow_{i} q'$

$...a...$

$1i_n$
Demo: known transitions

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

\[ \cdots a \cdots \]

\[ 1 \quad i \quad n \]

\[ \cdots a \cdots \]

\[ 1 \quad i \quad n \]
Demo: locally fresh transitions
Demo: locally fresh transitions
Demo: locally fresh transitions

$q \xrightarrow{i} q'$

$a\ fresh$

\[ \begin{array}{ccc}
... & \cdots & ... \\
1 & i & n \\
\end{array} \]
Demo: *locally fresh* transitions

\[ q \xrightarrow{a} q' \]
Demo: globally fresh transitions

$q \xrightarrow{i^\star} q'$
Demo: *globally fresh* transitions
Demo: globally fresh transitions

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

\[\begin{array}{c}
\vdots \\
1 \\
\vdots \\
i \\
\vdots \\
n \\
\vdots \\
\end{array}\]

\textit{a fresh}
Demo: globally fresh transitions

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

\[ \ldots \quad \ldots \quad \ldots \]

1 \quad i \quad n

\[ \ldots \quad a \quad \ldots \]

1 \quad i \quad n

\textbf{a fresh}
Demo: constant transitions
Demo: constant transitions

In the diagram, we have two states, $q$ and $q'$, connected by a transition labeled $C$. The states $q$ and $q'$ are labeled with ellipses indicating that the actual values are not shown, and the numbers 1 to $n$ are indicated below the states, suggesting that these values are part of the transition or state annotations.
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,
- $q_0 \in Q$ is the initial state,
- $\sigma_0 \in \text{Reg}_n$ is the initial register assignment,
- $\delta \subseteq Q \times \mathbb{L}_n \times Q$ is the transition relation,
- $F \subseteq Q$ is the set of final states.

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

\[ q_0 \]

\[ 1^* \]

\[ \sigma_0 = \{ 1 \mapsto \# \} \]
A name generator

\[ q_0 \]

\[ \sigma_0 = \{1 \mapsto \#\} \]
A name generator

\[ \sigma_0 = \{1 \mapsto \#\} \]

\[ q_0 \]

\[ a_1 \]

#
A name generator

\[ q_0 \xrightarrow{1^*} \]

\[ \sigma_0 = \{1 \mapsto \#\} \]

\[ q_0 \xrightarrow{a_1} q_0 \xrightarrow{a_2} q_0 \]
A name generator

\[ q_0 \xrightarrow{1^*} q_0 \]

\[ \sigma_0 = \{1 \mapsto \#\}\]
A name generator

\[ L_1 \]

\[ q_0 \]

\[ 1^* \]

\[ \sigma_0 = \{1 \mapsto \# \} \]

\[ \mathcal{L}(A) = \{a_1 \cdots a_k \in A^* \mid \forall i \neq j. a_i \neq a_j \} \]

\[ q_0 \rightarrow a_1 \rightarrow q_0 \rightarrow a_2 \rightarrow q_0 \rightarrow a_3 \rightarrow q_0 \]
Another example
Another example

\[ C_1 C_2 C_3 \ldots \]
Another example

\[ C a_1 C a_2 C a_3 \ldots \]
Another example

```
q_0 \rightarrow \# \rightarrow q_1 \rightarrow \# \rightarrow q_2 \rightarrow 1^* \rightarrow q_3 \\
C \rightarrow q_0 \rightarrow C \rightarrow q_1 \rightarrow C \rightarrow q_2 \rightarrow C \rightarrow q_3
```
\[
\lambda d. (\text{let } x = \text{ref}() \text{ in } x) \quad \text{let } x = \text{ref}() \text{ in } \lambda d. x
\]

\[
\begin{array}{c}
q_0 \\
\rightarrow \\
\downarrow \\
C \\
\downarrow \\
q_1 \\
\downarrow \\
1^* \\
\downarrow \\
C \\
\rightarrow \\
\downarrow \\
q_2 \\
\downarrow \\
C \\
\rightarrow \\
\downarrow \\
q_3
\end{array}
\quad
\begin{array}{c}
q_0 \\
\rightarrow \\
\downarrow \\
C \\
\downarrow \\
q_1 \\
\downarrow \\
1^* \\
\downarrow \\
C \\
\rightarrow \\
\downarrow \\
q_2 \\
\downarrow \\
C \\
\rightarrow \\
\downarrow \\
q_3
\end{array}
\]

\[
Ca_1 Ca_2 Ca_3 \ldots
\quad
Ca_1 Ca_1 Ca_1 \ldots
\]
Properties

- Closure under union and intersection.
- Non-closure under concatenation and Kleene star.

E.g. \( L_1 \ast L_1 \) is not FRA-recognisable.
Properties

• Closure under union and intersection.
• Non-closure under concatenation and Kleene star.

E.g. $L_1 \cdot L_1$ is not FRA-recognisable.

Nominal versions of concatenation and Kleene star?
Properties

- Closure under union and intersection.
- Non-closure under concatenation and Kleene star.
  E.g. $L_1 \ast L_1$ is not FRA-recognisable.
- Non-closure under complementation.
  E.g. $\overline{L_1 \ast L_1}$ is FRA-recognisable.
Properties

- Closure under union and intersection.
- Non-closure under concatenation and Kleene star.
  E.g. $L_1 \ast L_1$ is not FRA-recognisable.
- Non-closure under complementation.
  E.g. $\overline{L_1} \ast \overline{L_1}$ is FRA-recognisable.
- Emptiness is decidable.
- Containment, universality are undecidable.
Bisimulations

- Recall: \( \hat{Q} = Q \times \text{Reg}_n \times \mathcal{P}_{fn}(\mathcal{A}) \)
- Let \( \mathcal{A}_1, \mathcal{A}_2 \) be FRA's. Consider relations \( R \subseteq \hat{Q}_1 \times \hat{Q}_2 \)
Bisimulations

- Recall: \( \hat{Q} = Q \times \text{Reg}_n \times \mathcal{P}_{\text{fn}}(A) \)
- Let \( A_1, A_2 \) be FRA's. Consider relations \( R \subseteq \hat{Q}_1 \times \hat{Q}_2 \)
Bisimulations

- Recall: \( \hat{Q} = Q \times \text{Reg}_n \times \mathcal{P}_{fn}(\mathbb{A}) \)

- Let \( \mathcal{A}_1, \mathcal{A}_2 \) be FRA's. Consider relations \( R \subseteq \hat{Q}_1 \times \hat{Q}_2 \)
Bisimulations

• Recall: \( \hat{Q} = Q \times \text{Reg}_n \times \mathcal{P}_{\text{fn}}(\mathbb{A}) \)

• Let \( \mathcal{A}_1, \mathcal{A}_2 \) be FRA's. Consider relations \( R \subseteq \hat{Q}_1 \times \hat{Q}_2 \)
Bisimulations

- Recall: \( \hat{Q} = Q \times \text{Reg}_n \times \mathcal{P}_{\text{fn}}(\mathbb{A}) \)
- Let \( \mathcal{A}_1, \mathcal{A}_2 \) be FRA's. Consider relations \( R \subseteq \hat{Q}_1 \times \hat{Q}_2 \)
Bisimulations

- Recall: $\hat{Q} = Q \times \text{Reg}_n \times \mathcal{P}_{\text{fn}}(A)$
- Let $\mathcal{A}_1, \mathcal{A}_2$ be FRA's. Consider relations $R \subseteq \hat{Q}_1 \times \hat{Q}_2$
Bisimulations

- Recall: $\hat{Q} = Q \times \text{Reg}_n \times \mathcal{P}_{\text{fn}}(\Delta)$
- Let $\mathcal{A}_1, \mathcal{A}_2$ be FRA's. Consider relations $R \subseteq \hat{Q}_1 \times \hat{Q}_2$
Bisimulations

- Recall: \( \hat{Q} = Q \times \text{Reg}_n \times \mathcal{P}_{fn}(\mathcal{A}) \)
- Let \( A_1, A_2 \) be FRA's. Consider relations \( R \subseteq \hat{Q}_1 \times \hat{Q}_2 \)

- Bisimilarity implies language equivalence.

\[ \hat{q}_1 \xrightarrow{\ell'} \hat{q}_1'' \]
\[ \hat{q}_2 \xrightarrow{\ell'} \hat{q}_2'' \]

\[ R \]

\[ \hat{q}_1 \xrightleftharpoons{\ell'} \hat{q}_1'' \]
\[ \hat{q}_2 \xrightleftharpoons{\ell'} \hat{q}_2'' \]
Example

$\sigma_0 = \{1 \mapsto \#\}$

$\sim$

$\sigma_0 = \{1 \mapsto \#\}$
Example

\[ \sigma_0 = \{ 1 \mapsto \# \} \]

\[ R = \{ ((q_0, \sigma_0, \emptyset), (q_0, \sigma_0, \emptyset)) \} \cup \{ ((q_0, \sigma_1, H), (q_1, \sigma_2, H)) \} \]
Symbolic bisimulations

- *Symbolic* reasoning for bisimulations:
  - We can define a finite notion of bisimulation to capture (actual) bisimilarity.

\[
R \subseteq Q_1 \times \{0, \ldots, n_1 + n_2\} \times (n_1 \equiv n_2) \times Q_2
\]
Symbolic bisimulations

- Symbolic reasoning for bisimulations:
  - We can define a finite notion of bisimulation to capture (actual) bisimilarity.

\[ R \subseteq Q_1 \times \{0, \ldots, n_1 + n_2\} \times (n_1 \models n_2) \times Q_2 \]

typed spans: \((S_1, \rho, S_2)\)

<table>
<thead>
<tr>
<th>(\rho: {1, \ldots, n_1} \xrightarrow{=} {1, \ldots, n_2})</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{dom}(\rho) \subseteq S_1 \subseteq {1, \ldots, n_1})</td>
</tr>
<tr>
<td>(\text{img}(\rho) \subseteq S_2 \subseteq {1, \ldots, n_2})</td>
</tr>
</tbody>
</table>
Symbolic bisimulations

- *Symbolic* reasoning for bisimulations:
  - We can define a finite notion of bisimulation to capture (actual) bisimilarity.

\[ R \subseteq Q_1 \times \{0, \ldots, n_1 + n_2\} \times (n_1 \iff n_2) \times Q_2 \]

FRA-bisimilarity is decidable
Results on FRA's

- As language acceptors:
  - Closure under union and intersection.
  - Non-closure under concatenation and Kleene star.
  - Non-closure under complementation.
  - Emptiness is decidable.
  - Containment, universality are undecidable.

- Bisimilarity is decidable by use of symbolic means.
Application: the pi-calculus
Application: the pi-calculus

• Algorithmic description which is:
  • name-free
  • finitely-branching
• Bisimilarity can be captured symbolically
  • In the finitary case: decide bisimilarity
Interesting directions

- *Algorithmic game semantics*
  - e.g. (finitary) Reduced ML
Interesting directions

- *Algorithmic game semantics*
  - e.g. (finitary) Reduced ML

Algorithmic nominal game semantics

A. S. Murawski* and N. Tzevelekos**

1 Department of Computer Science, University of Leicester
University Road, Leicester LE1 7RH, UK
2 Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD, UK

Abstract. We employ automata over infinite alphabets to capture the semantics of a finitary fragment of ML with ground-type references. Our approach is founded on game semantics, which allows us to translate programs into automata in such a way that contextual equivalence is characterized by a finitary notion of bisimilarity. As a corollary, we derive a decidability result for a class of first-order programs, including open ones that contain unspecified first-order procedures.
Interesting directions

- **Algorithmic game semantics**
  - e.g. (finitary) Reduced ML
- Connections to HD-automata
- Nominal notions of concatenation, Kleene closure
- Variations / extensions:
  - Labels/tags (*data words*)
  - Stores
  - Stack (*pushdown*)
Interesting directions

- Algorithmic game semantics
  - e.g. (finitary) Reduced ML
- Connections to HD-automata
- Nominal notions of concatenation, Kleene closure
- Variations / extensions:
  - Labels/tags (*data words*)
  - Stores
  - Stack (*pushdown*)

Thanks!