Fresh-Register Automata

Nikos Tzevelekos

Oxford University Computing Laboratory

What this talk is about

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

Names in computation

```
int mainClass NameTest {
  public void main(String[] args){

    Object A = new Object();
    Object B = new Object();

    System.out.println(A.equals(B));
}}
```

Names in computation

```
new x=3 in f(); assert(x==3)
```

Names in computation

```
new x1,x2,x3,x4,x5 in
    fn x => case x of x1 => x2
                    | x2 => x3
                    | x3 => x4
                    | x4 => x5
                    | _  => x1
```

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
  - Nominal game semantics

Names in computation (II)

$$P(a) = \nu b.\ \overline{a}b.\ P(b)$$

Names in computation (II)

$$P(a) = \nu b.\ \bar{a}b.\ P(b)$$

$P(a)$

Names in computation (II)

$$P(a) = \nu b.\ \bar{a}b.\ P(b)$$

$$P(a) \xrightarrow{\bar{a}b} P(b)$$

Names in computation (II)

$$P(a) = \nu b.\ \bar{a}b.\ P(b)$$

$$P(a) \xrightarrow{\bar{a}b} P(b) \xrightarrow{\bar{b}c} P(c)$$

Names in computation (II)

$$P(a) = \nu b.\ \bar{a}b.\ P(b)$$

$$P(a) \xrightarrow{\bar{a}b} P(b) \xrightarrow{\bar{b}c} P(c) \xrightarrow{\bar{c}d} P(d) \cdots\cdots$$

Names in computation (II)

$$P(a) = \nu b.\ \bar{a}b.\ P(b)$$

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

Names in computation (II)

$$P(a) = \nu b.\ \bar{a}b.\ P(b)$$

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

Motivation and related work

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

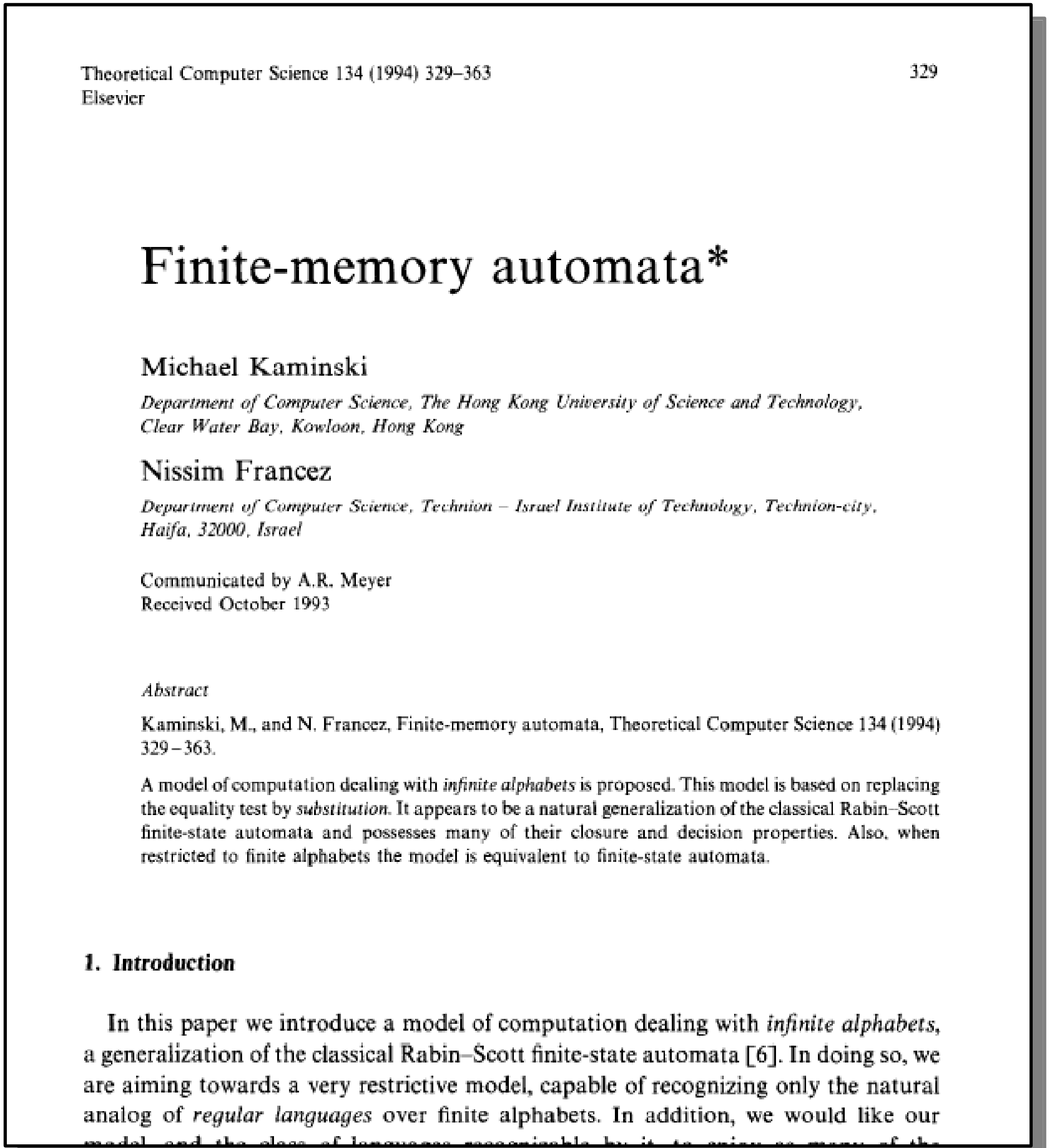
- Programming languages
  - Operational, denotational models of higher-order computation with names
  - Nominal game semantics
- Process calculi
  - Semantics of mobility
  - History-Dependent automata

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

An appealing paradigm



Theoretical Computer Science 134 (1994) 329–363
Elsevier

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

Kaminski, M., and N. Francez, Finite-memory automata, Theoretical Computer Science 134 (1994) 329–363.

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

An appealing paradigm

Theoretical Computer Science 134 (1994) 329–363 329
Elsevier

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

Kaminski, M., and N. Francez, Finite-memory automata, Theoretical Computer Science 134 (1994) 329–363.

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

- *Regular languages* over infinite alphabets
- *Closure* properties
- Finite-state + finite memory
- *Local* reasoning

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:

$$\mathbb{L}_n = \mathbb{C} \cup \{\, i, i^{\bullet}, i^{\circledast} \mid 1 \leq i \leq n \,\}$$

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:

$$\mathbb{L}_n = \mathbb{C} \cup \{\, i, i^{\bullet}, i^{\circledast} \mid 1 \leq i \leq n \,\}$$

- but operate in reality over:

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

Definition

- Recall: $\mathbb{L}_n = \mathbb{C} \cup \{\, i, i^\bullet, i^\circledast \mid 1 \leq i \leq n \,\}$

- Define *register assignments* of size n by:

$$\mathsf{Reg}_n = \{\, \sigma : \{1, \ldots, n\} \to \mathbb{A} \cup \{\sharp\} \\ \mid \forall i \neq j.\, \sigma(i) = \sigma(j) \implies \sigma(i) = \sharp \,\}$$

Definition

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,
- $\sigma_0 \in \mathsf{Reg}_n$ 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.

$$\mathbb{L}_n = \mathbb{C} \cup \{\, i, i^{\bullet}, i^{\circledast} \mid 1 \leq i \leq n \,\}$$

$$\mathsf{Reg}_n = \{\, \sigma : \{1, \ldots, n\} \to \mathbb{A} \cup \{\sharp\} \mid \forall i \neq j.\, \sigma(i) = \sigma(j) \implies \sigma(i) = \sharp \,\}$$

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,
- $\sigma_0 \in \mathsf{Reg}_n$ 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.

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,
- $\sigma_0 \in \mathsf{Reg}_n$ 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.

- A configuration is a triple:

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

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,
- $\sigma_0 \in \mathsf{Reg}_n$ 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.

- A configuration is a triple:

$$(\underbrace{q}_{\text{state}}, \underbrace{\sigma}_{\text{register assignment}}, \underbrace{H}_{\text{history}}) \in \underbrace{Q \times \mathsf{Reg}_n \times \mathcal{P}_{\mathsf{fn}}(\mathbb{A})}_{\hat{Q}}$$

Configurations

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

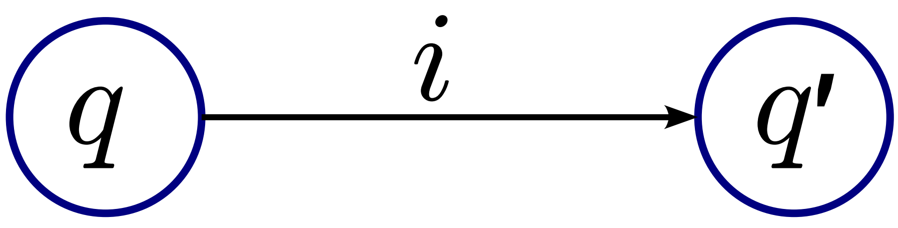
- Q is a finite set of states,
- $q_0 \in Q$ is the initial state,
- $\sigma_0 \in \mathsf{Reg}_n$ is the initial register assignment,
- $\delta \subseteq Q \times \mathbb{L}_n \times Q$ is the transition relation,
- $F \subseteq Q$ is the set of final states.

- A configuration is a triple:

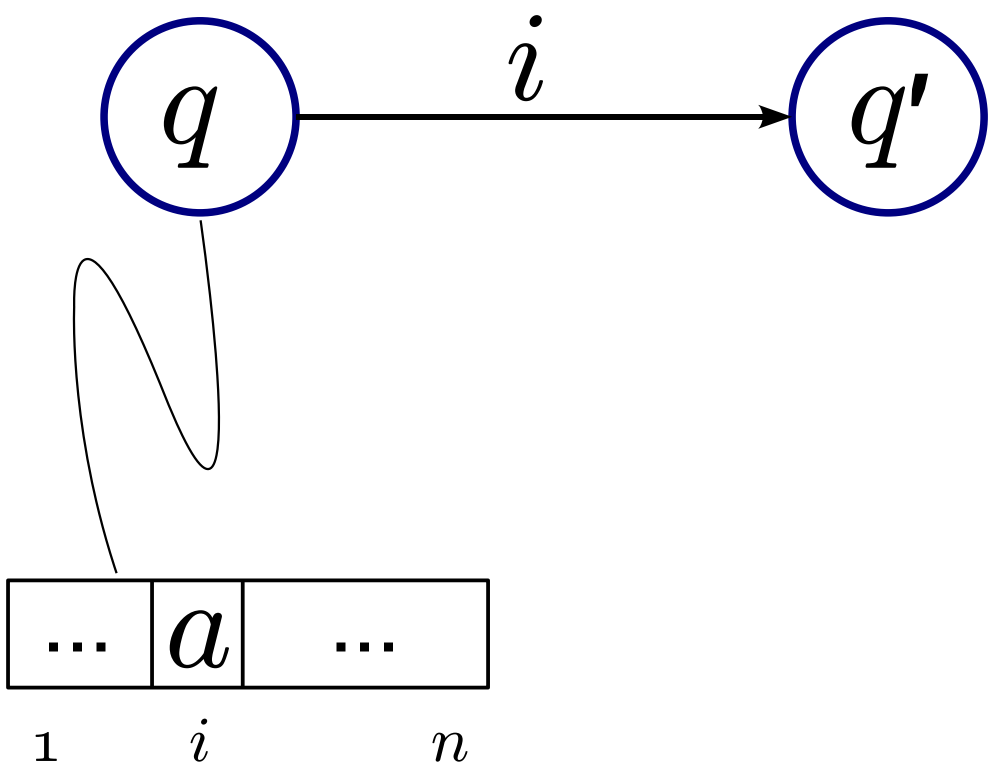
$$(q, \sigma, H) \in Q \times \mathsf{Reg}_n \times \mathcal{P}_{\mathsf{fn}}(\mathbb{A})$$

- Transitions between config's: the 'true' semantics

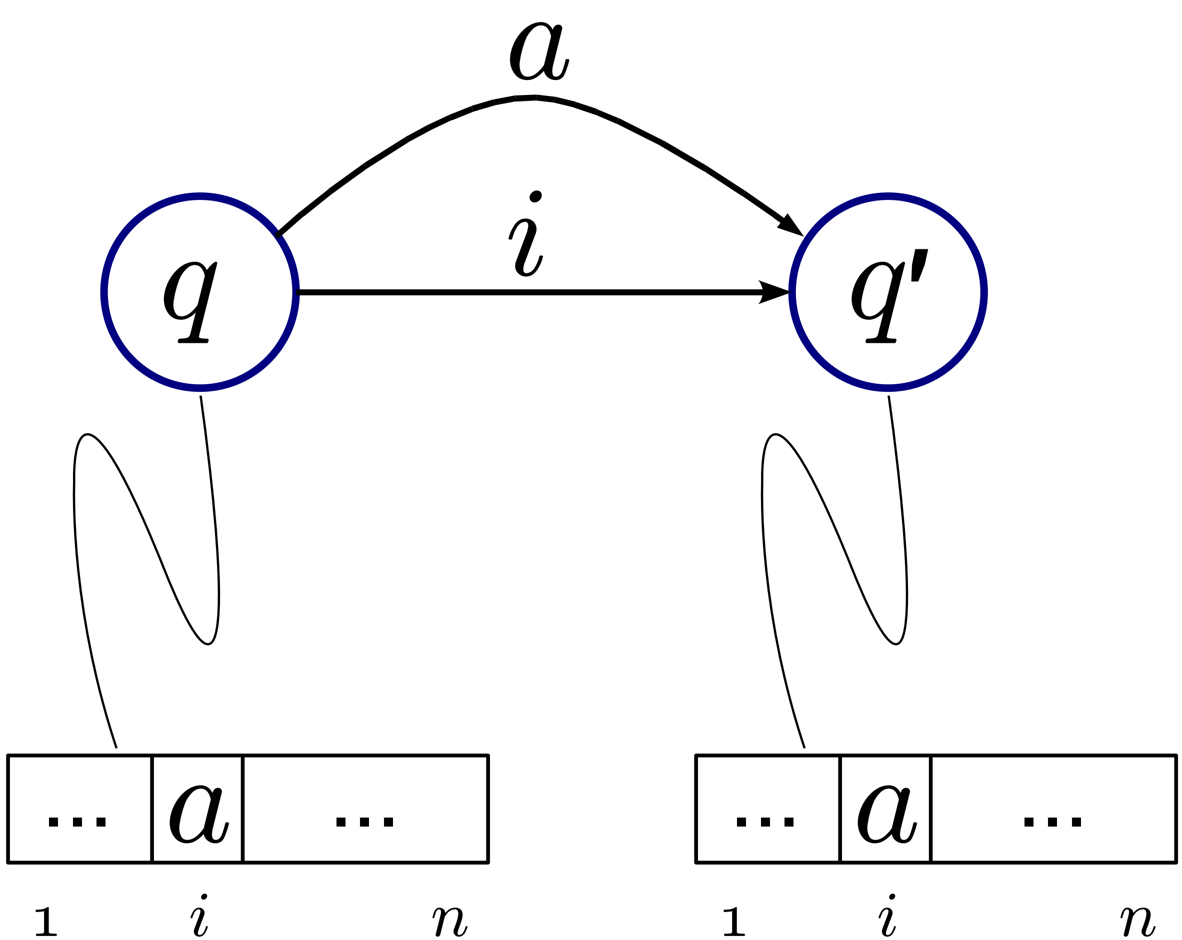
Demo: *known* transitions



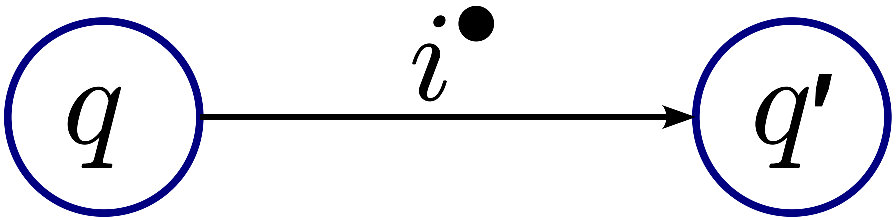
Demo: *known* transitions



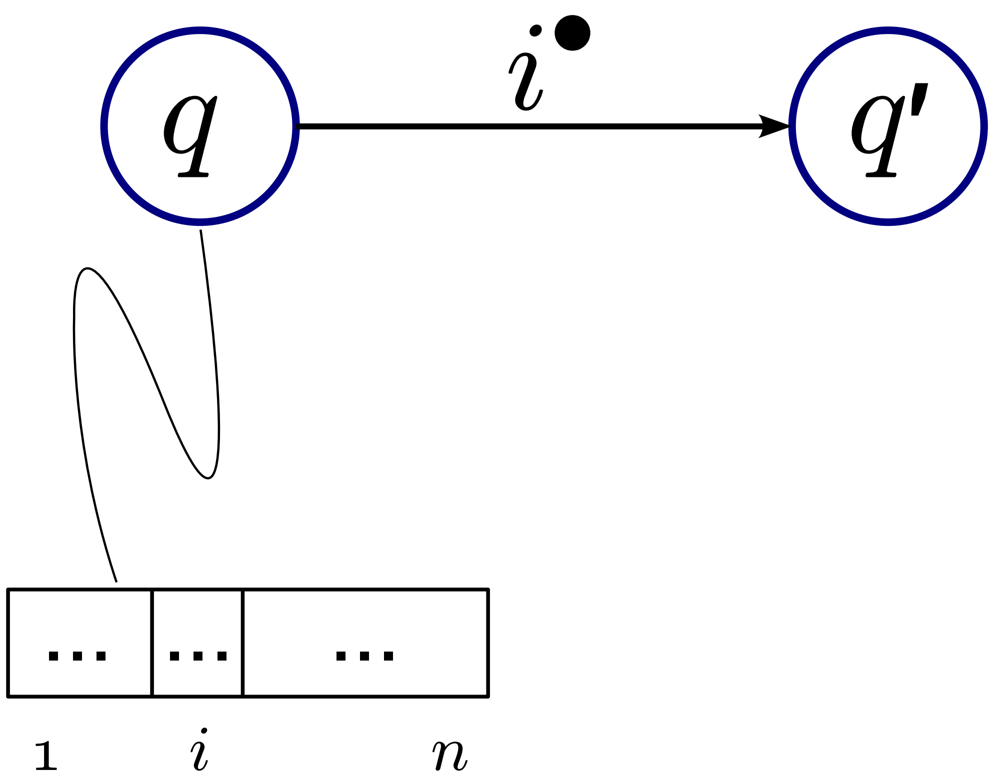
Demo: *known* transitions



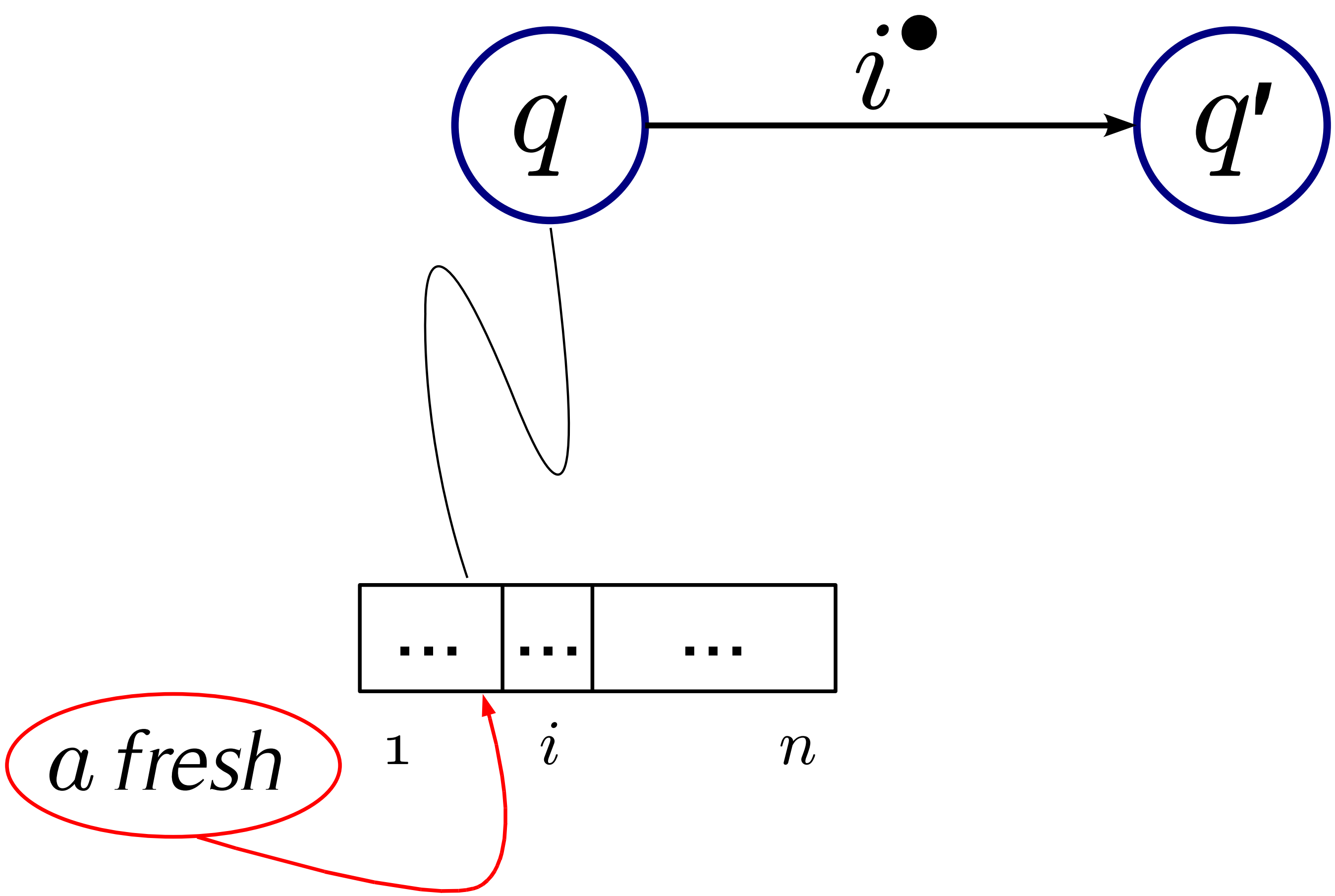
Demo: *locally fresh* transitions



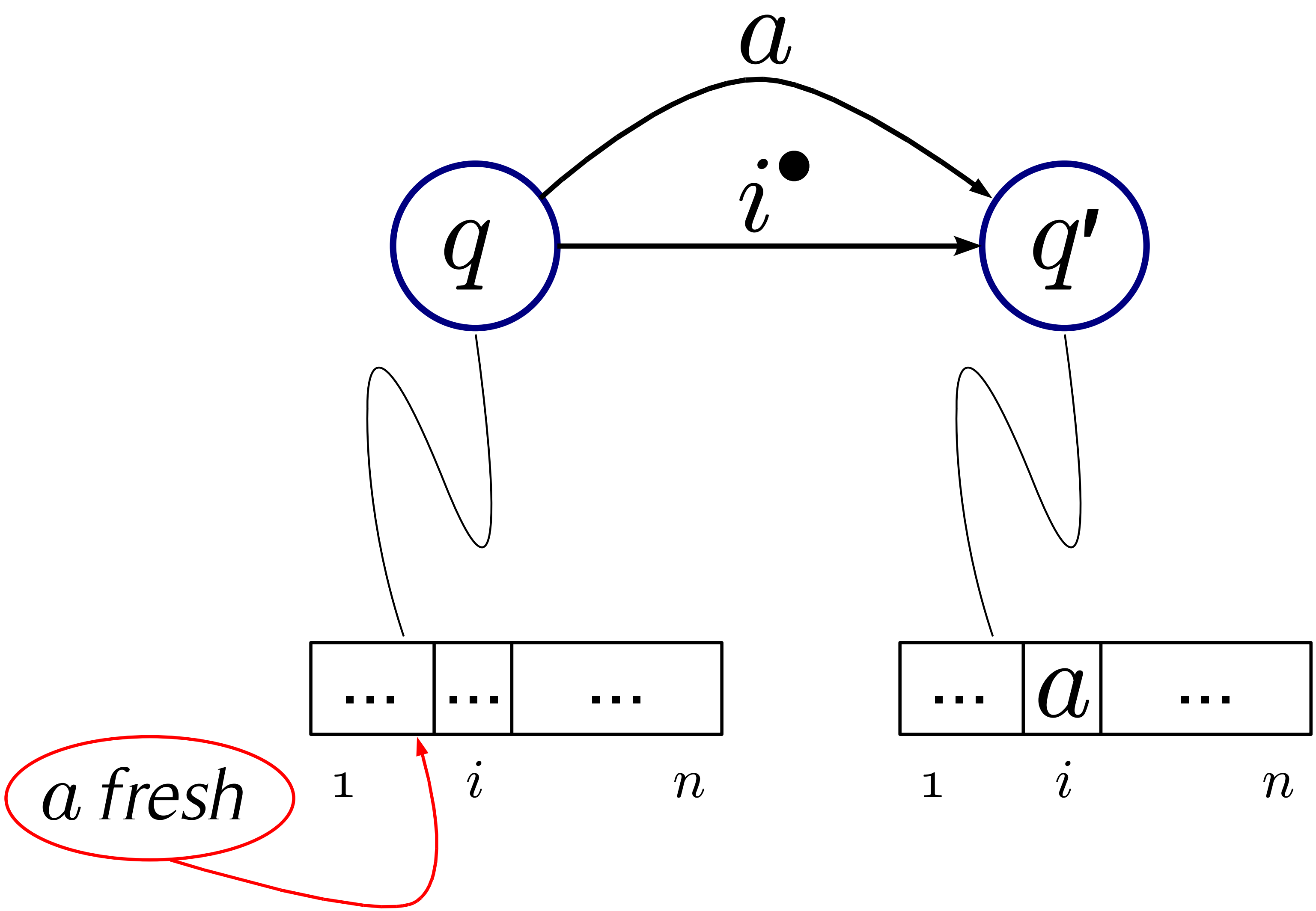
Demo: *locally fresh* transitions



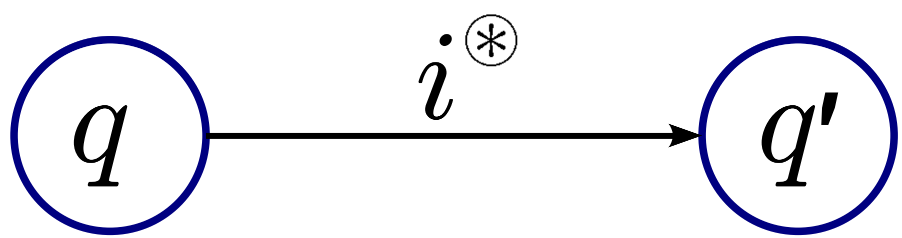
Demo: *locally fresh* transitions



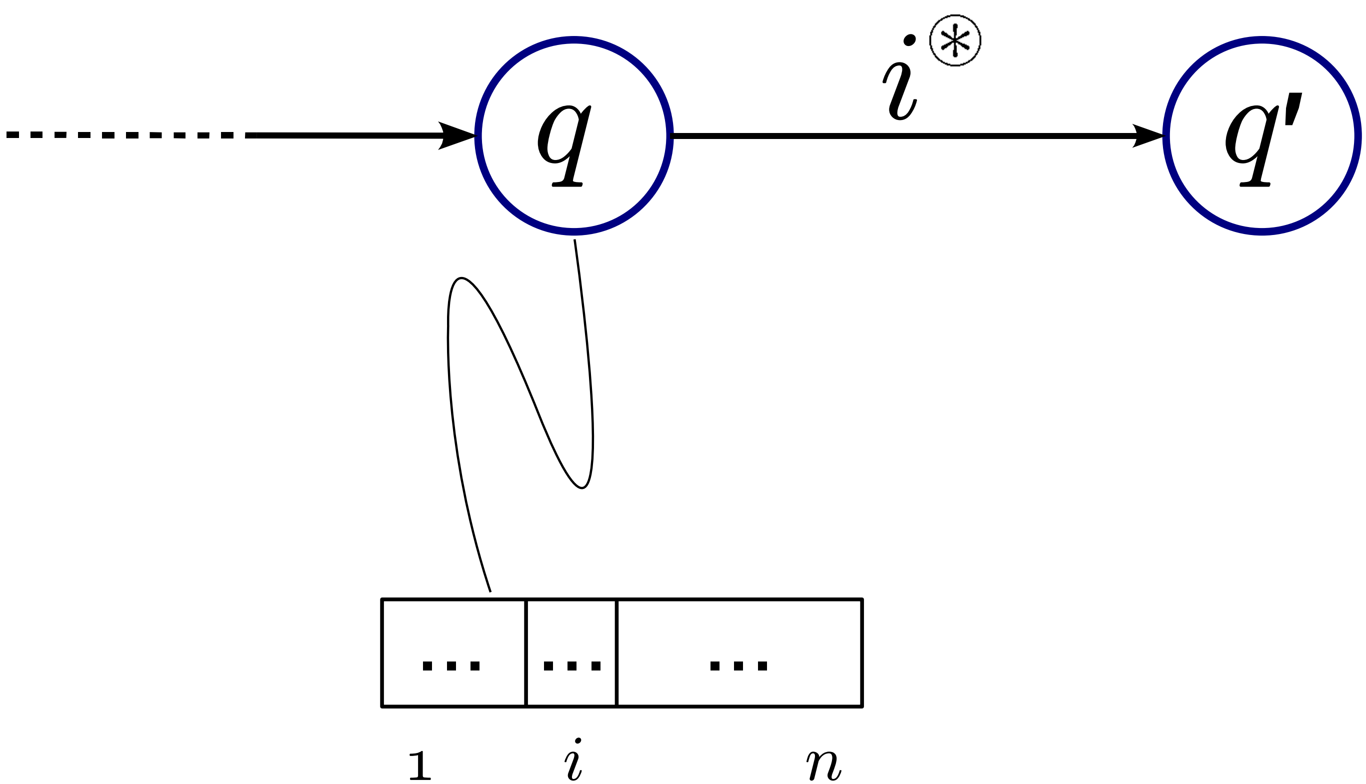
Demo: *locally fresh* transitions



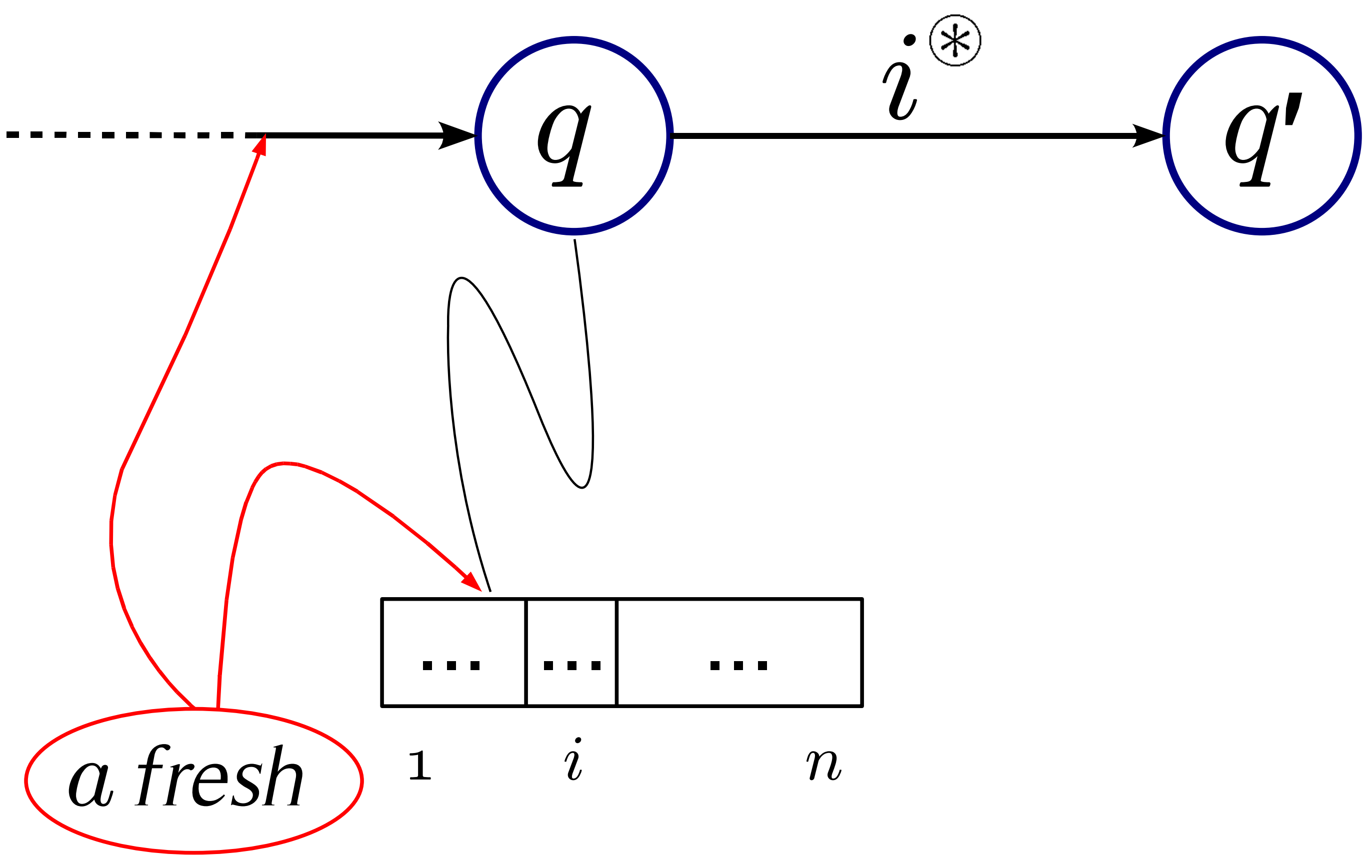
Demo: *globally fresh* transitions



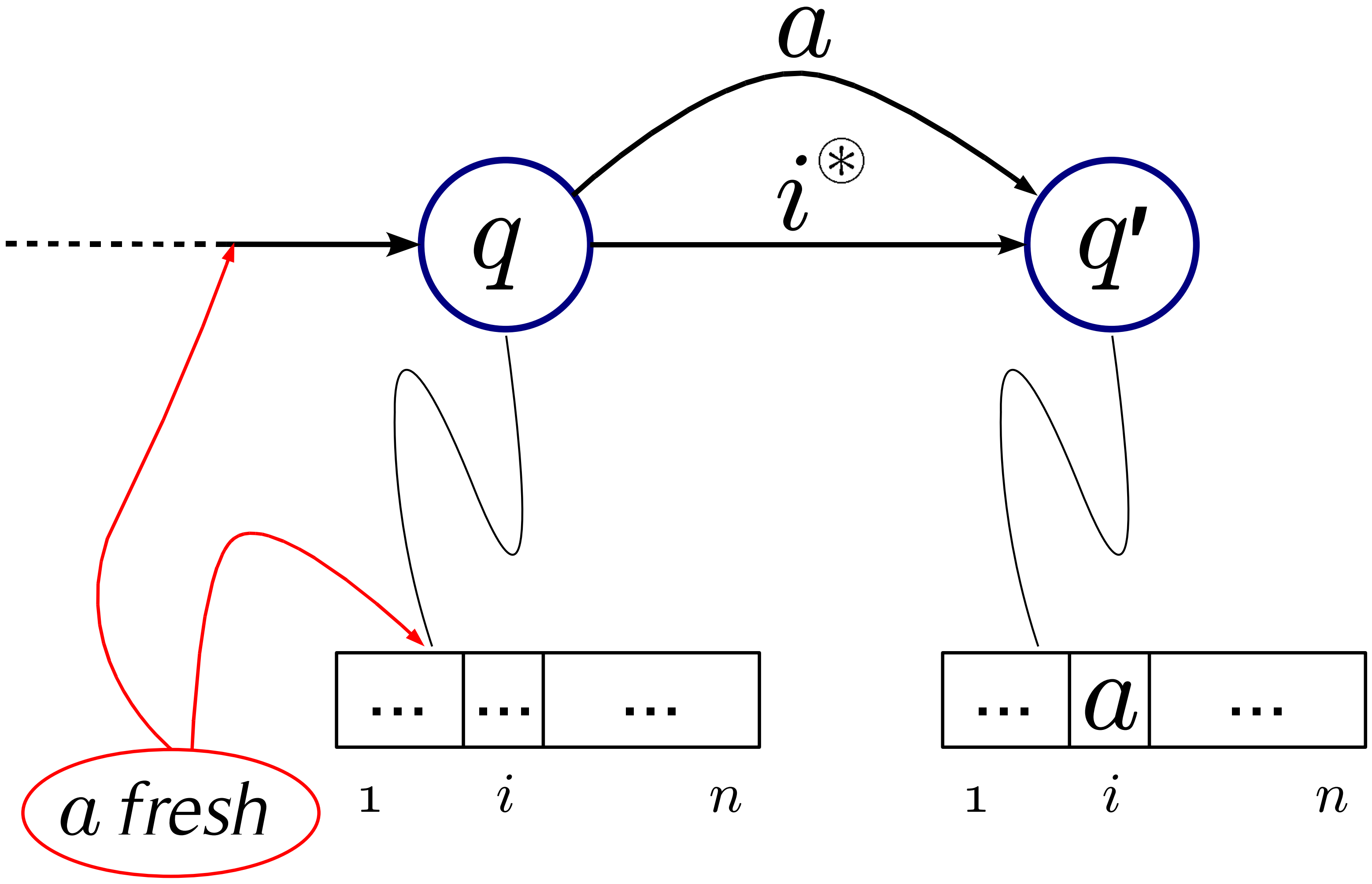
Demo: *globally fresh* transitions



Demo: *globally fresh* transitions

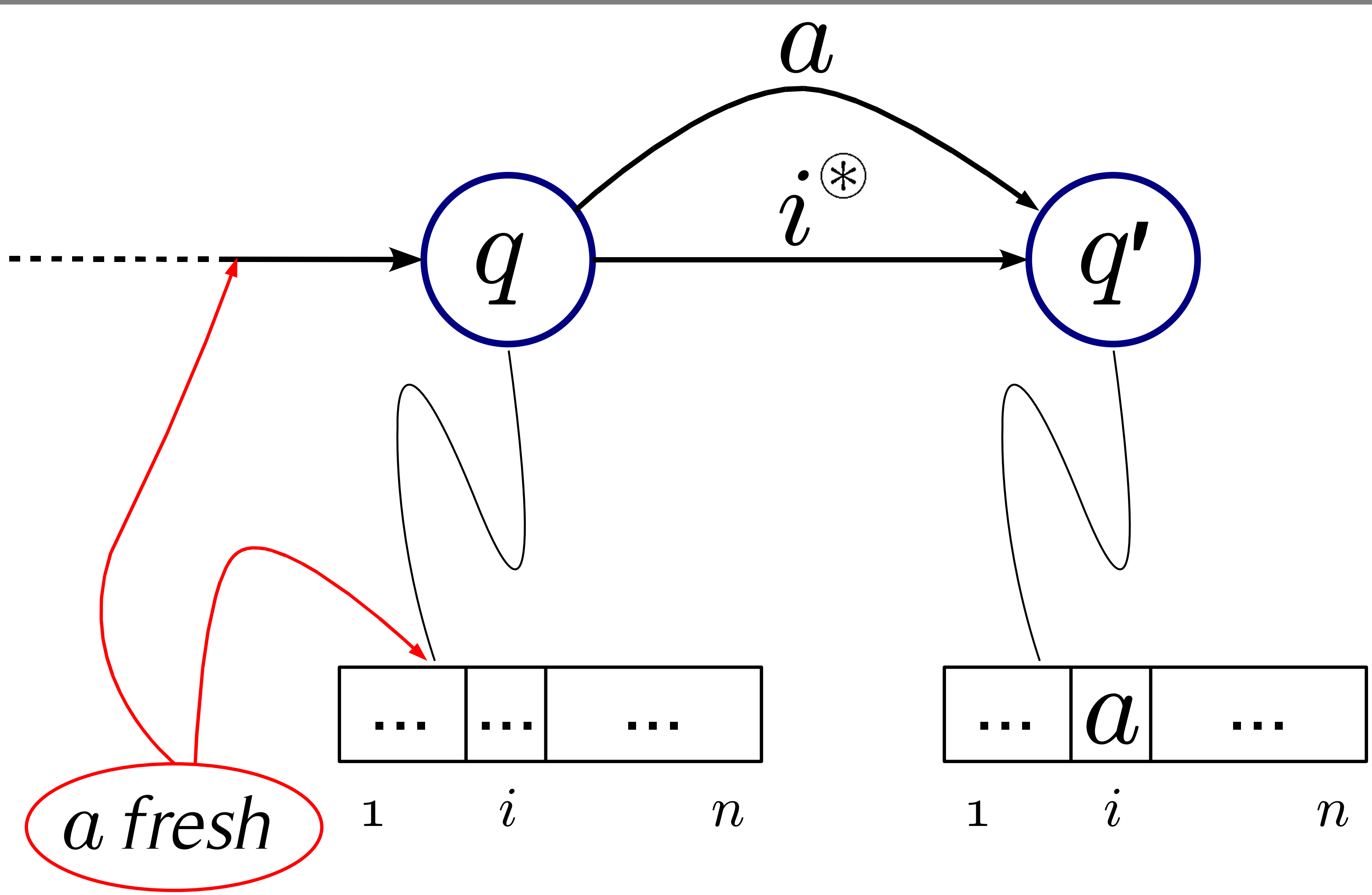


Demo: *globally fresh* transitions

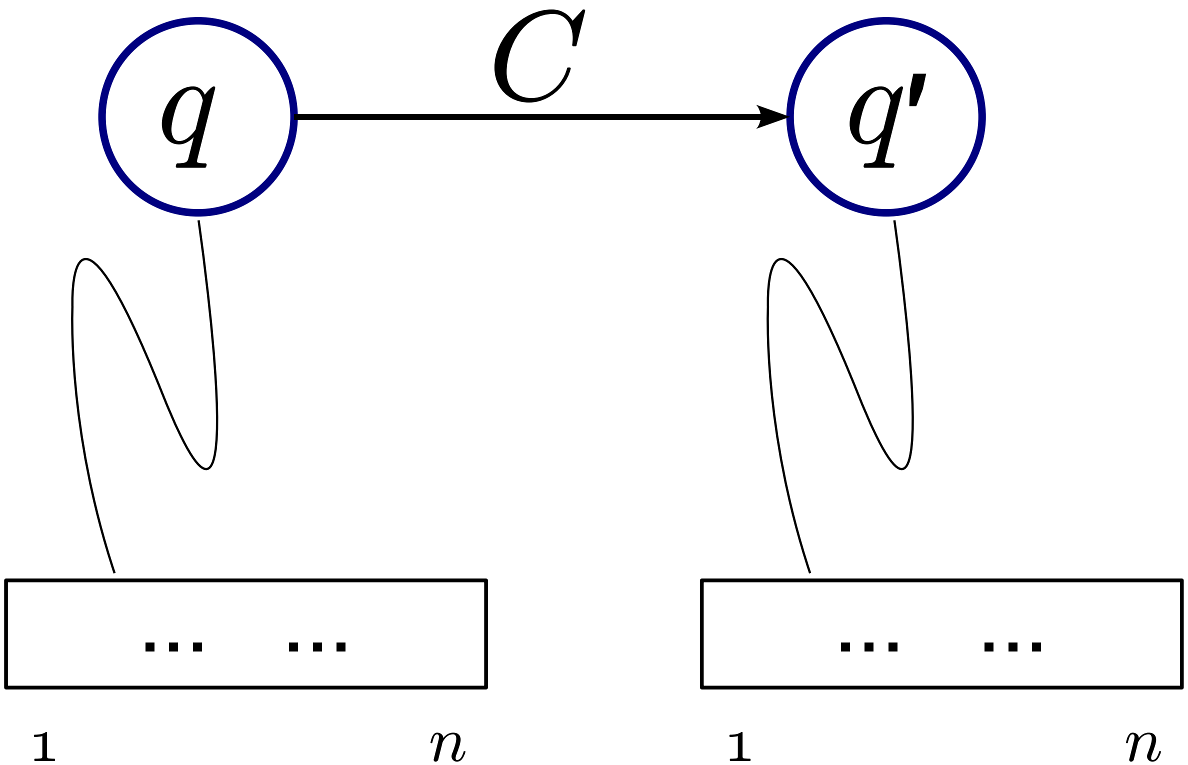


Demo: *globally fresh* transitions

An FMA is (equivalent to) an FRA without $(q, i^{\circledast}, q') \in \delta$



Demo: *constant* transitions



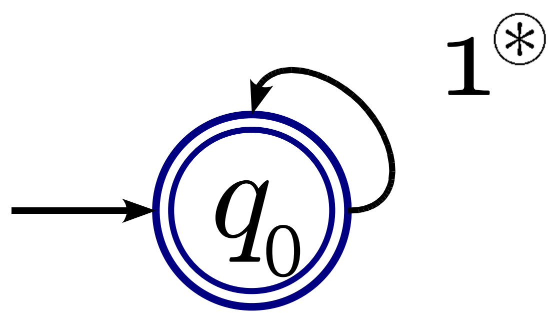
FRA's as language acceptors

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,
- $\sigma_0 \in \mathsf{Reg}_n$ 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.

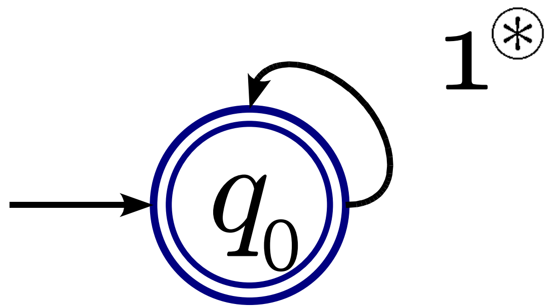
$$\mathcal{L}(\mathcal{A}) = \{\, \vec{\ell} \in (\mathbb{A} \cup \mathbb{C})^* \mid (q_0, \sigma_0, \emptyset) \xrightarrow{\vec{\ell}} (q, \sigma, H) \wedge q \in F \,\}$$

A name generator

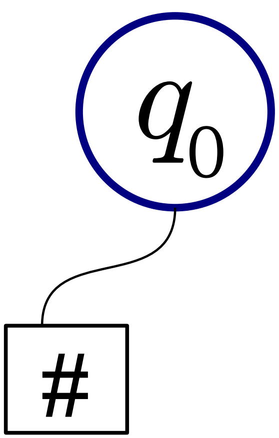


$$\sigma_0 = \{1 \mapsto \#\}$$

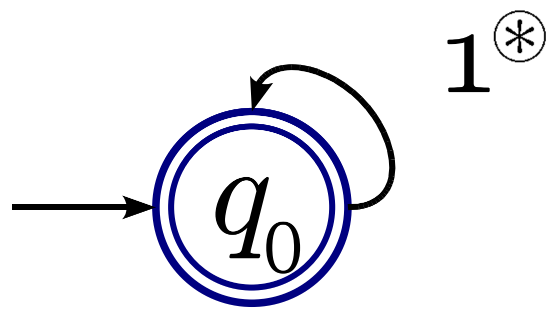
A name generator



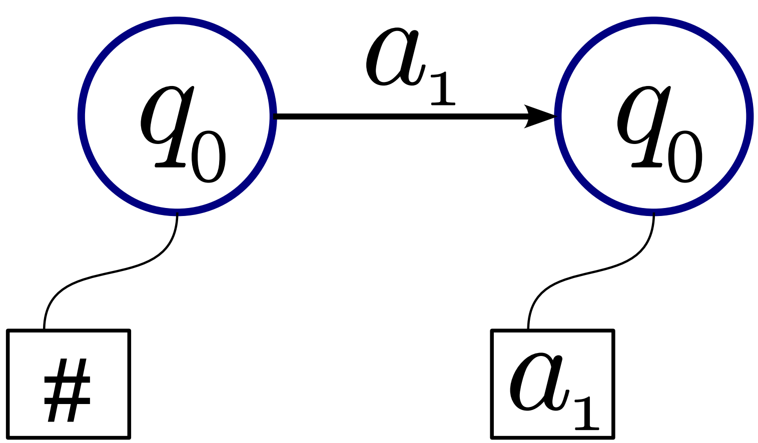
$$\sigma_0 = \{1 \mapsto \#\}$$



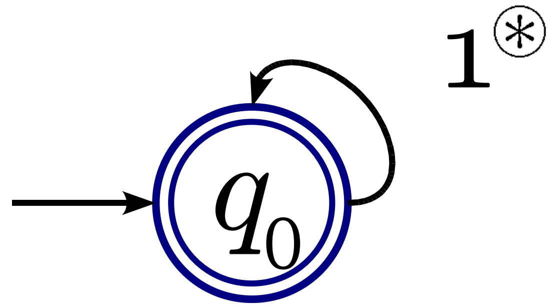
A name generator



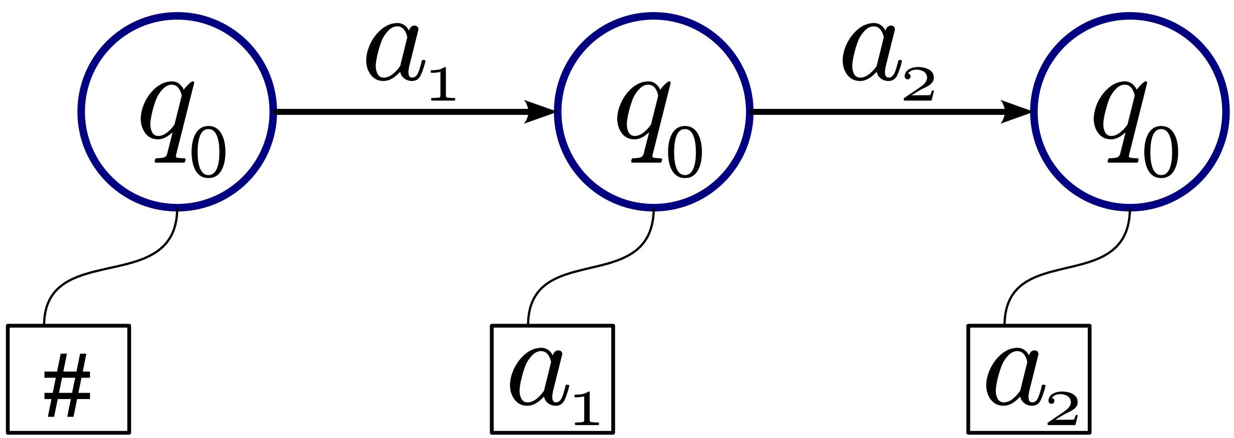
$$\sigma_0 = \{1 \mapsto \#\}$$



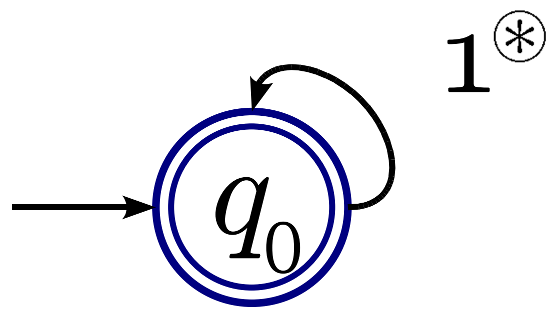
A name generator



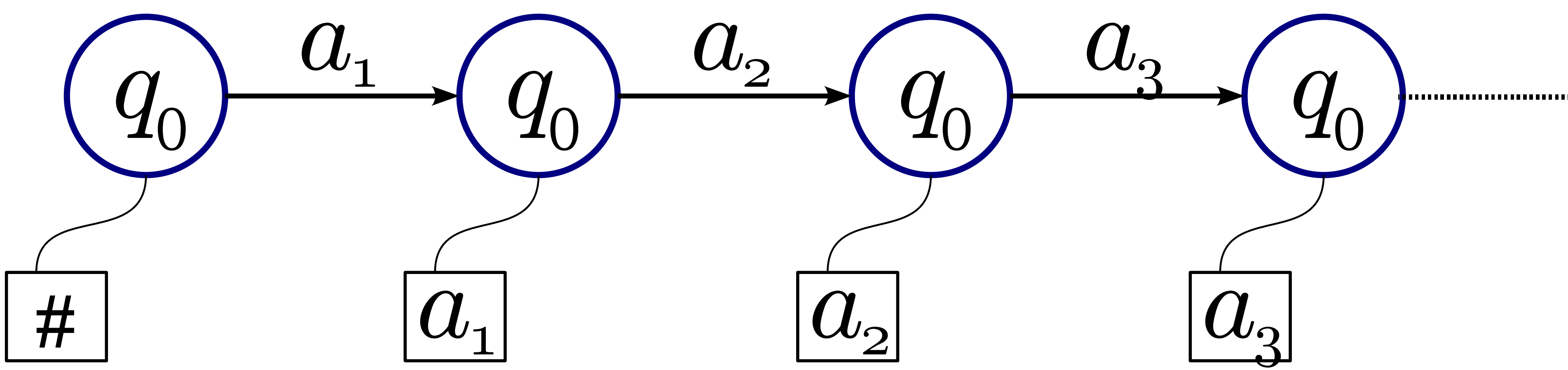
$$\sigma_0 = \{1 \mapsto \#\}$$



A name generator

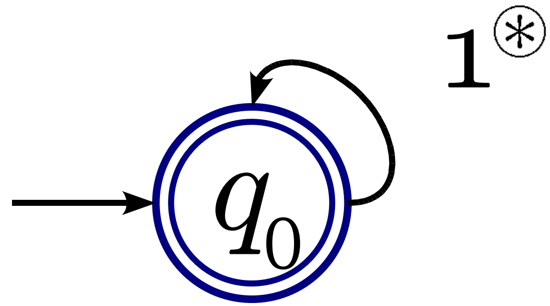


$$\sigma_0 = \{1 \mapsto \#\}$$



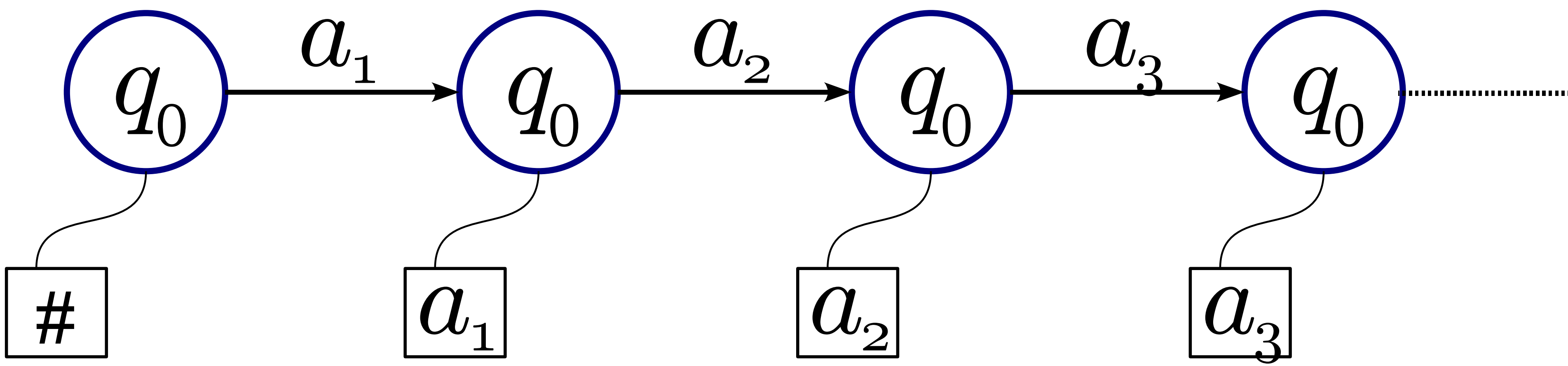
A name generator

L_1

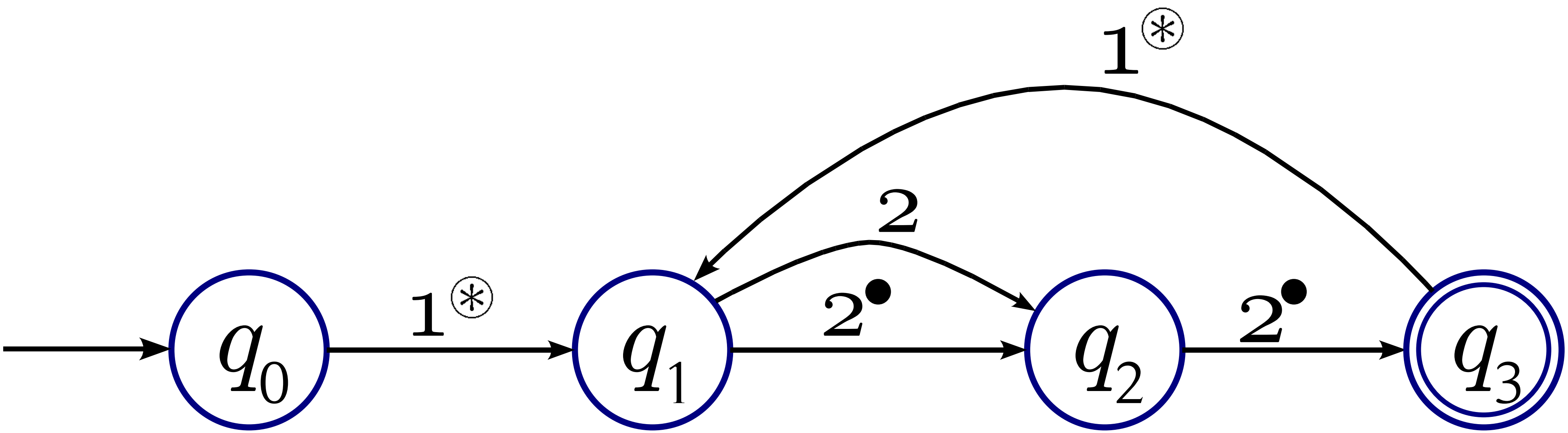


$$\sigma_0 = \{1 \mapsto \#\}$$

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



Another example



#	#

Properties

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

E.g. $L_1 * L_1$ is not FRA-recognisable.

Properties

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

E.g. $L_1 * L_1$ is not FRA-recognisable.

<u>Nominal</u> versions of concatenation and Kleene star?

Properties

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

  E.g. $L_1 * L_1$ is not FRA-recognisable.
- Non-closure under complementation.

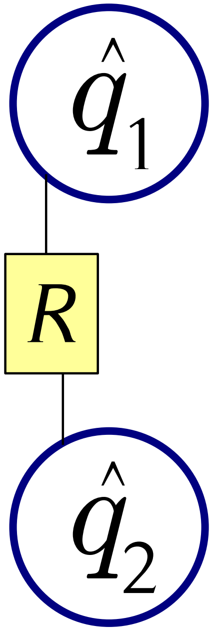
  E.g. $\overline{L_1 * L_1}$ is FRA-recognisable.

Bisimulations

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

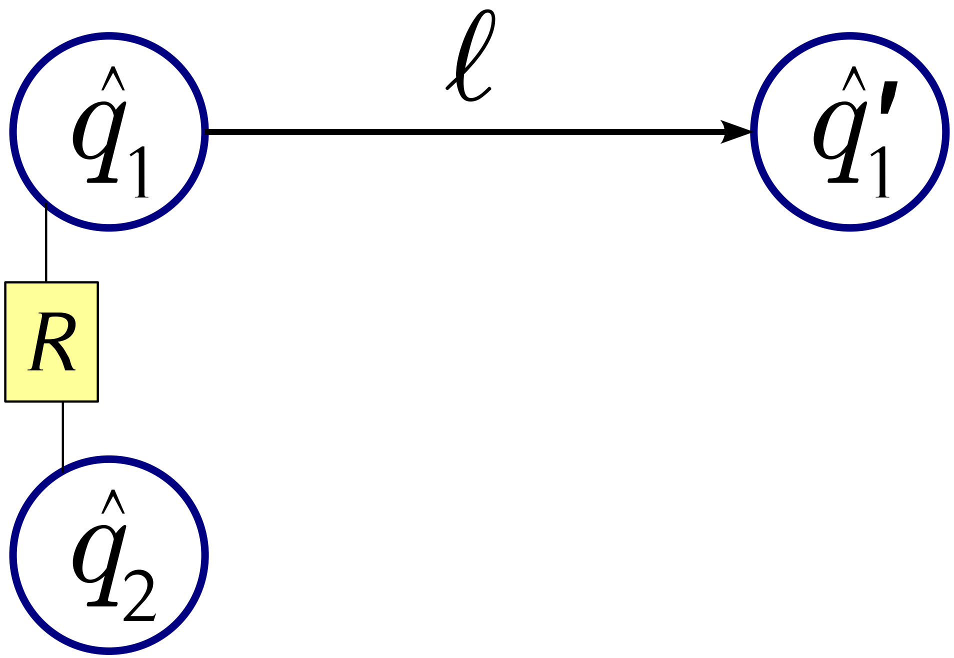
Bisimulations

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



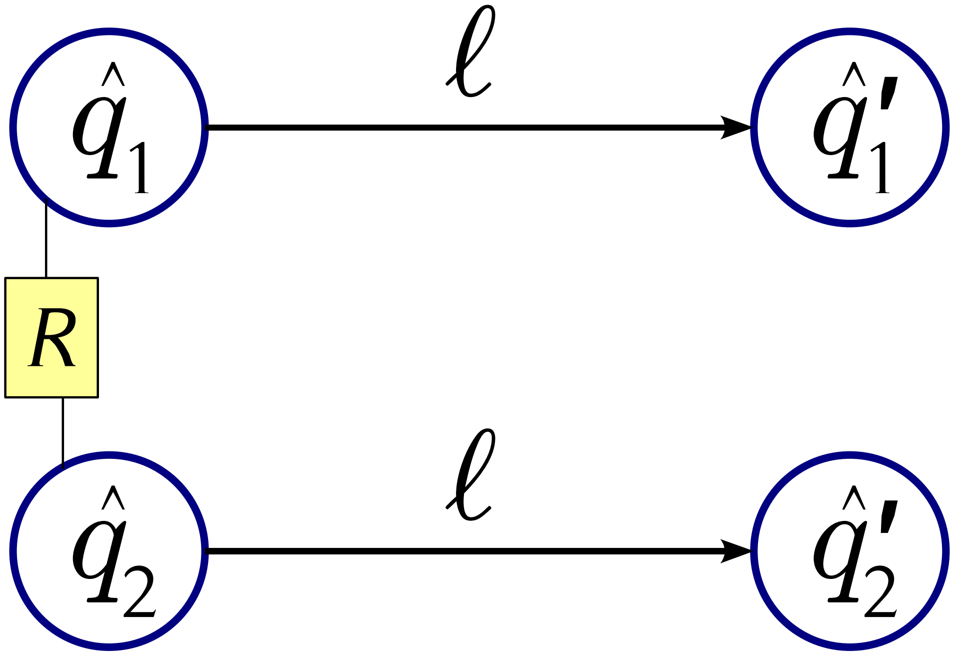
Bisimulations

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



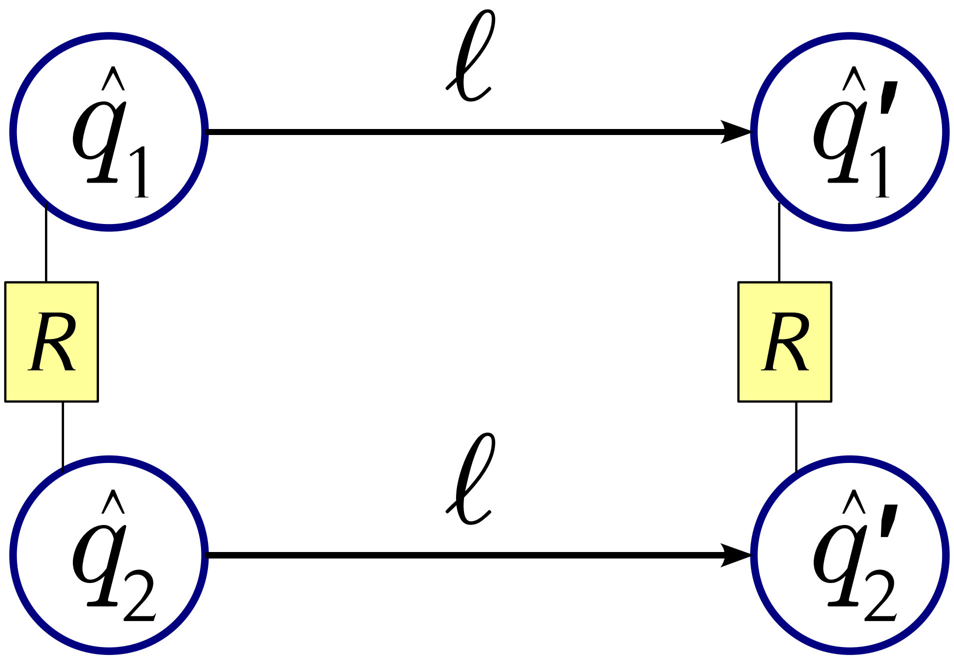
Bisimulations

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



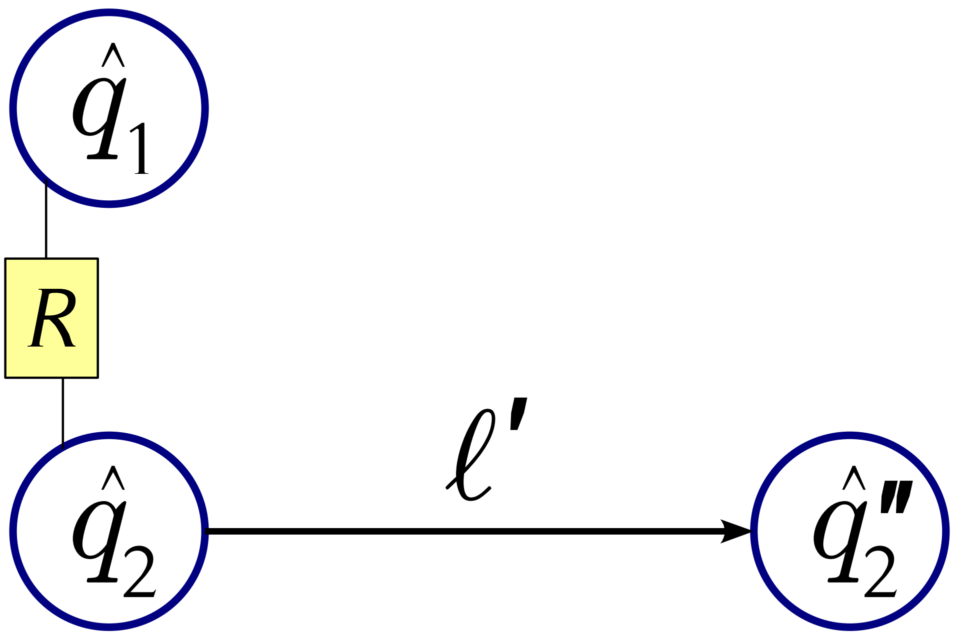
Bisimulations

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



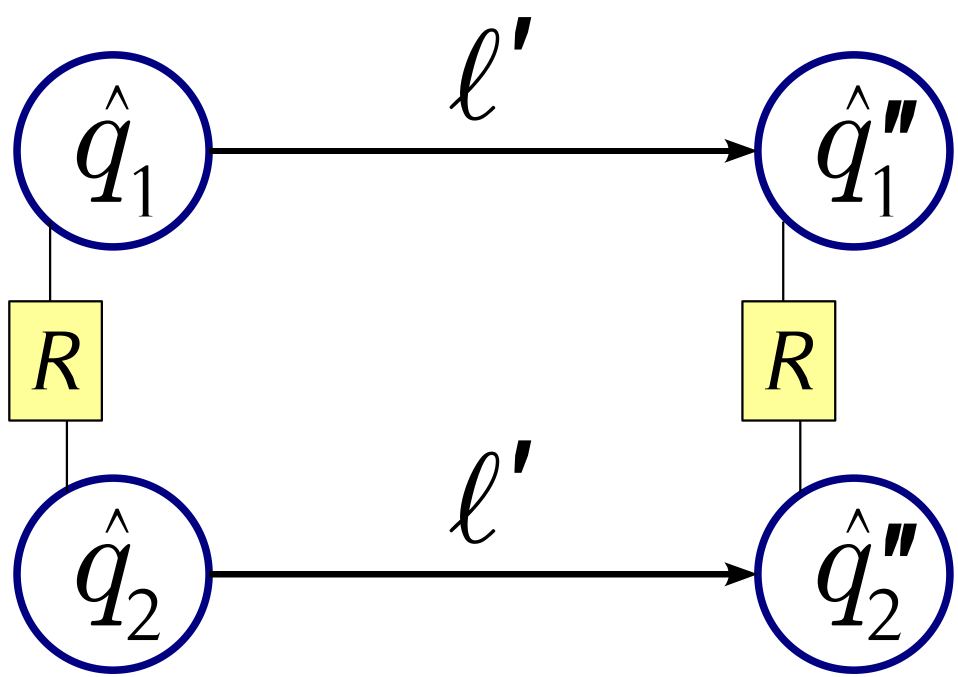
Bisimulations

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



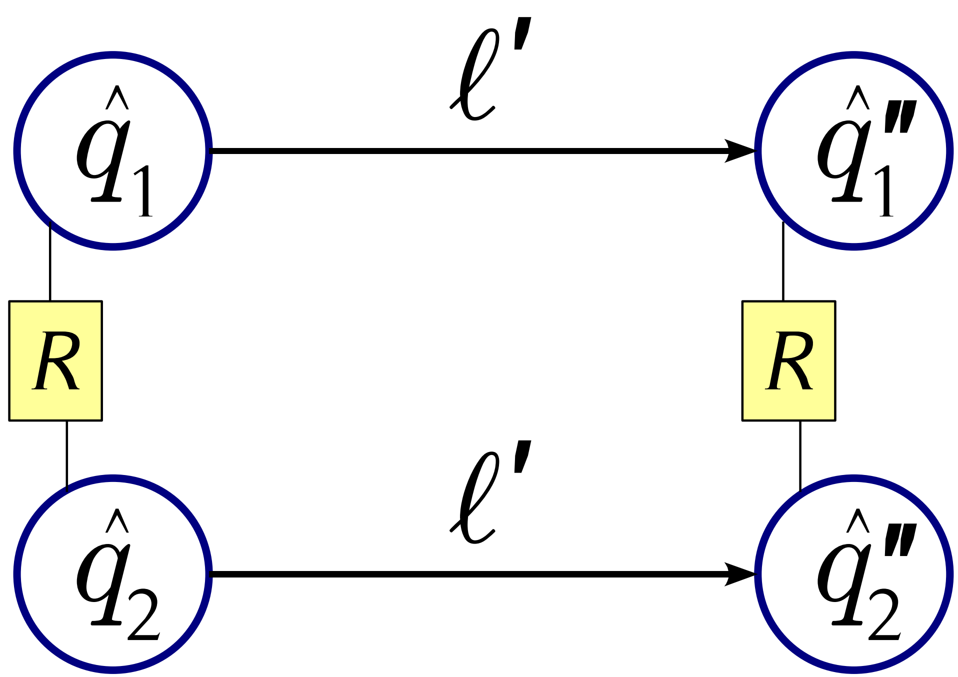
Bisimulations

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



Bisimulations

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



Lemma. Bisimilarity implies language equivalence.

Bisimulations formally

$R \subseteq \hat{Q}_1 \times \hat{Q}_2$ is called a ***simulation*** on $\mathcal{A}_1$ and $\mathcal{A}_2$ if, whenever $(q_1, \sigma_1, H_1)\, R\, (q_2, \sigma_2, H_2)$,

- if $q_1 \in F_1$ then $q_2 \in F_2$,
- if $(q_1, \sigma_1, H_1) \xrightarrow{\ell} (q_1', \sigma_1', H_1')$ then there is (q_2', σ_2', H_2') with:

$$(q_2, \sigma_2, H_2) \xrightarrow{\ell} (q_2', \sigma_2', H_2') \text{ and } (q_1', \sigma_1', H_1')\, R\, (q_2', \sigma_2', H_2').$$

R is called a ***bisimulation*** if both R and R^{-1} are simulations.

Bisimulations formally

$R \subseteq \hat{Q}_1 \times \hat{Q}_2$ is called a ***simulation*** on $\mathcal{A}_1$ and $\mathcal{A}_2$ if, whenever $(q_1, \sigma_1, H_1)\, R\, (q_2, \sigma_2, H_2)$,

- if $q_1 \in F_1$ then $q_2 \in F_2$,
- if $(q_1, \sigma_1, H_1) \xrightarrow{\ell} (q_1', \sigma_1', H_1')$ then there is (q_2', σ_2', H_2') with:

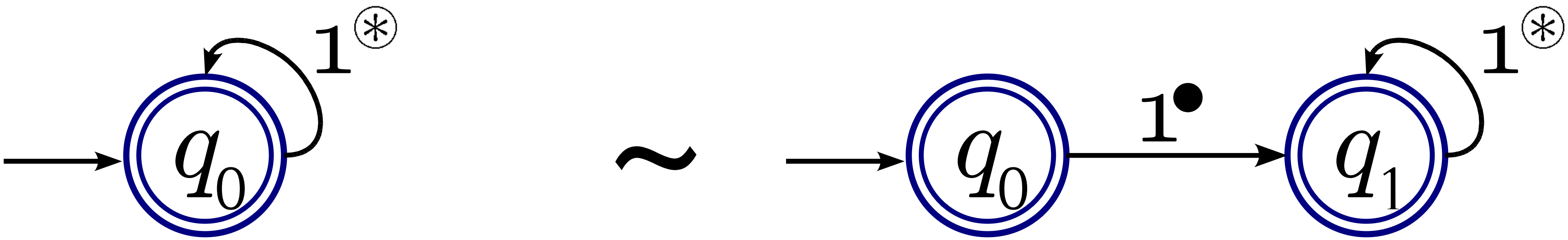
$$(q_2, \sigma_2, H_2) \xrightarrow{\ell} (q_2', \sigma_2', H_2') \text{ and } (q_1', \sigma_1', H_1')\, R\, (q_2', \sigma_2', H_2').$$

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:

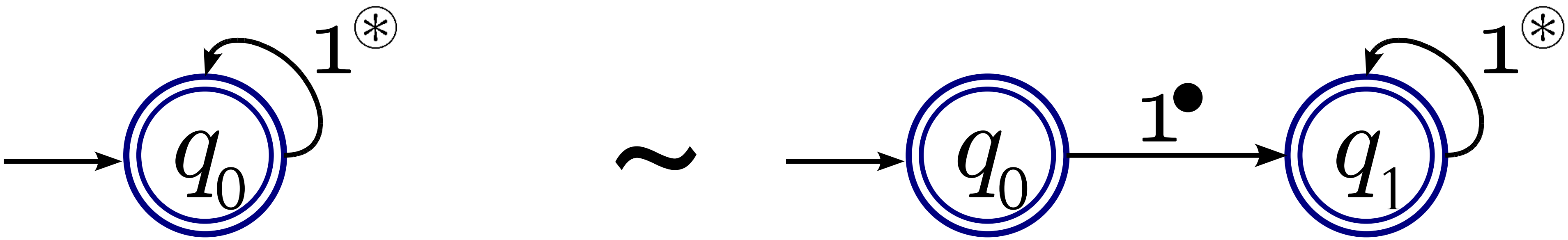
$$\boxed{(q_{01}, \sigma_{01}, \emptyset)\ R\ (q_{02}, \sigma_{02}, \emptyset)}$$

Example



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

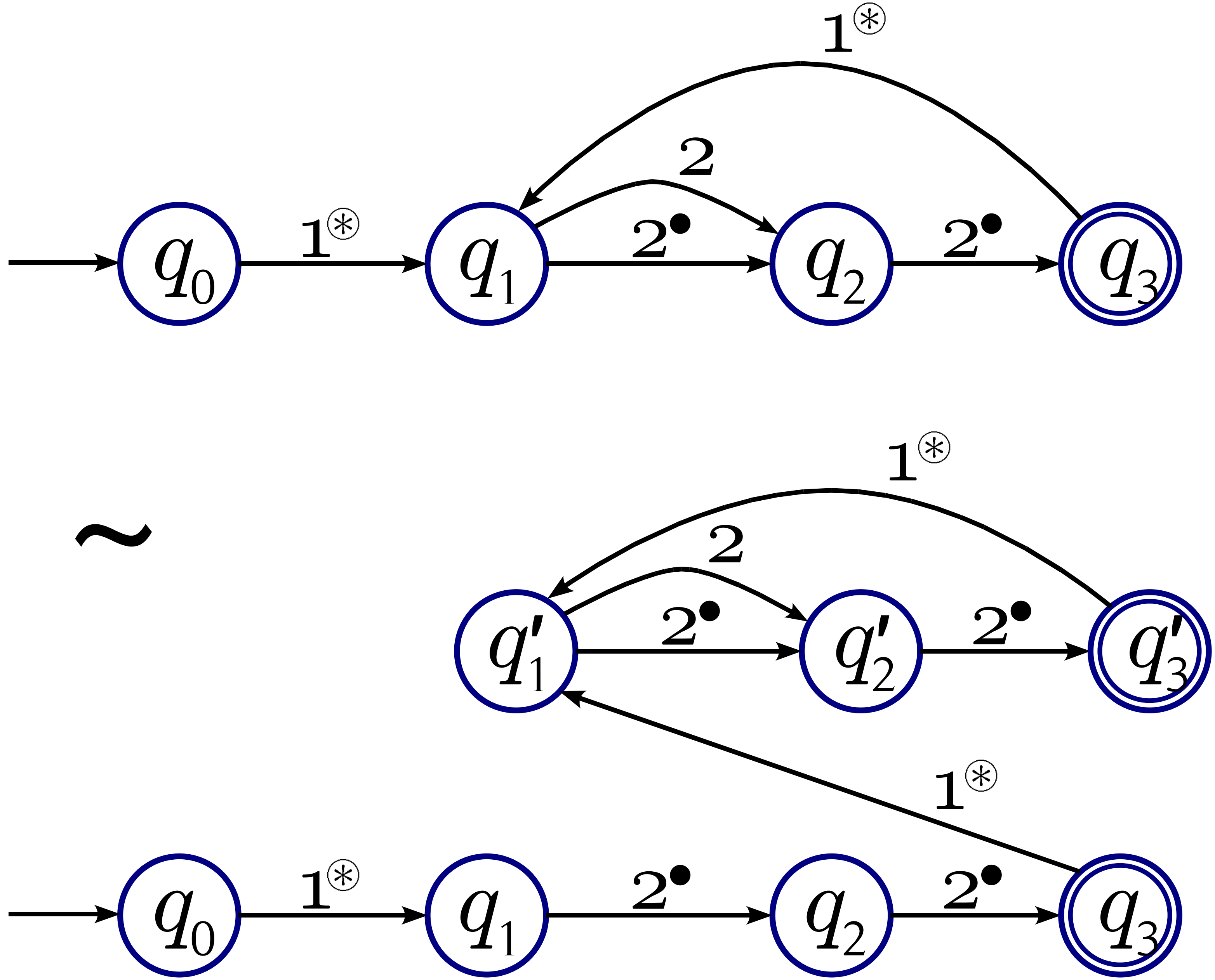
Example



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

$$R = \{\, ((q_0, \sigma_0, \emptyset), (q_0, \sigma_0, \emptyset)) \,\} \cup \{\, ((q_0, \sigma_1, H), (q_1, \sigma_2, H)) \,\}$$

Another example



Closed FRA's

- $\mathcal{A}$ is *closed* if it has no blocking transitions:

$$(q_0, \sigma_0, \emptyset) \xrightarrow{\vec{\ell}} (q, \sigma, H) \wedge (q, i, q') \in \delta \implies \sigma(i) \neq \#$$

Closed FRA's

- $\mathcal{A}$ is *closed* if it has no blocking transitions:

$$(q_0, \sigma_0, \emptyset) \xrightarrow{\vec{\ell}} (q, \sigma, H) \wedge (q, i, q') \in \delta \implies \sigma(i) \neq \#$$

Lemma. For any FRA $\mathcal{A}$ we can effectively construct a closed FRA $\overline{\mathcal{A}} \sim \mathcal{A}$.

Corollary. FRA-emptiness is decidable.

Symbolic bisimulations

- *Symbolic* reasoning can be used for bisimulations too:
  - We can define a notion of symbolic bisimulation:

$$R \subseteq Q_1 \times \{0, \ldots, n_1 + n_2\} \times (n_1 \rightleftharpoons n_2) \times Q_2$$

  - capturing (actual) bisimilarity.

Symbolic bisimulations

- *Symbolic* reasoning can be used for bisimulations too:
  - We can define a notion of symbolic bisimulation:

$$R \subseteq Q_1 \times \{0, \ldots, n_1 + n_2\} \times (n_1 \rightleftharpoons n_2) \times Q_2$$

  - capturing (actual) bisimilarity.

Corollary. 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 symbolic means.

Application: the pi-calculus

Application: the pi-calculus

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

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

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

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

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

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

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

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

Application: the pi-calculus

$$\textsc{Inp1}\ \frac{}{\sigma \vdash a(b).P \xrightarrow{i} \sigma \vdash (b).P}\ \sigma(i)=a$$

$$\textsc{Inp2a}\ \frac{}{\sigma \vdash (b).P \xrightarrow{i} \sigma \vdash P\{a/b\}}\ \sigma(i)=a$$

$$\textsc{Inp2b}\ \frac{}{\sigma \vdash (b).P \xrightarrow{i^{\bullet}} \sigma[i \mapsto b] \vdash P}\ \sigma(i)\notin \mathsf{fn}(P)$$

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

Application: the pi-calculus

$$\textsc{Out1}\ \frac{}{\sigma \vdash \bar{a}b.P \xrightarrow{i} \sigma \vdash b.P}\ \sigma(i){=}a$$

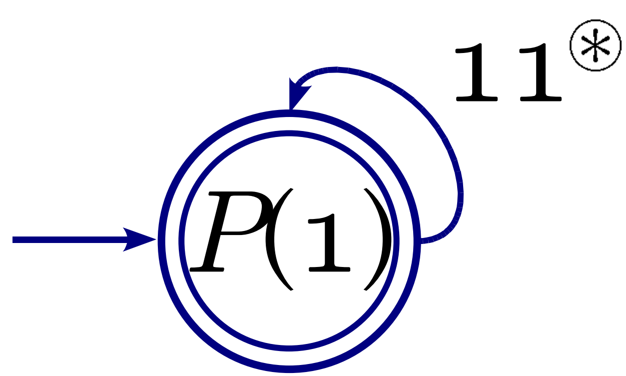
$$\textsc{Out2}\ \frac{}{\sigma \vdash b.P \xrightarrow{i} \sigma \vdash P}\ \sigma(i){=}b$$

$$\textsc{Open}\ \frac{\sigma[i \mapsto a] \vdash P_{\mathsf{out}} \xrightarrow{i} \sigma[i \mapsto a] \vdash P}{\sigma \vdash \nu a.P_{\mathsf{out}} \xrightarrow{i^{\circledast}} \sigma[i \mapsto a] \vdash P}\ \sigma(i){\notin}\mathsf{fn}(P)$$

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

Application: the pi-calculus

$$P(a) = \nu b.\ \overline{a}b.\ P(b)$$



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
- Connections to HD-automata
- Nominal notions of concatenation, Kleene closure
- Variations:
  - Labels
  - Stores
  - Stack