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)

Names in computation

```
let x₁,x₂,x₃,x₄ = ref() in
    λx. case x of  x₁ => x₂
                 | x₂ => x₃
                 | x₃ => x₄
                 | _  => 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
- 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
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:

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

Definition

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

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

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

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

Definition

$\mathcal{A} = \langle Q, q_0, \sigma_0, \delta, F \rangle$ 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.

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

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

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.

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

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

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:

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

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:

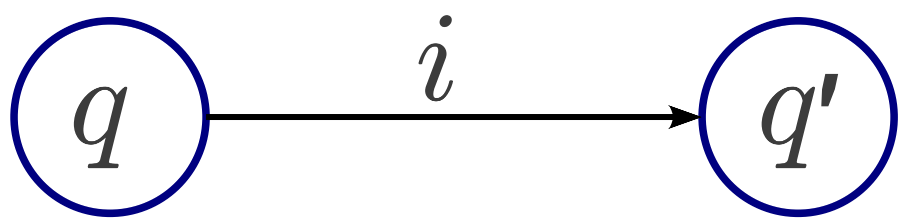
- Q is a finite set of states,
- $q_0 \in Q$ is the initial state,
- $\sigma_0 \in \mathsf{Reg}_n$ is the initial register assignment,
- $\delta \subseteq Q \times \mathbb{L}_n \times Q$ is the transition relation,
- $F \subseteq Q$ is the set of final states.

- A configuration is a triple:

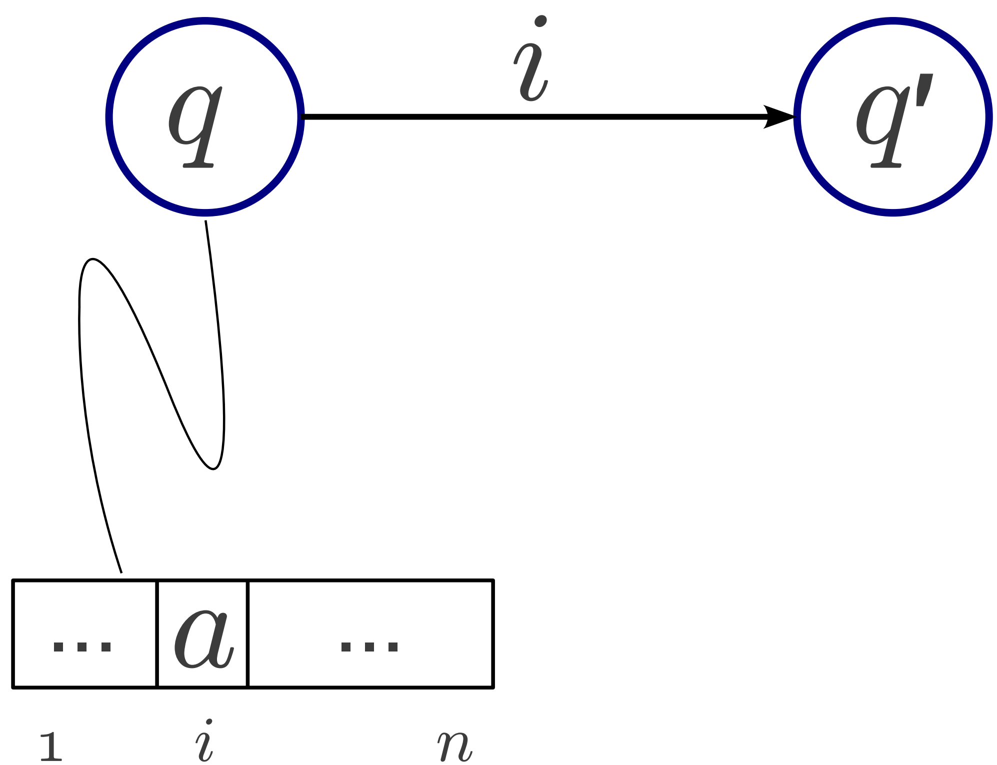
$$(q, \sigma, H) \in Q \times \mathsf{Reg}_n \times \mathcal{P}_{\mathsf{fn}}(\mathbb{A})$$

- Semantics: Transition relation on configurations.

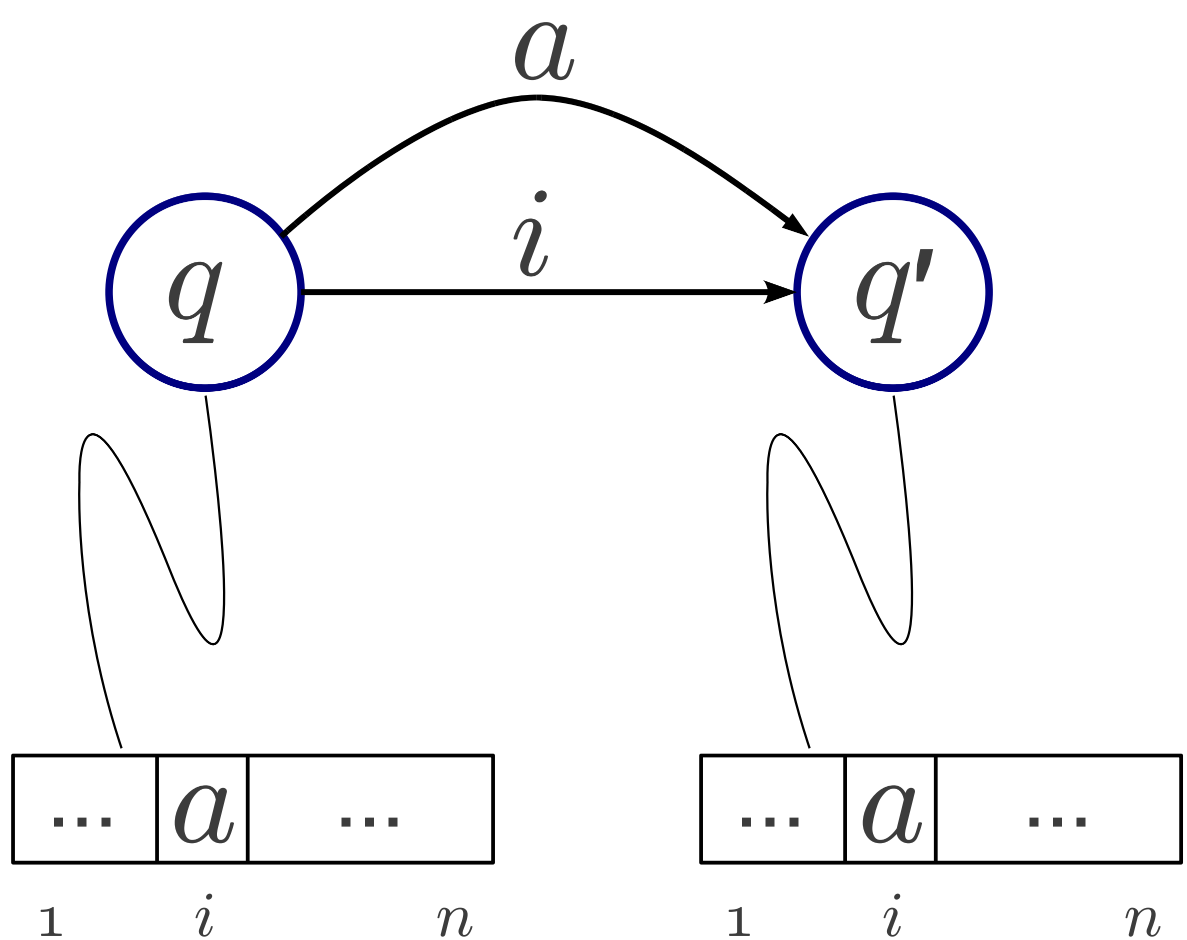
Demo: *known* transitions



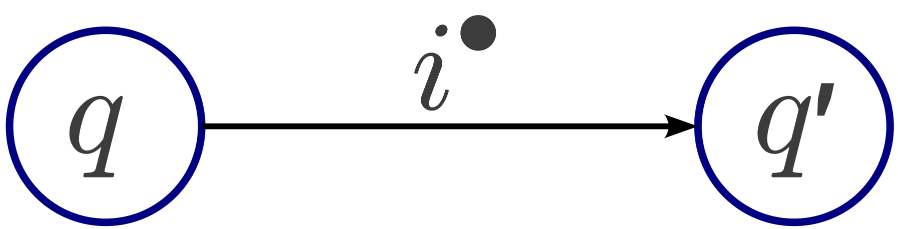
Demo: *known* transitions



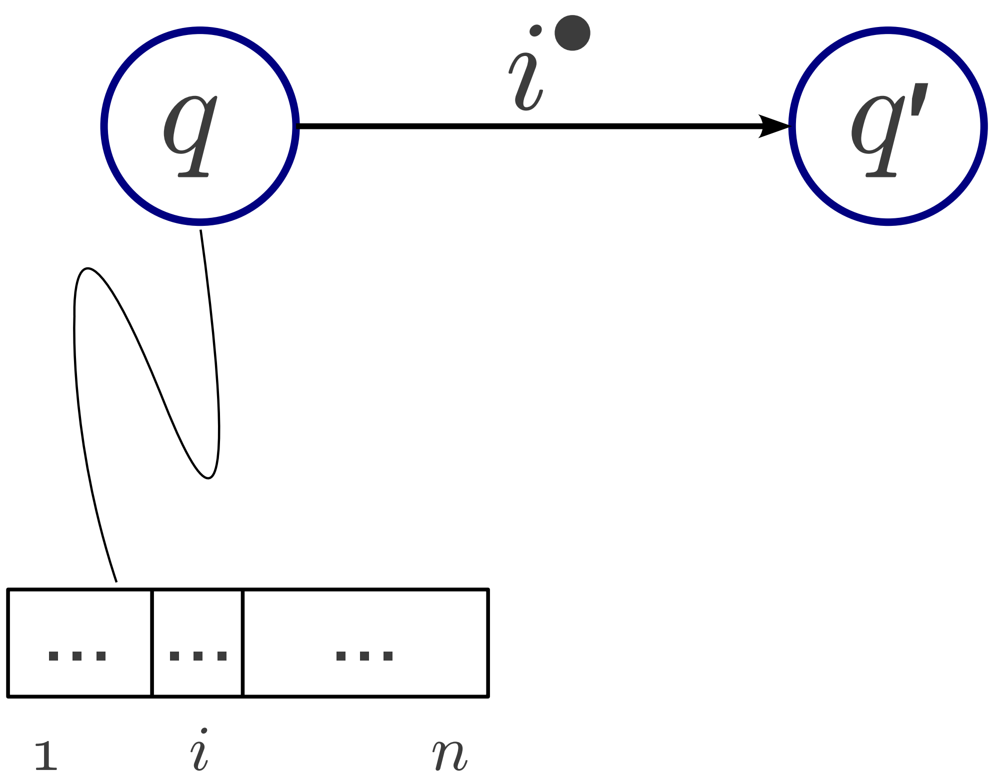
Demo: *known* transitions



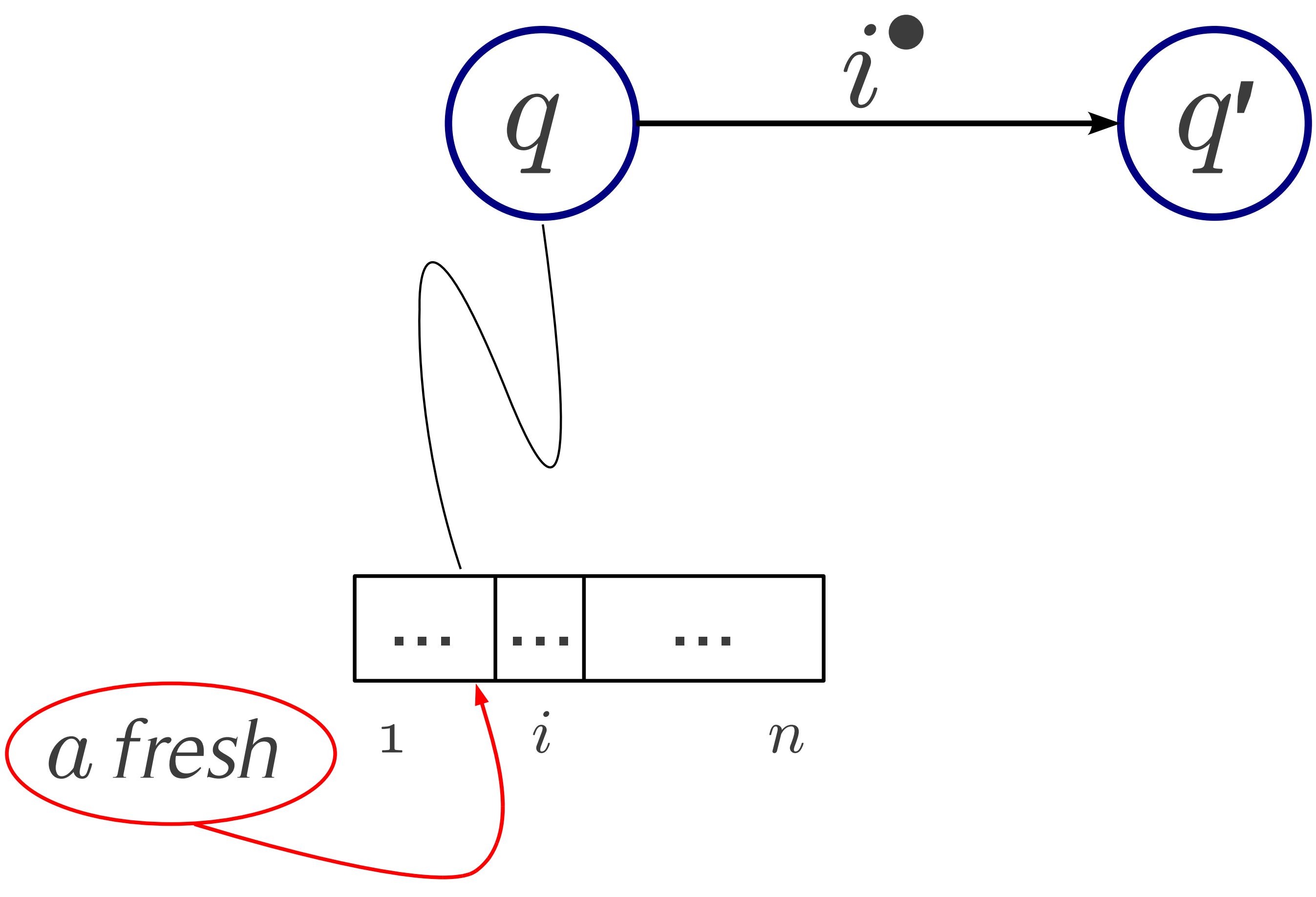
Demo: *locally fresh* transitions



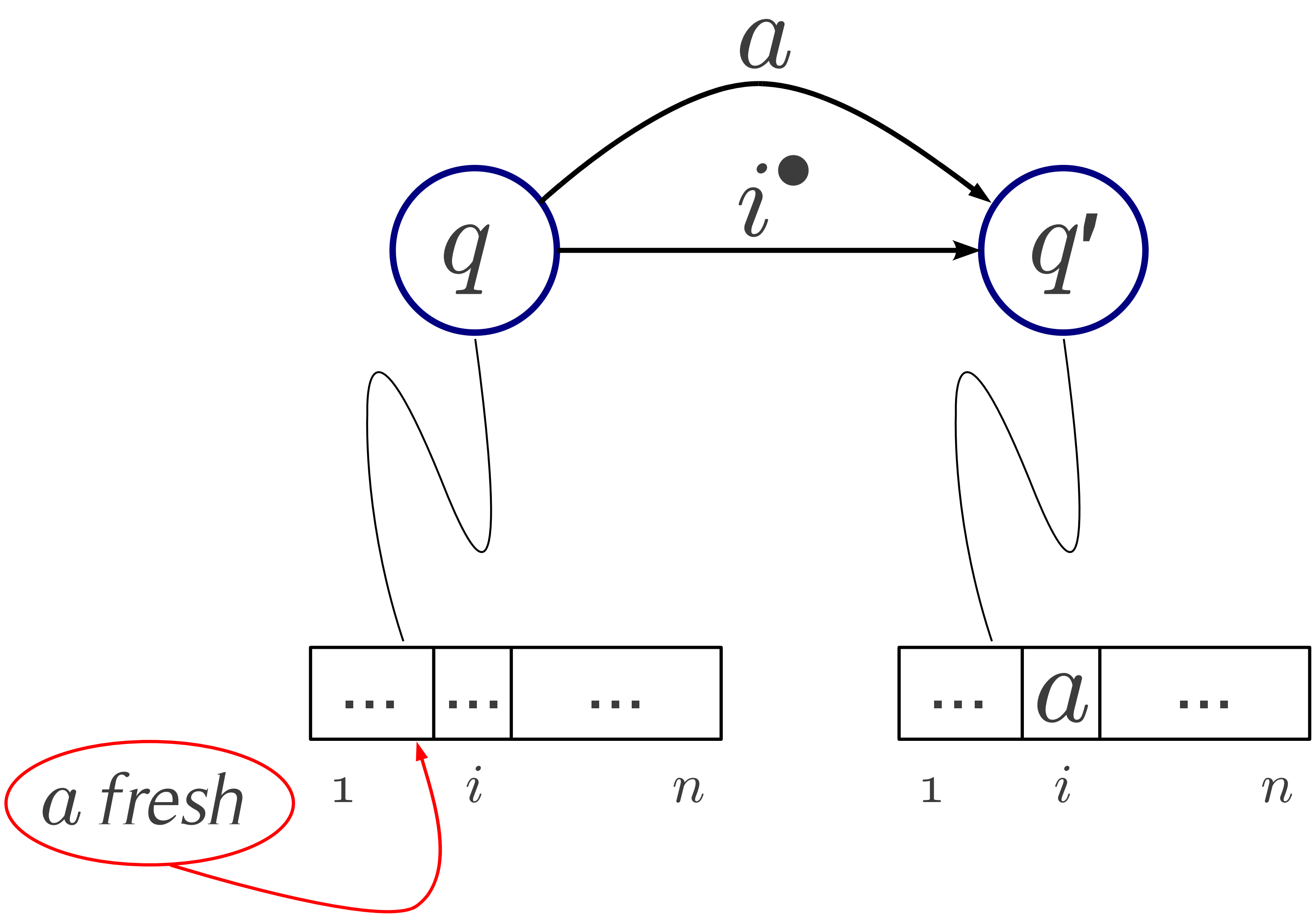
Demo: *locally fresh* transitions



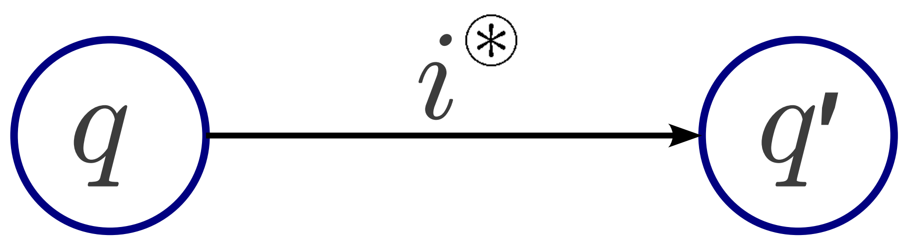
Demo: *locally fresh* transitions



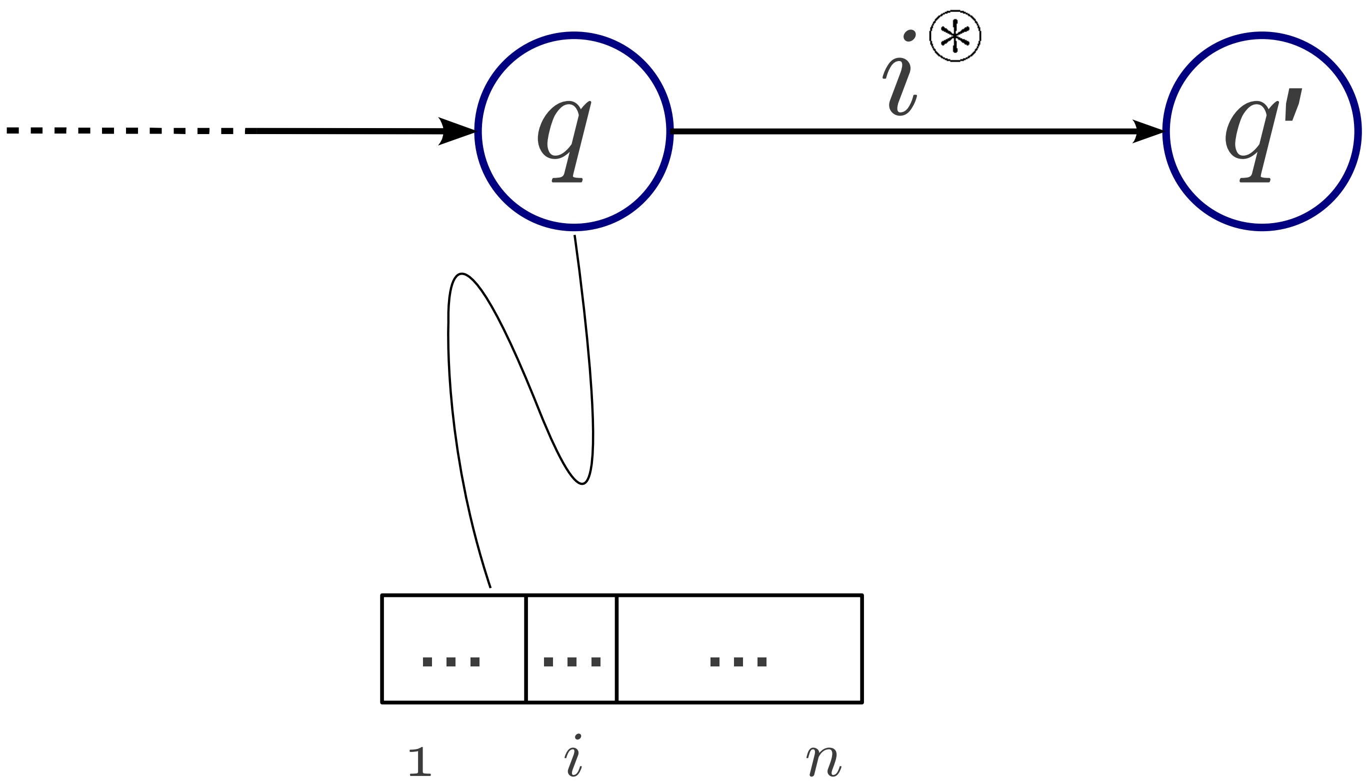
Demo: *locally fresh* transitions



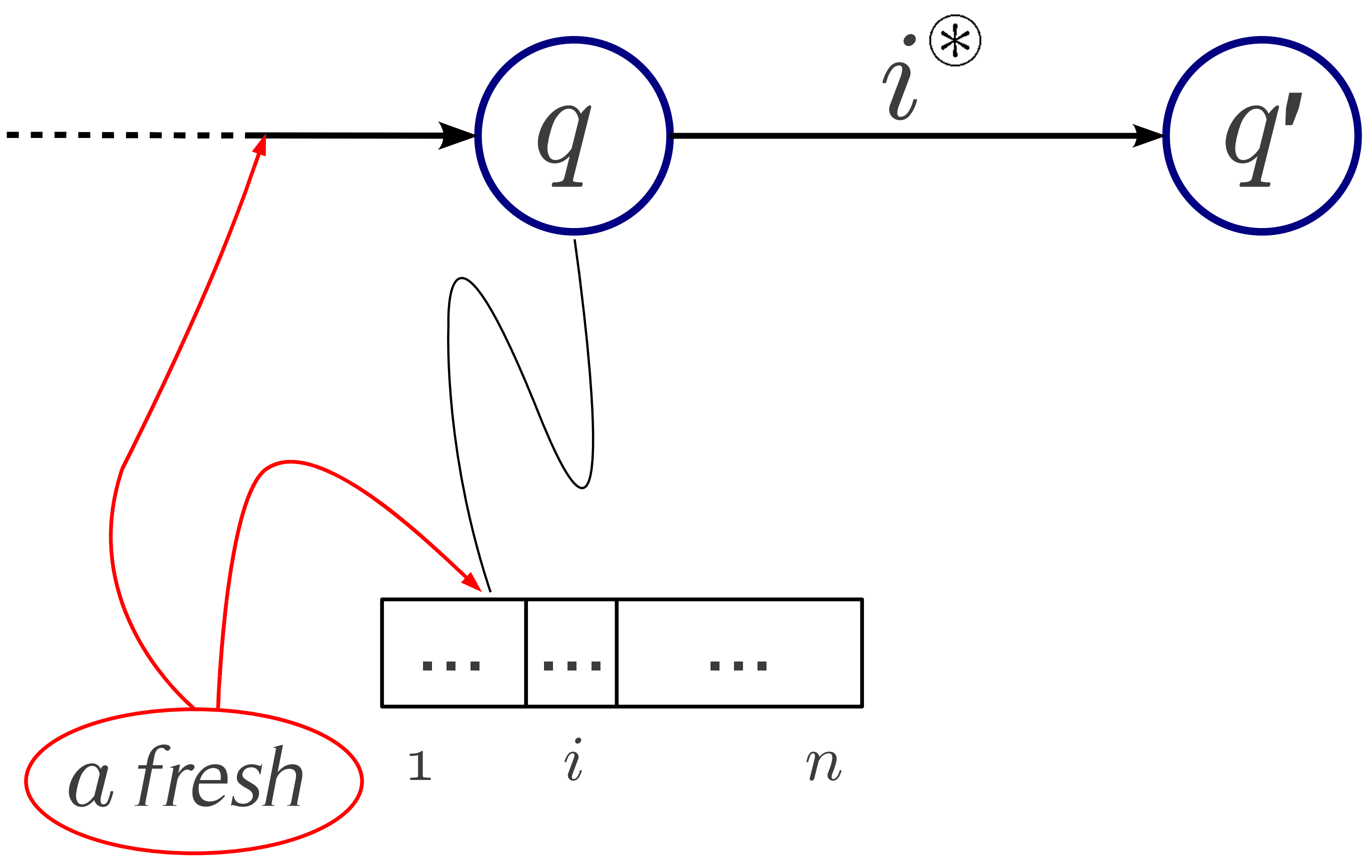
Demo: *globally fresh* transitions



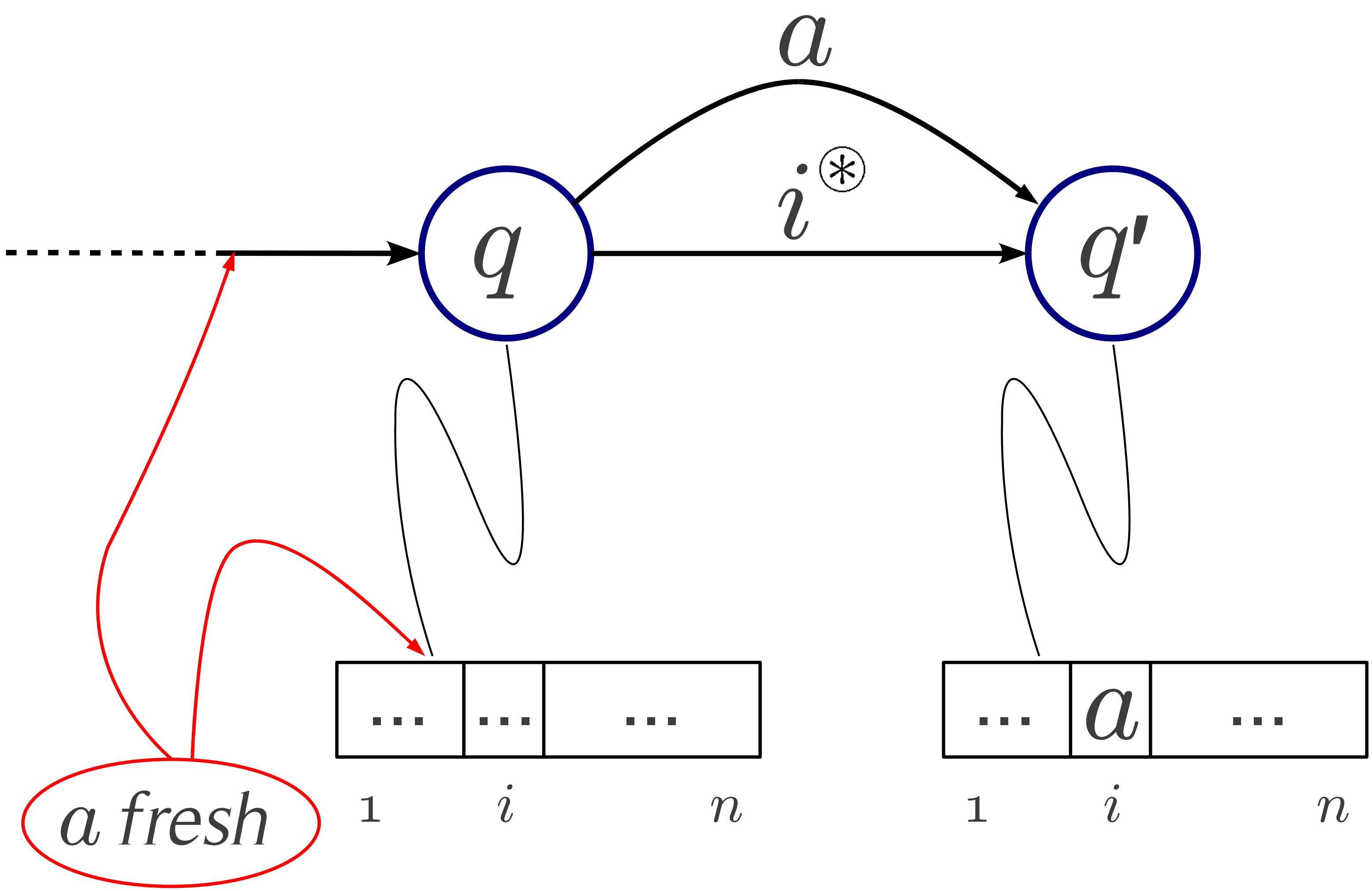
Demo: *globally fresh* transitions



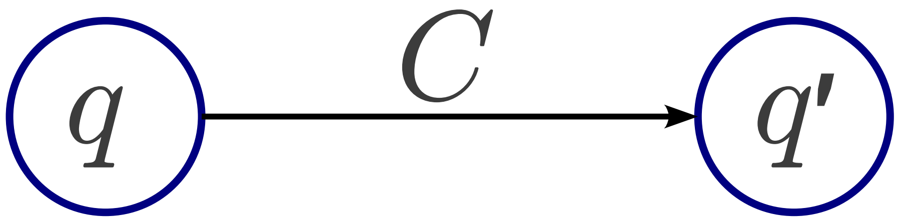
Demo: *globally fresh* transitions



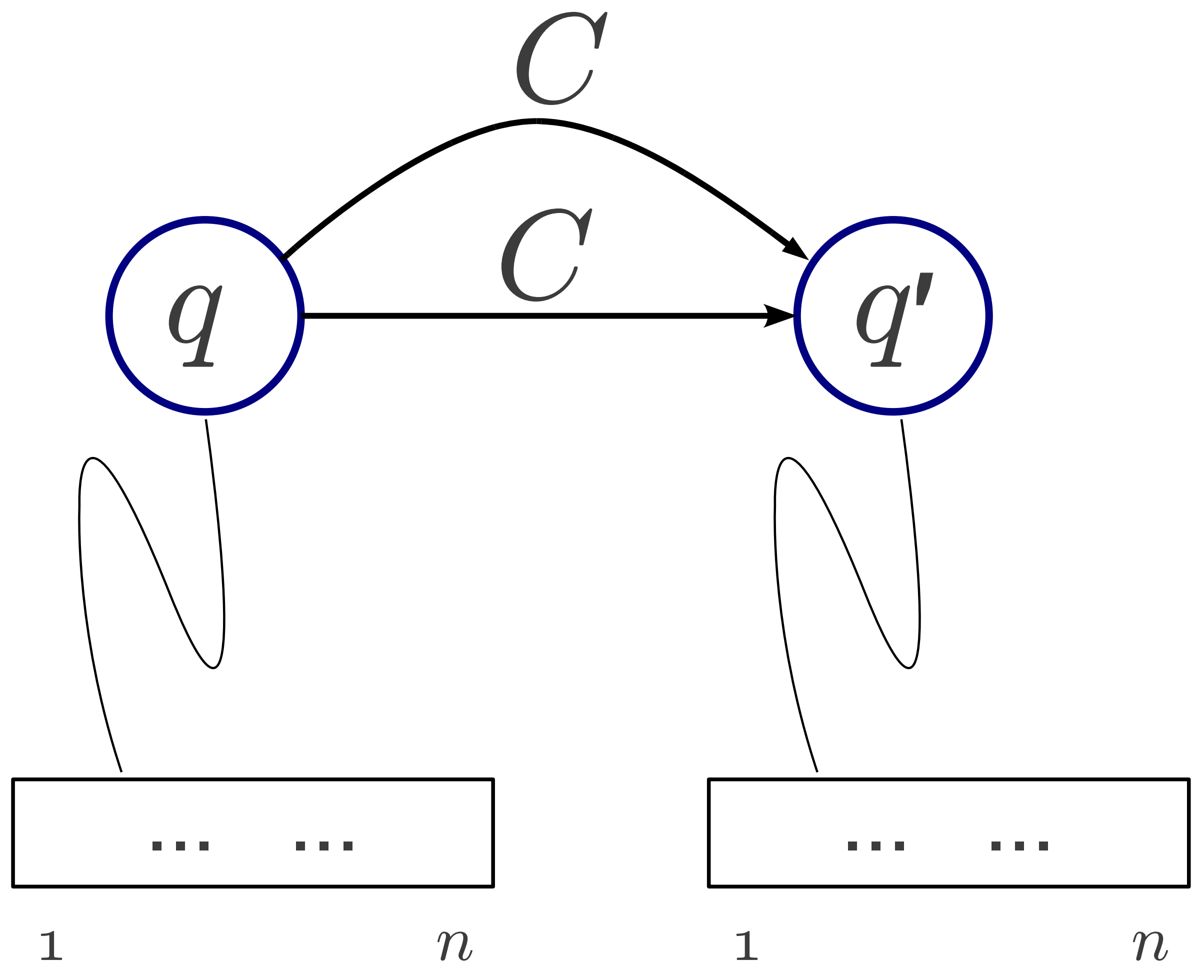
Demo: *globally fresh* transitions



Demo: *constant* transitions



Demo: *constant* transitions



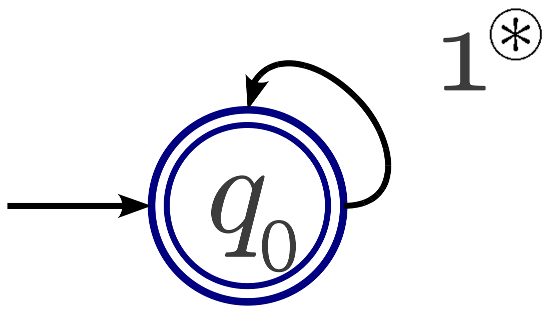
FRA's as language acceptors

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.

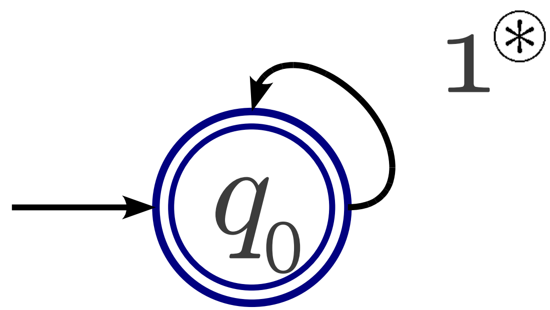
$$\mathcal{L}(\mathcal{A}) = \{\, \vec{\ell} \in (\mathbb{A} \cup \mathbb{C})^* \mid (q_0, \sigma_0, \emptyset) \xrightarrow{\vec{\ell}} (q, \sigma, H) \wedge q \in F \,\}$$

A name generator

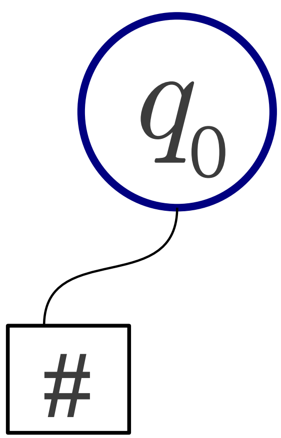


$$\sigma_0 = \{1 \mapsto \#\}$$

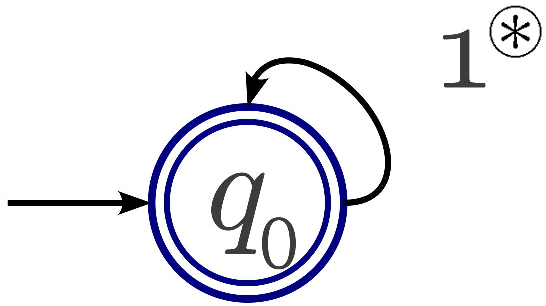
A name generator



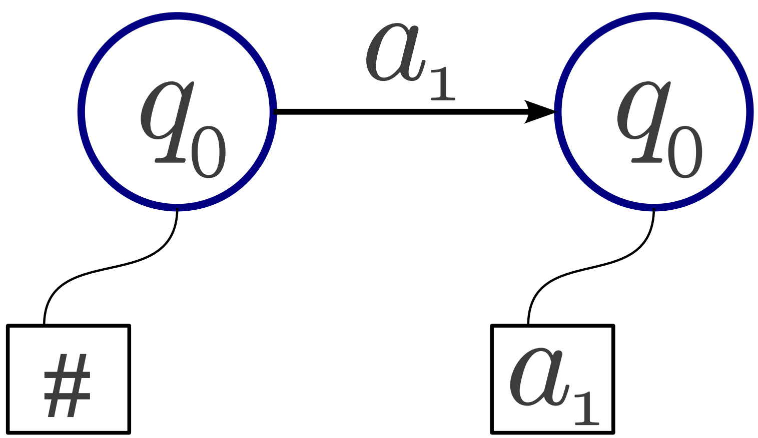
$$\sigma_0 = \{1 \mapsto \#\}$$



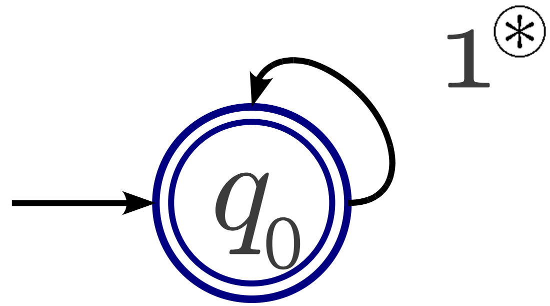
A name generator



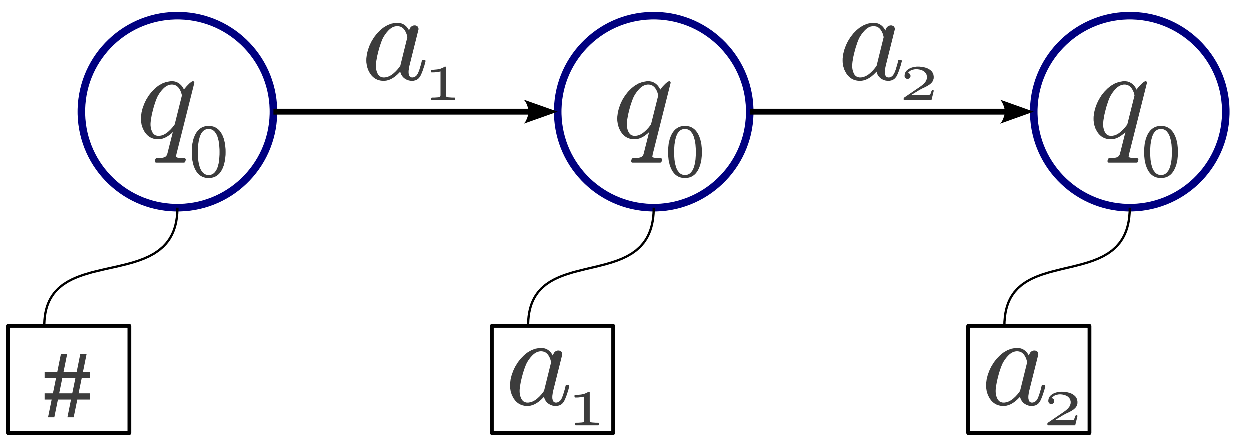
$$\sigma_0 = \{1 \mapsto \#\}$$



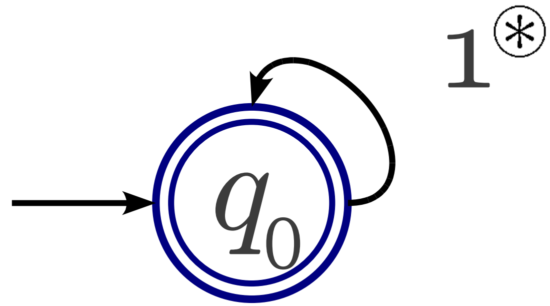
A name generator



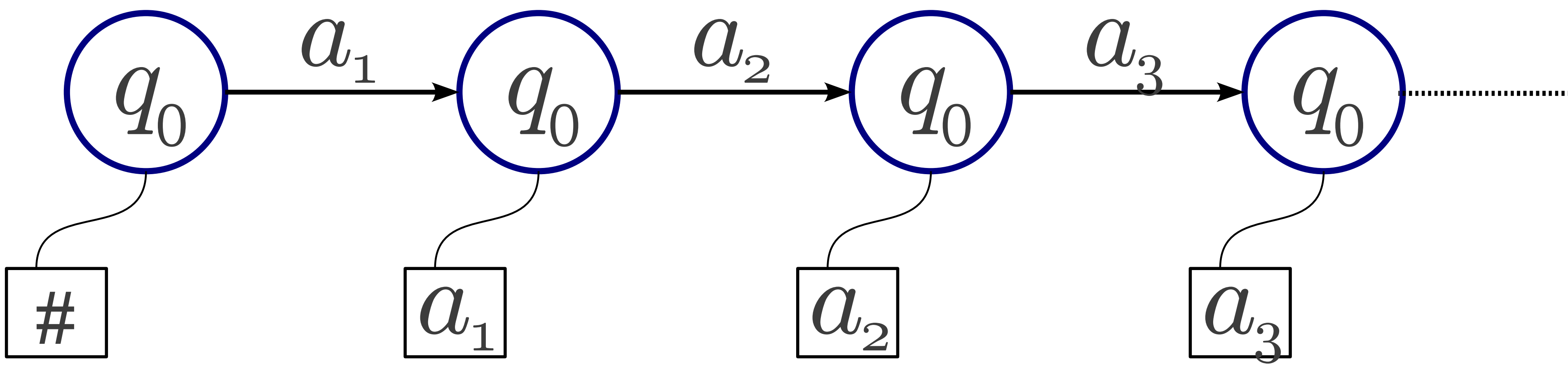
$$\sigma_0 = \{1 \mapsto \#\}$$



A name generator

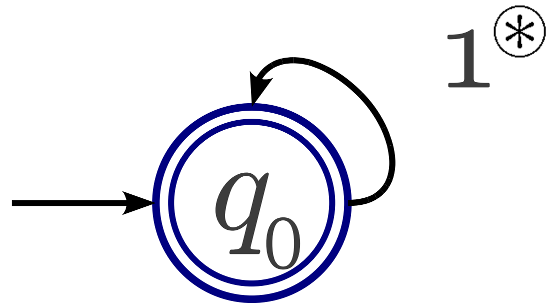


$$\sigma_0 = \{1 \mapsto \#\}$$



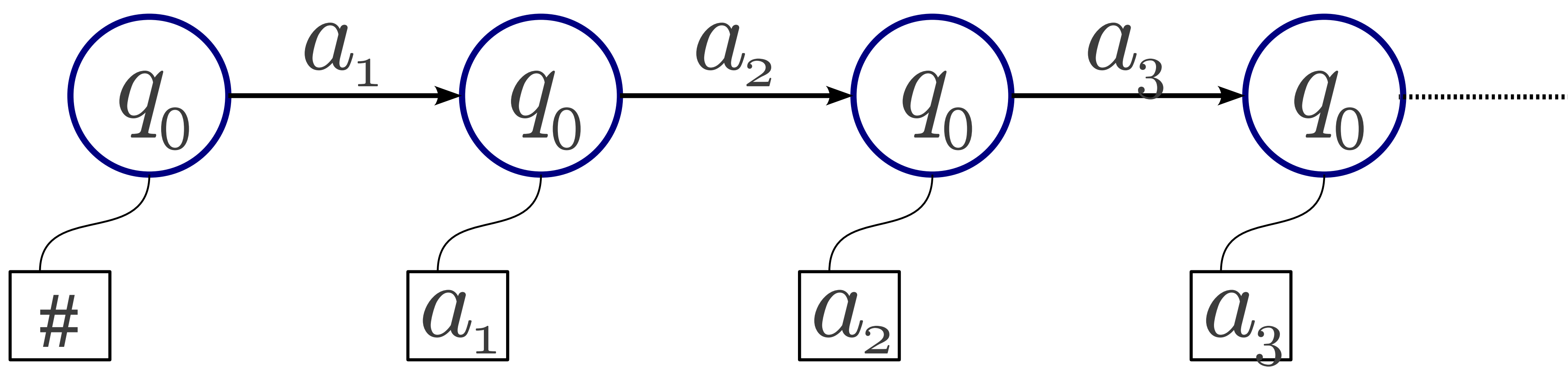
A name generator

L_1

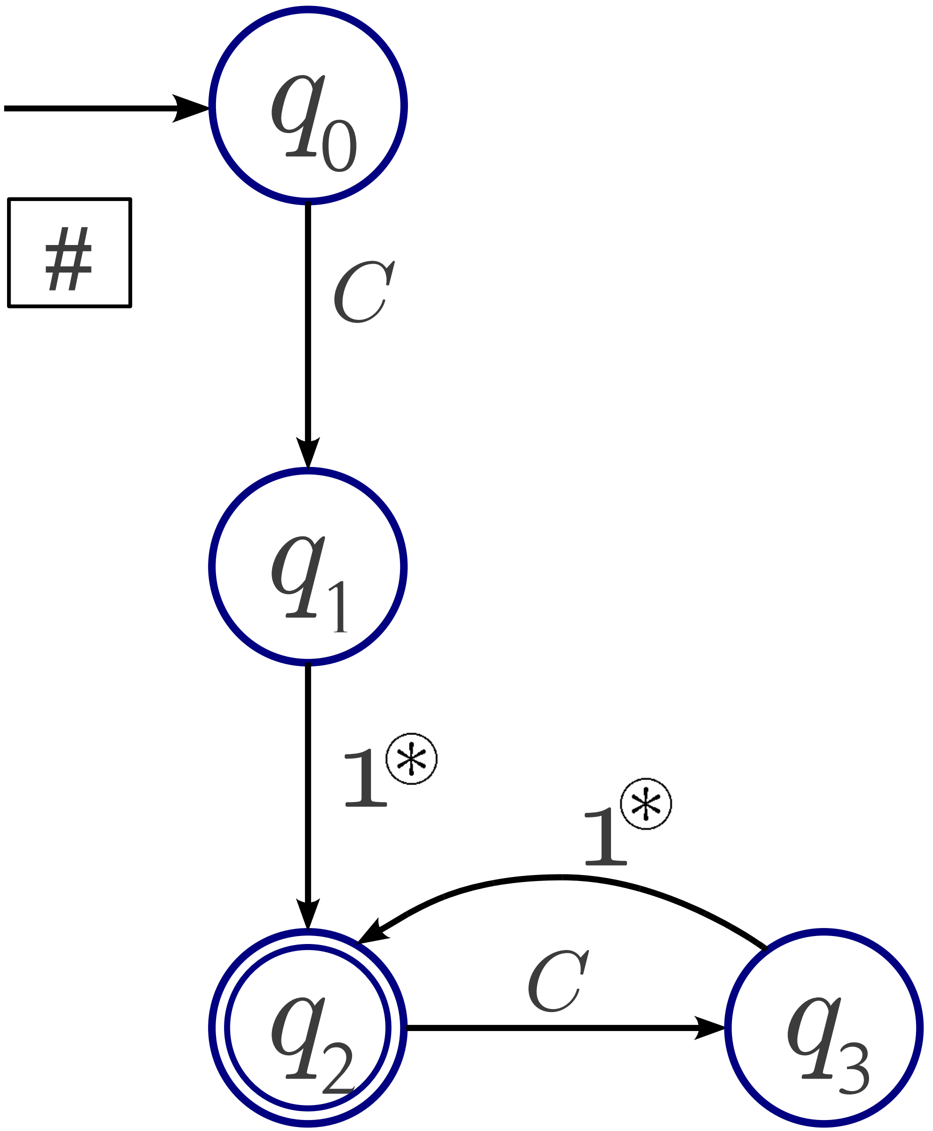


$$\sigma_0 = \{1 \mapsto \#\}$$

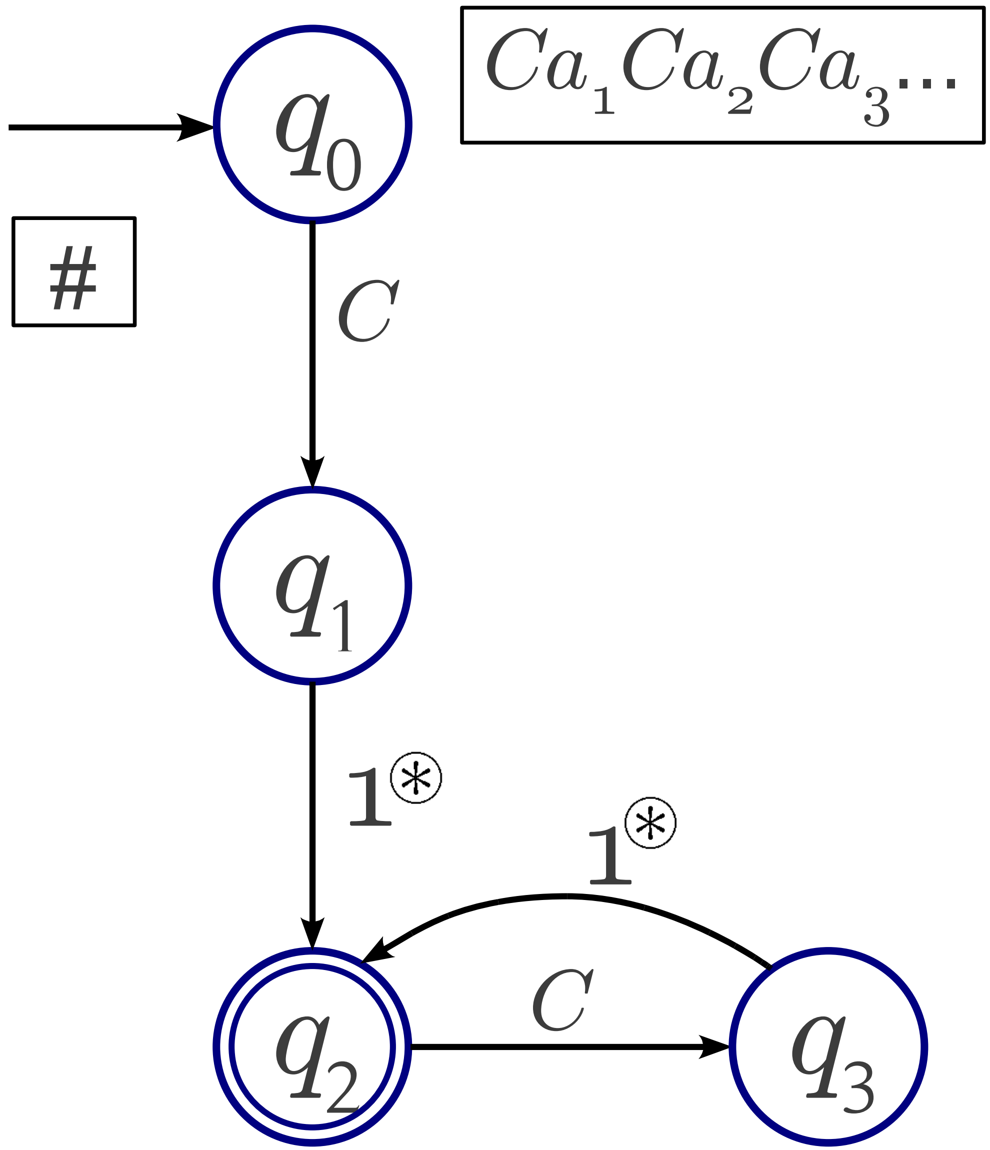
$$\boxed{\mathcal{L}(\mathcal{A}) = \{\, a_1 \cdots a_k \in \mathbb{A}^* \mid \forall i \neq j.\ a_i \neq a_j \,\}}$$



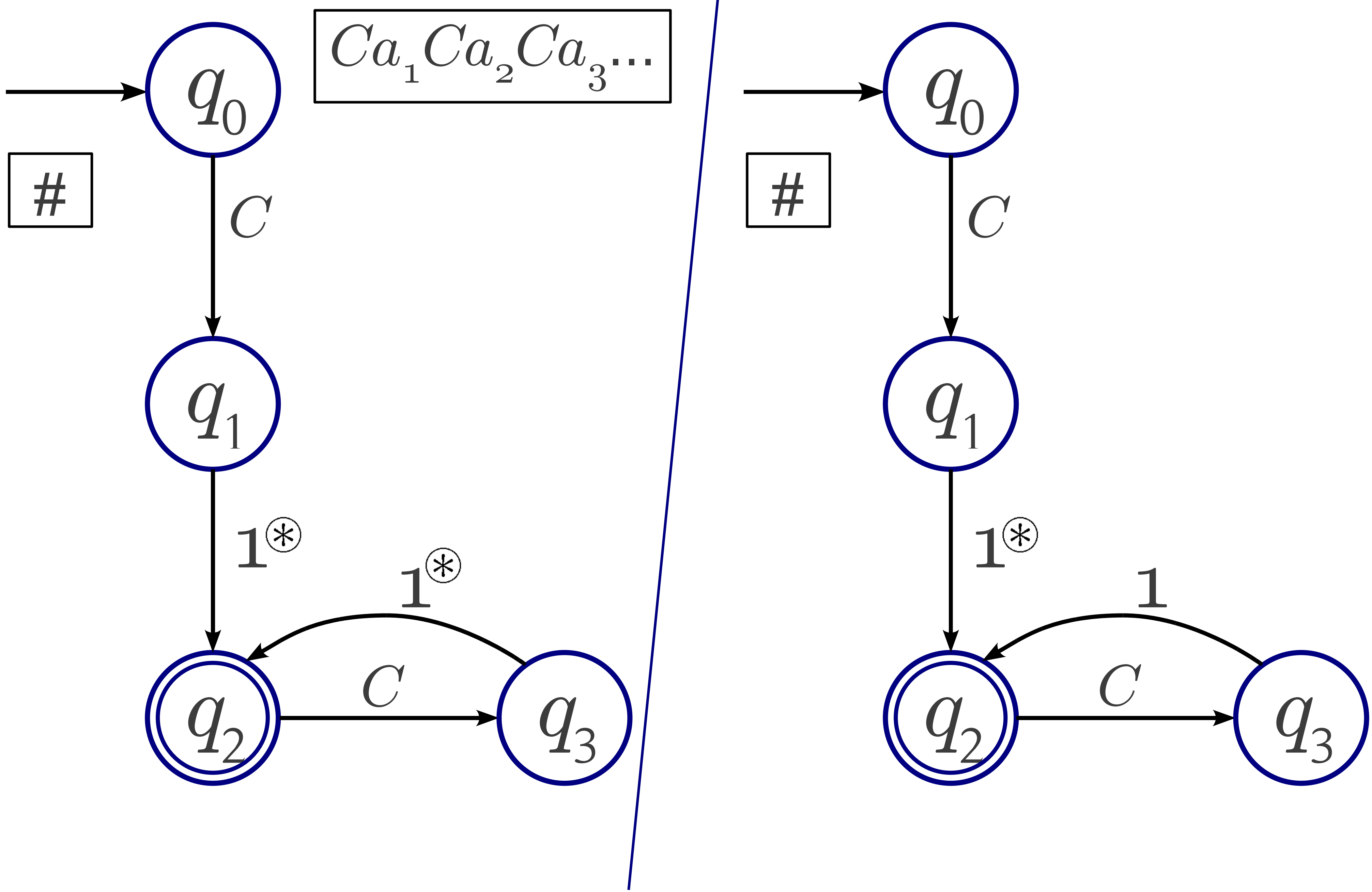
Another example



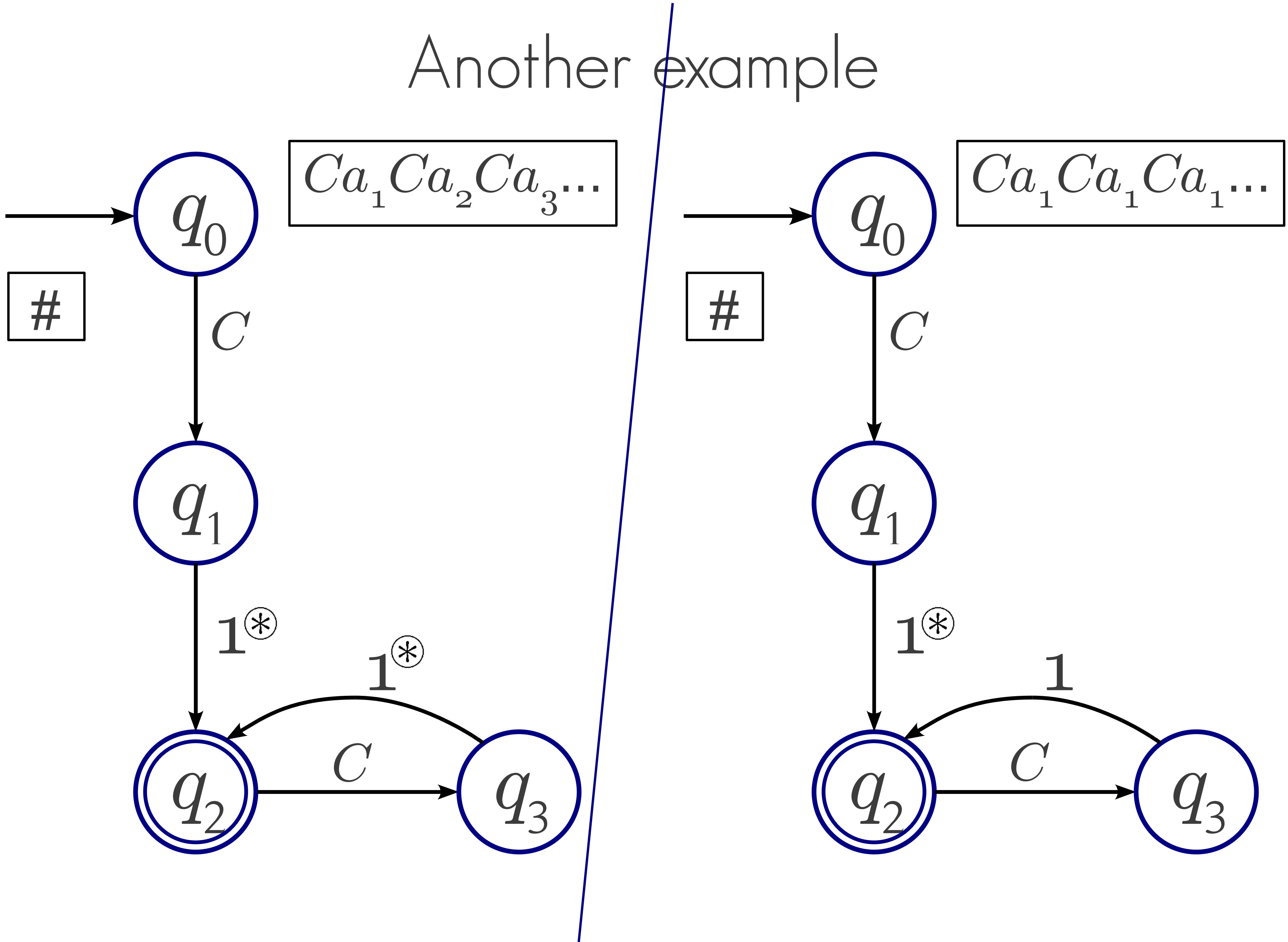
Another example

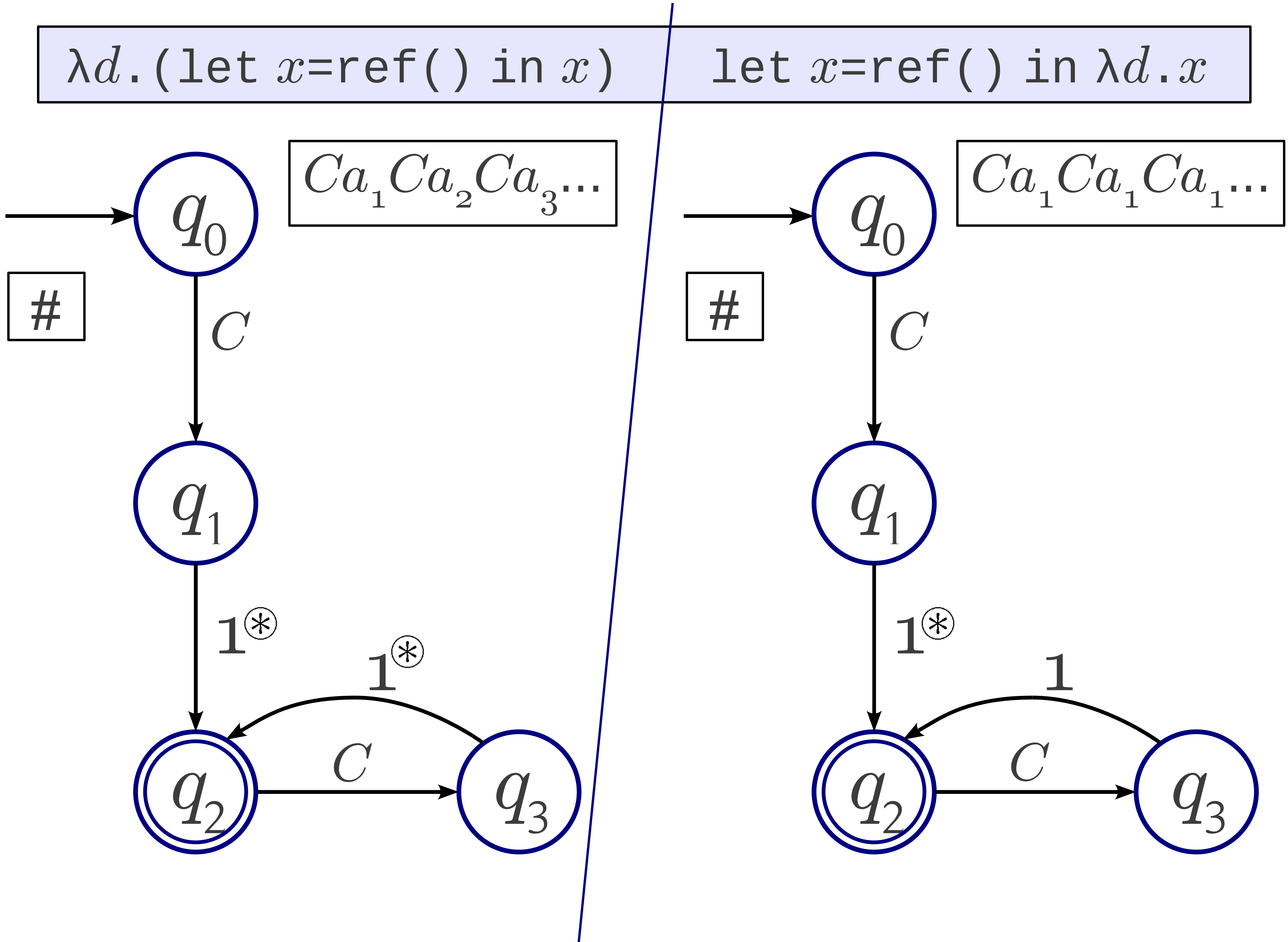


Another example



Another example




λd.(let x=ref() in x)
let x=ref() in λd.x
Ca1Ca2Ca3...
Ca1Ca1Ca1...
#
q0
q1
q2
q3
C
1⊛
1
#
q0
q1
q2
q3
C
1⊛
1

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.

Nominal 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.

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.

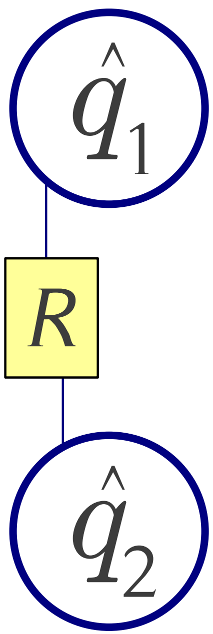
  E.g. $\overline{L_1 * L_1}$ is FRA-recognisable.
- Emptiness is decidable.
- Containment, universality are undecidable.

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 $R \subseteq \hat{Q}_1 \times \hat{Q}_2$

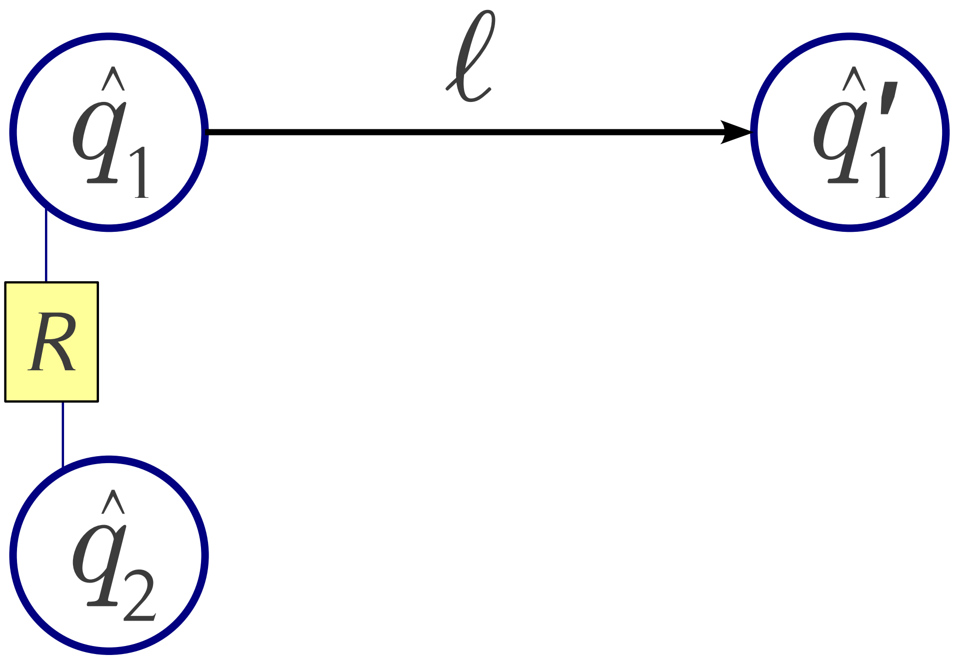
Bisimulations

- 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 $R \subseteq \hat{Q}_1 \times \hat{Q}_2$



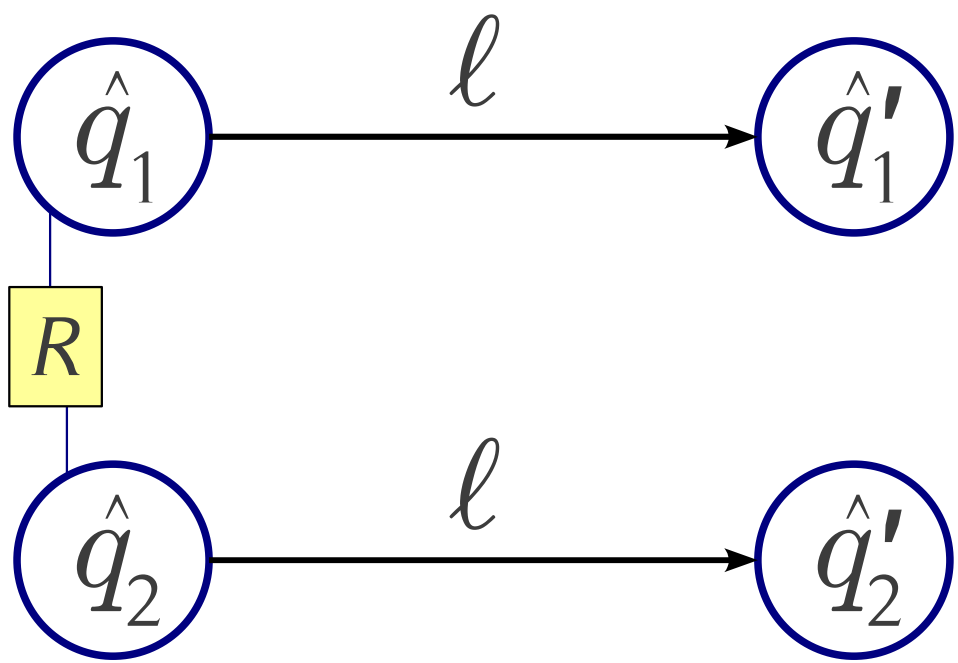
Bisimulations

- 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 $R \subseteq \hat{Q}_1 \times \hat{Q}_2$



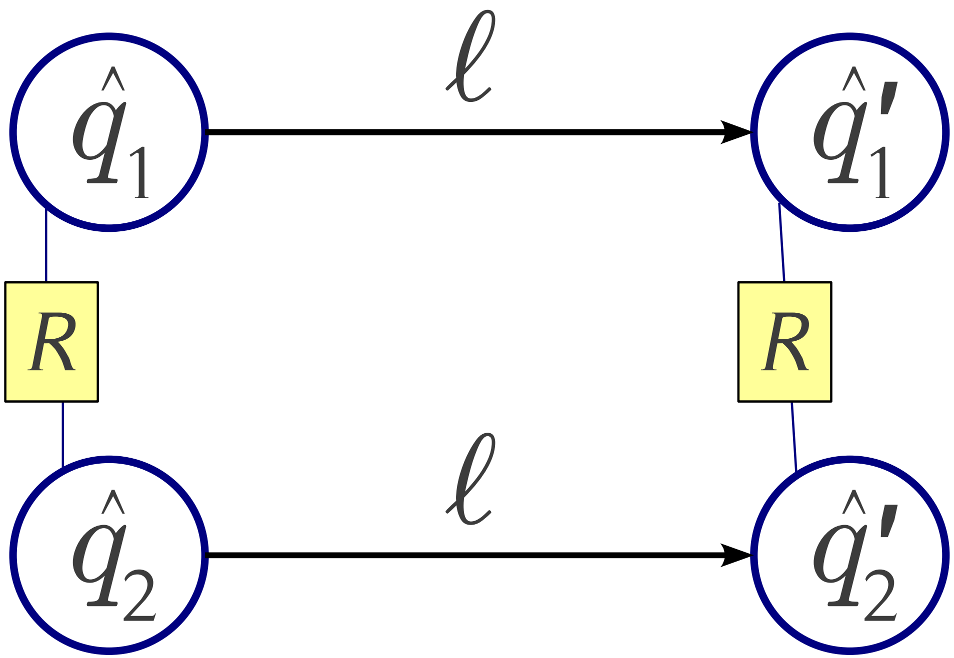
Bisimulations

- 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 $R \subseteq \hat{Q}_1 \times \hat{Q}_2$



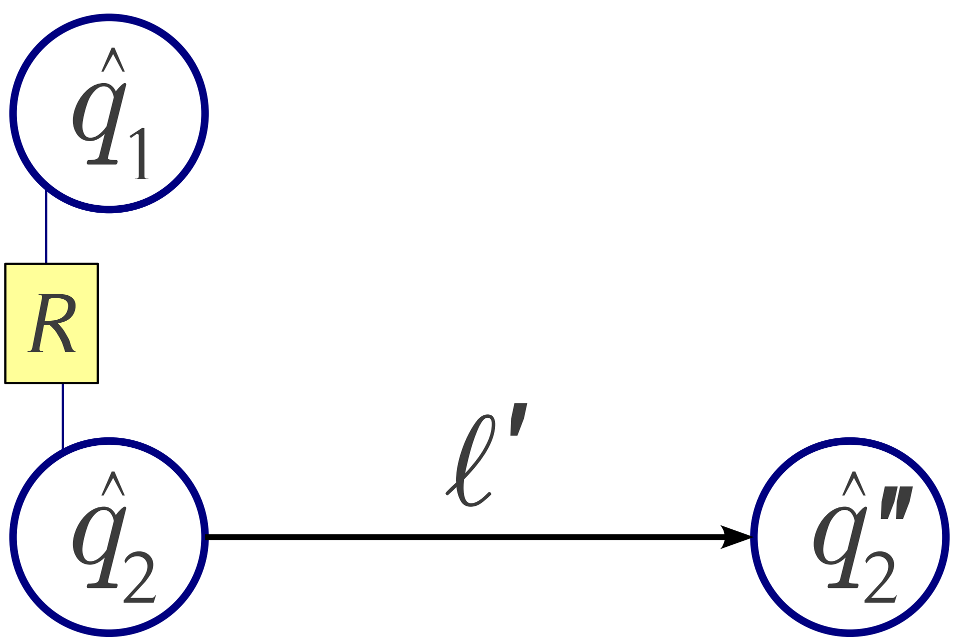
Bisimulations

- 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 $R \subseteq \hat{Q}_1 \times \hat{Q}_2$



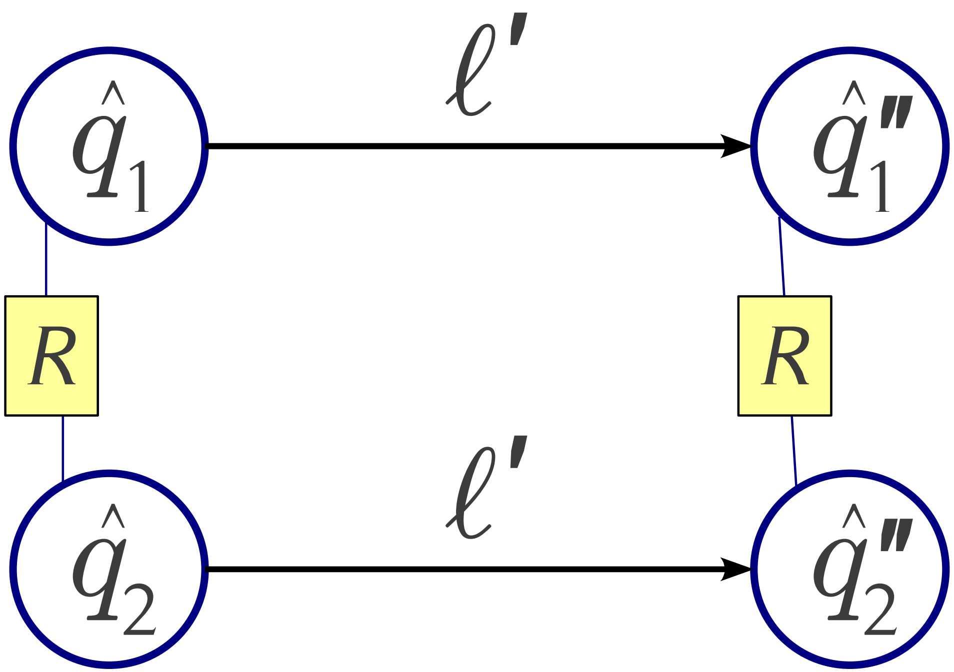
Bisimulations

- 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 $R \subseteq \hat{Q}_1 \times \hat{Q}_2$



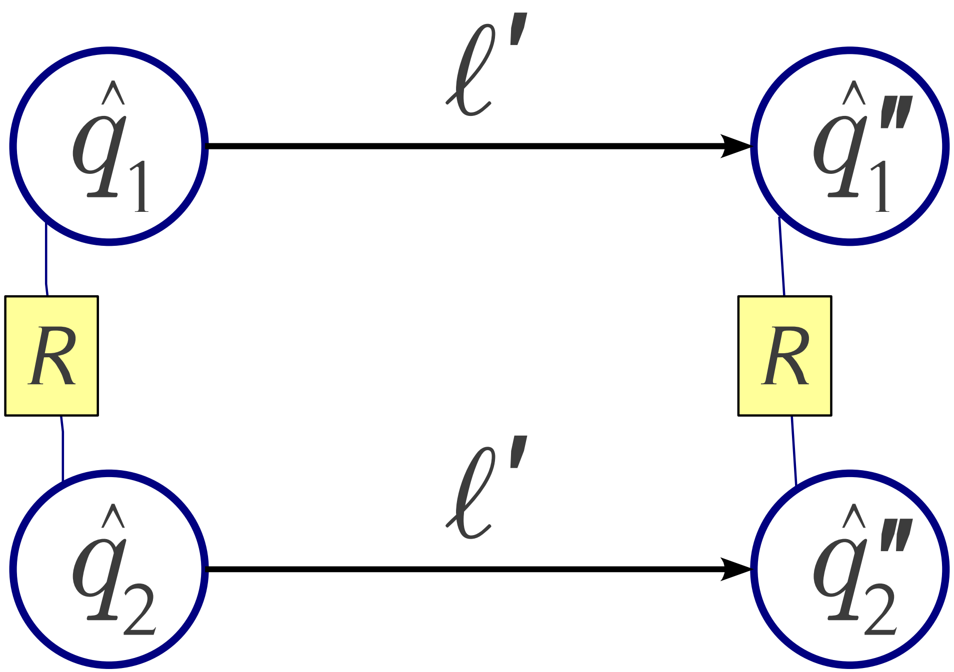
Bisimulations

- 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 $R \subseteq \hat{Q}_1 \times \hat{Q}_2$



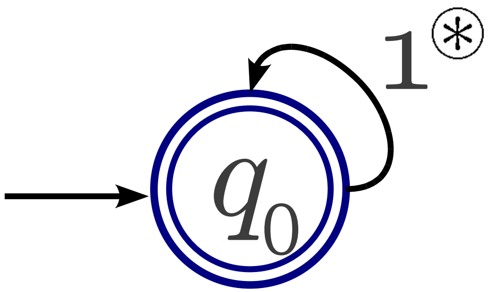
Bisimulations

- 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 $R \subseteq \hat{Q}_1 \times \hat{Q}_2$

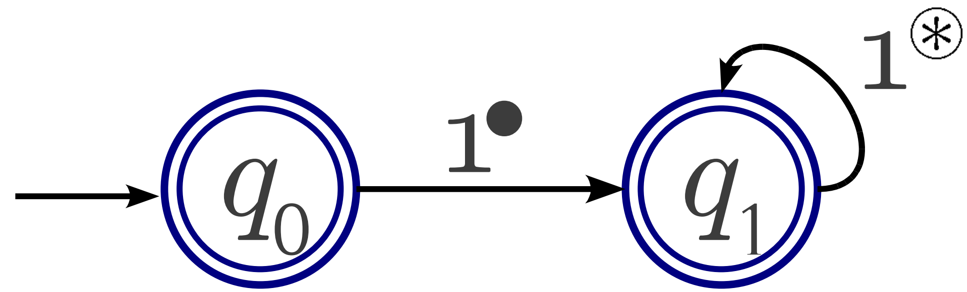


- Bisimilarity implies language equivalence.

Example



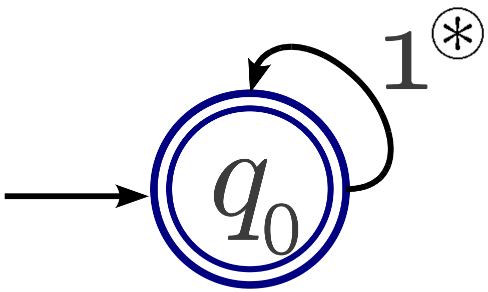
$\sim$



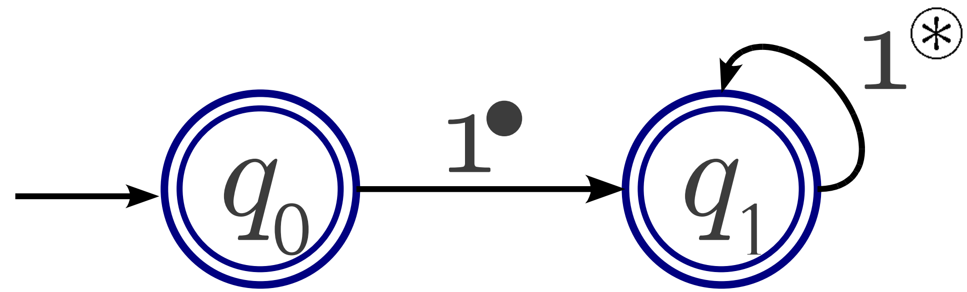
$\sigma_0 = \{1 \mapsto \#\}$

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

Example



$\sim$



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

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

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

Symbolic bisimulations

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

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

Symbolic bisimulations

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

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

typed spans: (S_1, ρ, S_2)

$$\frac{\rho : \{1, \ldots, n_1\} \overset{\cong}{\rightharpoonup} \{1, \ldots, n_2\}}{\begin{array}{l} \mathsf{dom}(\rho) \subseteq S_1 \subseteq \{1, \ldots, n_1\} \\ \mathsf{img}(\rho) \subseteq S_2 \subseteq \{1, \ldots, n_2\} \end{array}}$$

Symbolic bisimulations

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

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

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

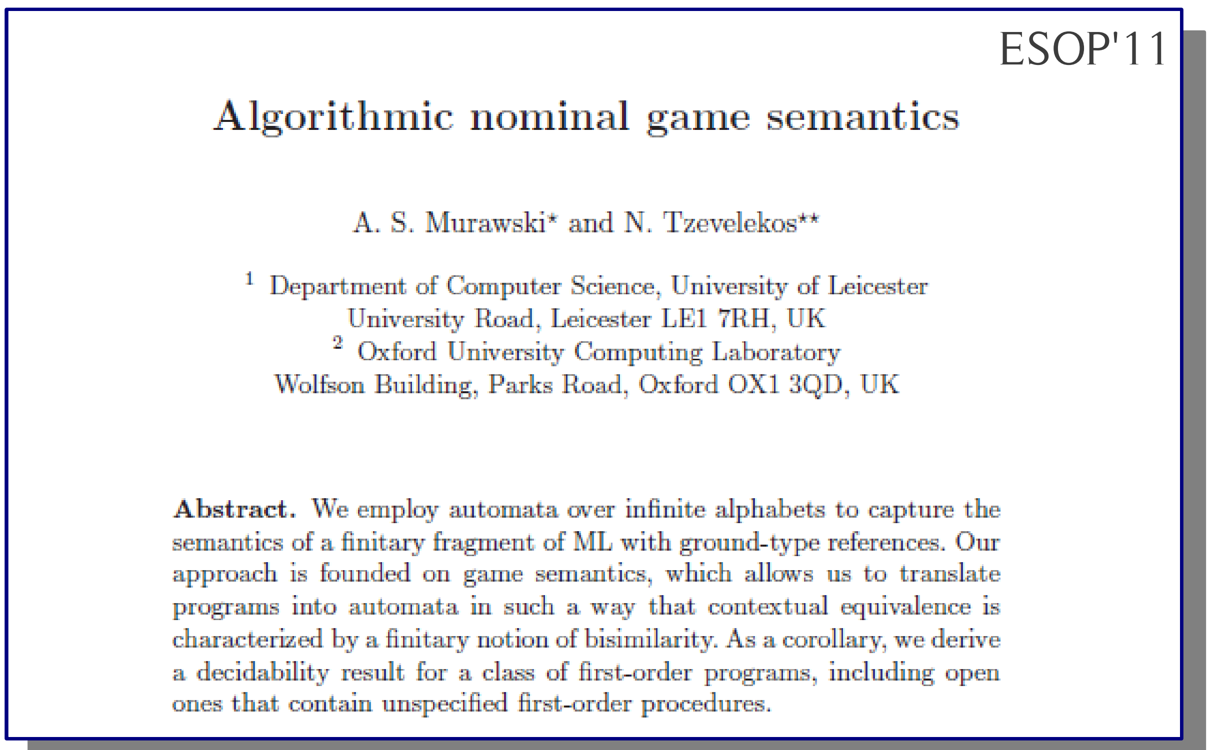
- Algorithmic description which is:
  - name-free
  - finitely-branching
- Bisimilarity can be captured symbolically
  - In the finitary case: decide bisimilarity

Interesting directions

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

Interesting directions

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



ESOP'11

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!