Introduction to the Theory of Computation

Automata

1VO + 1PS

Lecturer: Dr. Ana Sokolova

http://cs.uni-salzburg.at/~anas/

Setup and Dates

- Lectures Tuesday 10:45 pm - 12:15 pm

  Instructions Tuesday 12:30 pm - 2 pm

21.10.	4.11.	18.11.	2.12.	16.12.	13.1.	27.1.

- Books

  Introduction to the Theory of Computation by M. Sipser

  Introduction to Automata Theory, Languages, and Computation by J. E. Hopcroft, R. Motwani, and J.D. Ullman

Setup and Dates

- **Lectures** Tuesday 10:45 pm - 12:15 pm

  Every "second" Tuesday starting today

  Instructions Tuesday 12:30 pm - 2 pm

21.10.	4.11.	18.11.	2.12.	16.12.	13.1.	27.1.

- **Books**

  Introduction to the Theory of Computation by M. Sipser

  Introduction to Automata Theory, Languages, and Computation by J. E. Hopcroft, R. Motwani, and J.D. Ullman

Setup and Dates

- Lectures Tuesday 10:45 pm - 12:15 pm

  Every "second" Tuesday starting today

  Instructions Tuesday 12:30 pm - 2 pm

21.10.	4.11.	18.11.	2.12.	16.12.	13.1.	27.1.

automata

- Books

  Introduction to the Theory of Computation by M. Sipser

  Introduction to Automata Theory, Languages, and Computation by J. E. Hopcroft, R. Motwani, and J.D. Ullman

Setup and Dates

- **Lectures** Tuesday 10:45 pm - 12:15 pm

  Every "second" Tuesday starting today

  Instructions Tuesday 12:30 pm - 2 pm

21.10.	4.11.	18.11.	2.12.	16.12.	13.1.	27.1.

automata

grammars

- **Books**

  Introduction to the Theory of Computation by M. Sipser

  Introduction to Automata Theory, Languages, and Computation by J. E. Hopcroft, R. Motwani, and J.D. Ullman

Setup and Dates

- **Lectures** Tuesday 10:45 pm - 12:15 pm

  Every "second" Tuesday starting today

  Instructions Tuesday 12:30 pm - 2 pm

21.10.	4.11.	18.11.	2.12.	16.12.	13.1.	27.1.

automata

grammars

push-down automata

- **Books**

  Introduction to the Theory of Computation by M. Sipser

  Introduction to Automata Theory, Languages, and Computation by J. E. Hopcroft, R. Motwani, and J.D. Ullman

Setup and Dates

- **Lectures** Tuesday 10:45 pm - 12:15 pm

  Every "second" Tuesday starting today

  Instructions Tuesday 12:30 pm - 2 pm

21.10.	4.11.	18.11.	2.12.	16.12.	13.1.	27.1.

automata

grammars

push-down automata

Turing machines

- **Books**

  Introduction to the Theory of Computation by M. Sipser

  Introduction to Automata Theory, Languages, and Computation by J. E. Hopcroft, R. Motwani, and J.D. Ullman

Setup and Dates

- **Lectures** Tuesday 10:45 pm - 12:15 pm

  Instructions Tuesday 12:30 pm - 2 pm

Every "second" Tuesday starting today

21.10.	4.11.	18.11.	2.12.	16.12.	13.1.	27.1.

automata

grammars

push-down automata

Turing machines

computability

- **Books**

  Introduction to the Theory of Computation by M. Sipser

  Introduction to Automata Theory, Languages, and Computation by J. E. Hopcroft, R. Motwani, and J.D. Ullman

The Rules... Instructions

- Instruction exercises on the web
  http://cs.uni-salzburg.at/~anas/Ana_Sokolova/Automata2014.html
  on Tuesday afternoons, after class
- To be solved by the students (in groups of at most 3 students) and handed in as homework at the next meeting.
- In class I will present a sample solution and the students will be asked to present solutions/discuss the exercises

The Rules... Instructions

- One randomly chosen exercise will be graded each week
- The graded exercise will be returned at the next meeting.
- Grade based on
  (1) exam
  (2) the grades of the corrected exercise and
  (3) activity in class (ability to present solutions)
- All information about the course / rules / exams / grading is / will be on the course webpage

The Rules... Grading

- Written exam on January 27, 10:45 am - 12:15 pm
- Grade based on the number of points on the written exam (80%), homework grades and activity in class (20%)
- For better grade oral exam after the written one upon appointment
- 55% of the maximal points are needed to pass.

Finite Automata

Alphabets and Languages

Alphabet and words

Alphabets and Languages

Alphabet and words

Σ - alphabet (finite set)

$\Sigma^n = \{a_1a_2..a_n \mid a_i \in \Sigma\}$ is the set of words of length n

$\Sigma^* = \{w \mid \exists n \in \mathbb{N}.\ \exists\ a_1, a_2, .., a_n \in \Sigma.\ w = a_1a_2..a_n\}$ is the set of all words over Σ

Alphabets and Languages

Alphabet and words

$\Sigma^0 = \{\varepsilon\}$ contains only the empty word

Σ - alphabet (finite set)

$\Sigma^n = \{a_1 a_2 .. a_n \mid a_i \in \Sigma\}$ is the set of words of length n

$\Sigma^* = \{w \mid \exists n \in \mathbb{N}.\ \exists\ a_1, a_2, .. , a_n \in \Sigma.\ w = a_1 a_2 .. a_n\}$ is the set of all words over Σ

Alphabets and Languages

Alphabet and words

$\Sigma^0 = \{\varepsilon\}$ contains only the empty word

Σ - alphabet (finite set)

$\Sigma^n = \{a_1 a_2 .. a_n \mid a_i \in \Sigma\}$ is the set of words of length n

$\Sigma^* = \{w \mid \exists n \in \mathbb{N}. \exists\, a_1, a_2, .., a_n \in \Sigma. w = a_1 a_2 .. a_n\}$ is the set of all words over Σ

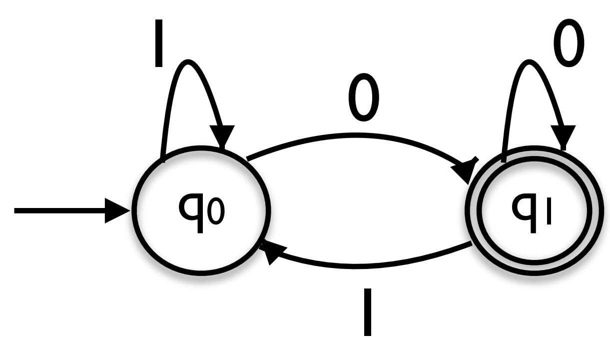
A language L over Σ is a subset $L \subseteq \Sigma^*$

Deterministic Automata (DFA)

Informal example

$\Sigma = \{0,1\}$

M_1:



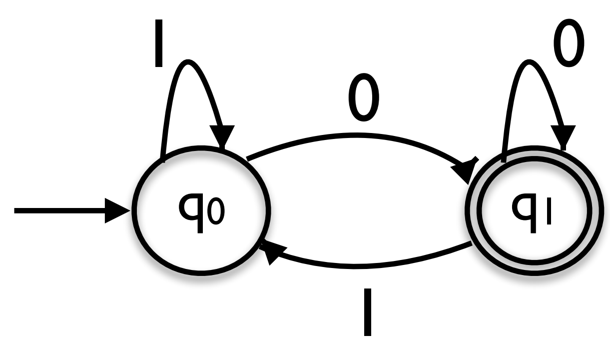
Deterministic Automata (DFA)

alphabet

Informal example

$\Sigma = \{0,1\}$

M_1:



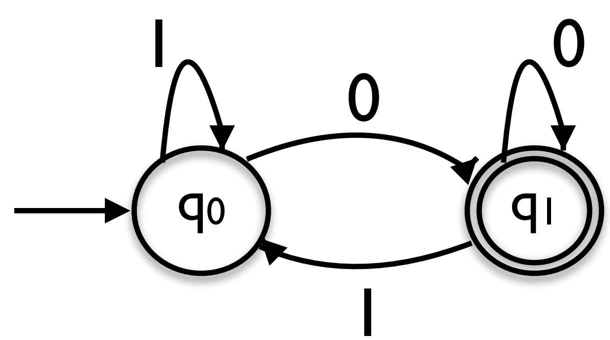
Deterministic Automata (DFA)

alphabet

Informal example

$\Sigma = \{0,1\}$

M_1:



q_0, q_1 are states

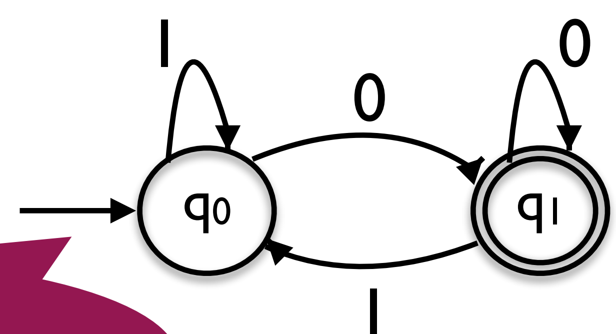
Deterministic Automata (DFA)

alphabet

Informal example

$\Sigma = \{0,1\}$

M_1:



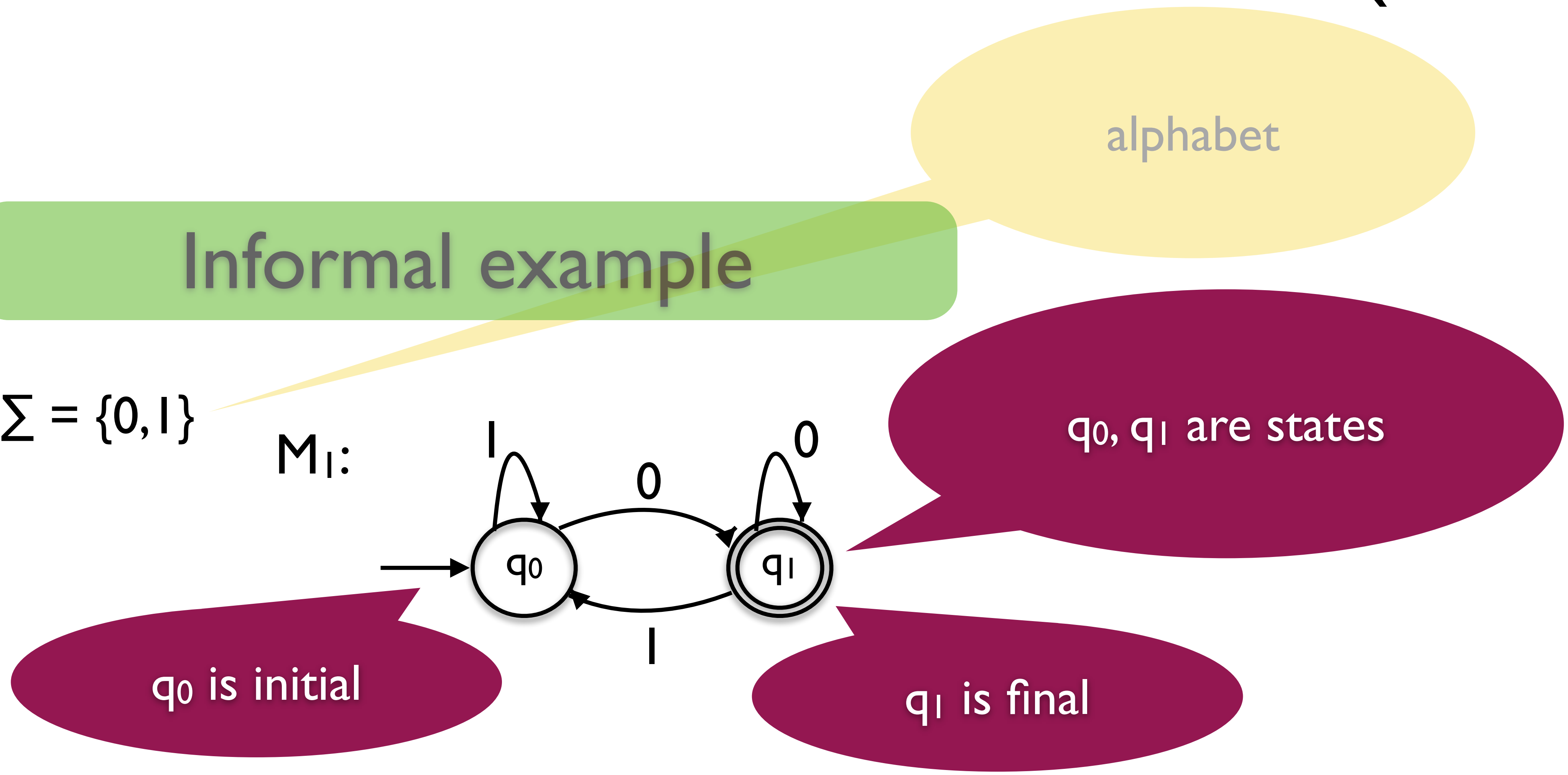
q_0, q_1 are states

q_0 is initial

Deterministic Automata (DFA)

Informal example

$\Sigma = \{0,1\}$

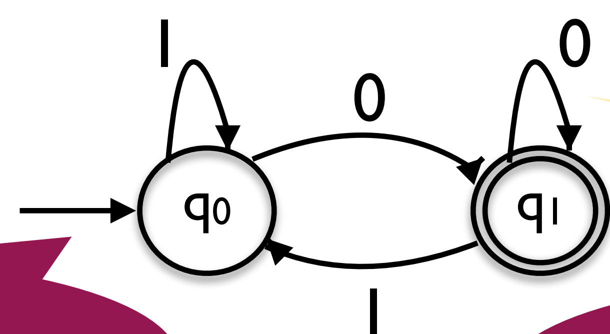


Deterministic Automata (DFA)

Informal example

$\Sigma = \{0,1\}$

M_1:



alphabet

q0, q1 are states

transitions, labelled by alphabet symbols

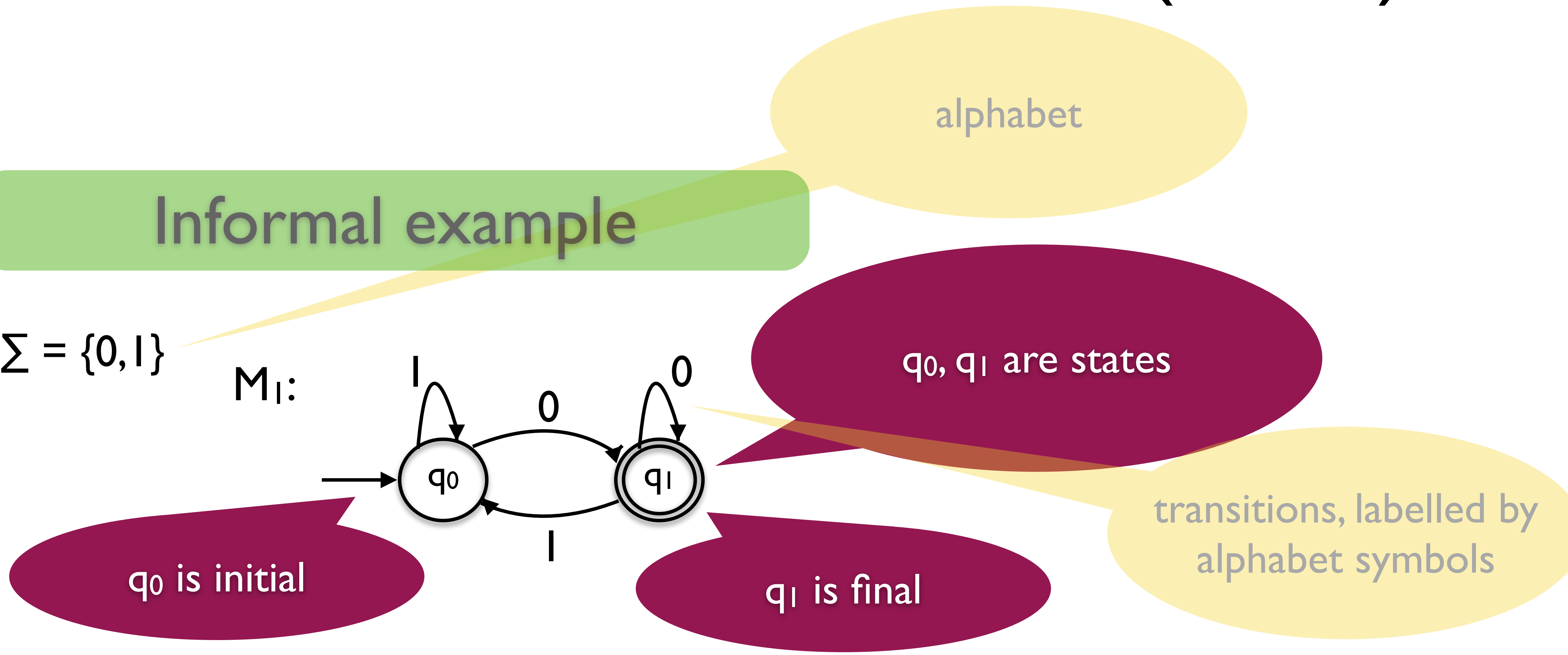
q0 is initial

q1 is final

Deterministic Automata (DFA)

Informal example

$\Sigma = \{0,1\}$



Accepts the language $L(M_1) = \{w \in \Sigma^* \mid w \text{ ends with a } 0\} = \Sigma^*0$

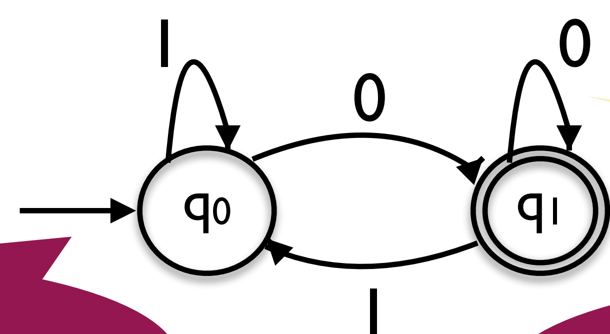
Deterministic Automata (DFA)

Informal example

alphabet

$\Sigma = \{0,1\}$

M_1:



q_0, q_1 are states

transitions, labelled by alphabet symbols

q_0 is initial

q_1 is final

Accepts the language $L(M_1) = \{w \in \Sigma^* \mid w \text{ ends with a } 0\} = \Sigma^*0$

regular language

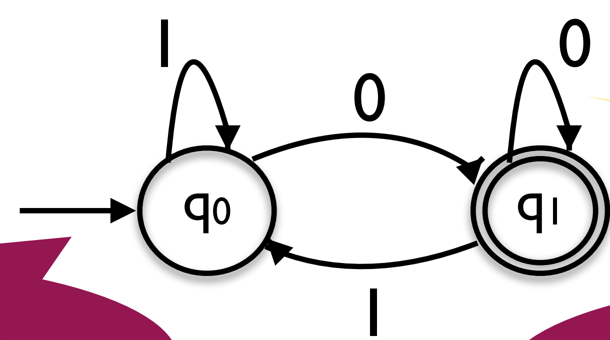
Deterministic Automata (DFA)

Informal example

alphabet

$\Sigma = \{0,1\}$

M_1:



q_0, q_1 are states

transitions, labelled by alphabet symbols

q_0 is initial

q_1 is final

Accepts the language $L(M_1) = \{w \in \Sigma^* \mid w \text{ ends with a } 0\} = \Sigma^*0$

regular language

regular expression

DFA

Definition

A deterministic finite automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma \longrightarrow Q$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

DFA

Definition

A deterministic finite automaton M is a tuple M = $(Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma \longrightarrow Q$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

In the example M

DFA

Definition

A deterministic finite automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma \longrightarrow Q$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

In the example M

$M_1 = (Q, \Sigma, \delta, q_0, F)$ for

DFA

Definition

A deterministic finite automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma \longrightarrow Q$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

In the example M

$M_1 = (Q, \Sigma, \delta, q_0, F)$ for

$Q = \{q_0, q_1\}$

DFA

Definition

A deterministic finite automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma \longrightarrow Q$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

In the example M

$M_1 = (Q, \Sigma, \delta, q_0, F)$ for

$Q = \{q_0, q_1\}$

$\Sigma = \{0, 1\}$

DFA

Definition

A deterministic finite automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma \longrightarrow Q$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

In the example M

$M_1 = (Q, \Sigma, \delta, q_0, F)$ for

$Q = \{q_0, q_1\}$ $F = \{q_1\}$

$\Sigma = \{0, 1\}$

DFA

Definition

A deterministic finite automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma \longrightarrow Q$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

In the example M

$M_1 = (Q, \Sigma, \delta, q_0, F)$ for

$Q = \{q_0, q_1\}$ $F = \{q_1\}$

$\Sigma = \{0, 1\}$

$\delta(q_0, 0) = q_1$, $\delta(q_0, 1) = q_0$
$\delta(q_1, 0) = q_1$, $\delta(q_1, 1) = q_0$

DFA

The extended transition function

DFA

The extended transition function

Given $M = (Q, \Sigma, \delta, q_0, F)$ we can extend $\delta: Q \times \Sigma \longrightarrow Q$ to

$\delta^*: Q \times \Sigma^* \longrightarrow Q$

inductively, by:

$\delta^*(q, \varepsilon) = q$ and $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

DFA

The extended transition function

Given $M = (Q, \Sigma, \delta, q_0, F)$ we can extend $\delta: Q \times \Sigma \longrightarrow Q$ to

$\delta^*: Q \times \Sigma^* \longrightarrow Q$

In M_1, $\delta^*(q_0, 110010) = q_1$

inductively, by:

$\delta^*(q, \varepsilon) = q$ and $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

DFA

The extended transition function

Given $M = (Q, \Sigma, \delta, q_0, F)$ we can extend $\delta: Q \times \Sigma \longrightarrow Q$ to

$\delta^*: Q \times \Sigma^* \longrightarrow Q$

In M_1, $\delta^*(q_0, 110010) = q_1$

inductively, by:

$\delta^*(q, \varepsilon) = q$ and $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

Definition

The language recognised / accepted by a deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$$

DFA

The extended transition function

Given $M = (Q, \Sigma, \delta, q_0, F)$ we can extend $\delta: Q \times \Sigma \longrightarrow Q$ to

$\delta^*: Q \times \Sigma^* \longrightarrow Q$

In M_1, $\delta^*(q_0, 110010) = q_1$

inductively, by:

$\delta^*(q, \varepsilon) = q$ and $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

Definition

$L(M_1) = \{w0 \mid w \in \{0,1\}^*\}$

The language recognised / accepted by a deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$ is

$$L(M) = \{w \in \Sigma^* \mid \delta^*(q_0, w) \in F\}$$

Regular languages and operations

Definition

Let Σ be an alphabet. A language L over Σ ($L \subseteq \Sigma^*$) is regular iff it is recognised by a DFA.

Regular languages and operations

$L(M_1) = \{w0 | w \in \{0,1\}^*\}$
is regular

Definition

Let Σ be an alphabet. A language L over Σ ($L \subseteq \Sigma^*$) is regular iff it is recognised by a DFA.

Regular languages and operations

$L(M_1) = \{w0 | w \in \{0,1\}^*\}$
is regular

Definition

Let Σ be an alphabet. A language L over Σ ($L \subseteq \Sigma^*$) is regular iff it is recognised by a DFA.

Regular operations

Regular languages and operations

$L(M_1) = \{w0 | w \in \{0,1\}^*\}$
is regular

Definition

Let Σ be an alphabet. A language L over Σ ($L \subseteq \Sigma^*$) is regular iff it is recognised by a DFA.

Regular operations

Let L, L_1, L_2 be languages over Σ. Then
$L_1 \cup L_2$, $L_1 \cdot L_2$, and L^* are languages, where

$$L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

$$L^* = \{w \mid \exists n \in \mathbb{N}.\ \exists\, w_1, w_2, .., w_n \in L.\ w = w_1 w_2 .. w_n\}$$

Regular languages and operations

$L(M_1) = \{w0 | w \in \{0,1\}^*\}$
is regular

Definition

Let Σ be an alphabet. A language L over Σ ($L \subseteq \Sigma^*$) is regular iff it is recognised by a DFA.

Regular operations

Let L, L_1, L_2 be languages over Σ. Then $L_1 \cup L_2$, $L_1 \cdot L_2$, and L^* are languages, where

$\varepsilon \in L^*$ always

$$L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1, w_2 \in L_2\}$$

$$L^* = \{w \mid \exists n \in \mathbb{N}.\ \exists\ w_1, w_2, .., w_n \in L.\ w = w_1 w_2 .. w_n\}$$

Regular expressions

Definition

finite representation of infinite languages

Regular expressions

Definition

finite representation of infinite languages

Regular expressions

Definition

Let Σ be an alphabet. The following are regular expressions

1. a for $a \in \Sigma$
2. ε
3. $\emptyset$
4. $(R_1 \cup R_2)$ for R_1, R_2 regular expressions
5. $(R_1 \cdot R_2)$ for R_1, R_2 regular expressions
6. $(R_1)^*$ for R_1 regular expression

finite representation of infinite languages

Regular expressions

inductive

Definition

Let Σ be an alphabet. The following are regular expressions

1. a for $a \in \Sigma$
2. ε
3. $\varnothing$
4. $(R_1 \cup R_2)$ for R_1, R_2 regular expressions
5. $(R_1 \cdot R_2)$ for R_1, R_2 regular expressions
6. $(R_1)^*$ for R_1 regular expression

finite representation of infinite languages

Regular expressions

inductive

example: $(ab \cup a)^*$

Definition

Let Σ be an alphabet. The following are regular expressions

1. a for $a \in \Sigma$
2. ε
3. $\varnothing$
4. $(R_1 \cup R_2)$ for R_1, R_2 regular expressions
5. $(R_1 \cdot R_2)$ for R_1, R_2 regular expressions
6. $(R_1)^*$ for R_1 regular expression

finite representation of infinite languages

Regular expressions

inductive

example:
$(ab \cup a)^*$

Definition

Let Σ be an alphabet. The following are regular expressions

1. a for $a \in \Sigma$
2. ε
3. $\emptyset$
4. $(R_1 \cup R_2)$ for R_1, R_2 regular expressions
5. $(R_1 \cdot R_2)$ for R_1, R_2 regular expressions
6. $(R_1)^*$ for R_1 regular expression

corresponding languages

$L(a) = \{a\}$

$L(\varepsilon) = \{\varepsilon\}$

$L(\emptyset) = \emptyset$

$L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$

$L(R_1 \cdot R_2) = L(R_1) \cdot L(R_2)$

$L(R_1^*) = L(R_1)^*$

Equivalence of regular expressions and regular languages

Equivalence of regular expressions and regular languages

Theorem (Kleene)

A language is regular (i.e., recognised by a finite automaton) iff it is the language of a regular expression.

Equivalence of regular expressions and regular languages

Theorem (Kleene)

A language is regular (i.e., recognised by a finite automaton) iff it is the language of a regular expression.

Proof $\Leftarrow$ easy, via the closure

properties discussed next,
$\Rightarrow$ not so easy, we'll skip it for now…

Closure under regular operations

Closure under regular operations

Theorem C1

The class of regular languages is closed under union

Closure under regular operations

also under intersection

Theorem C1

The class of regular languages is closed under union

Closure under regular operations

also under intersection

Theorem C1

The class of regular languages is closed under union

Theorem C2

The class of regular languages is closed under complement

Closure under regular operations

also under intersection

Theorem C1

The class of regular languages is closed under union

Theorem C2

The class of regular languages is closed under complement

Theorem C3

The class of regular languages is closed under concatenation

Closure under regular operations

also under intersection

Theorem C1

The class of regular languages is closed under union

Theorem C2

The class of regular languages is closed under complement

Theorem C3

The class of regular languages is closed under concatenation

Theorem C4

The class of regular languages is closed under Kleene star

Closure under regular operations

also under intersection

We can already prove these!

Theorem C1

The class of regular languages is closed under union

Theorem C2

The class of regular languages is closed under complement

Theorem C3

The class of regular languages is closed under concatenation

Theorem C4

The class of regular languages is closed under Kleene star

Closure under regular operations

also under intersection

Theorem C1

The class of regular languages is closed under union

We can already prove these!

Theorem C2

The class of regular languages is closed under complement

Theorem C3

The class of regular languages is closed under concatenation

But not yet these two…

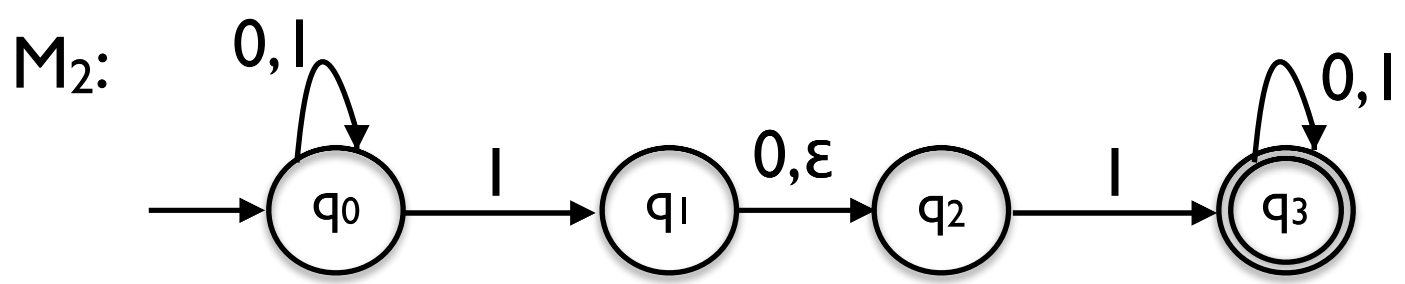
Theorem C4

The class of regular languages is closed under Kleene star

Nondeterministic Automata (NFA)

Informal example

$\Sigma = \{0,1\}$

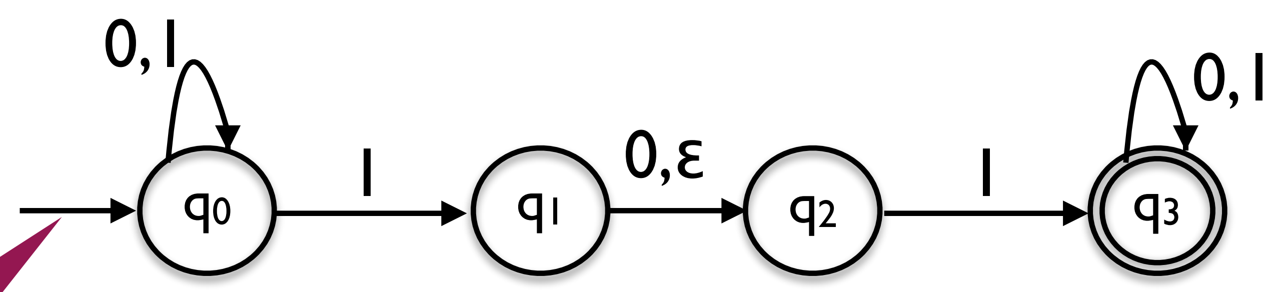


Nondeterministic Automata (NFA)

Informal example

$\Sigma = \{0,1\}$

M_2:

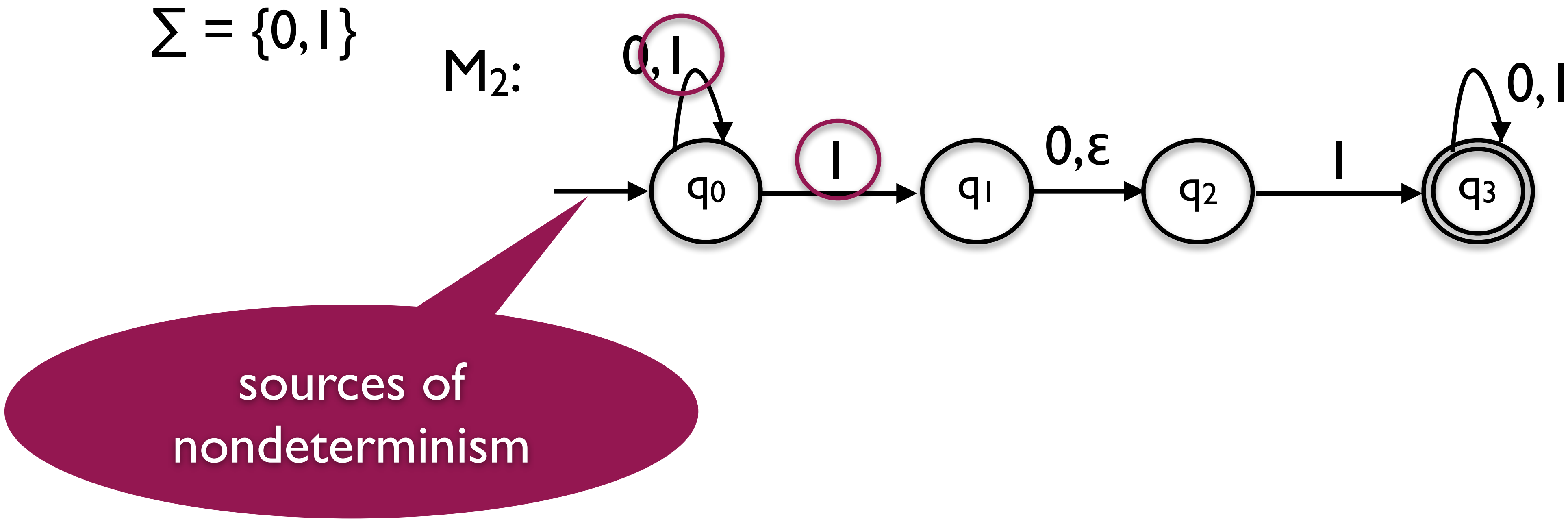


sources of nondeterminism

Nondeterministic Automata (NFA)

Informal example

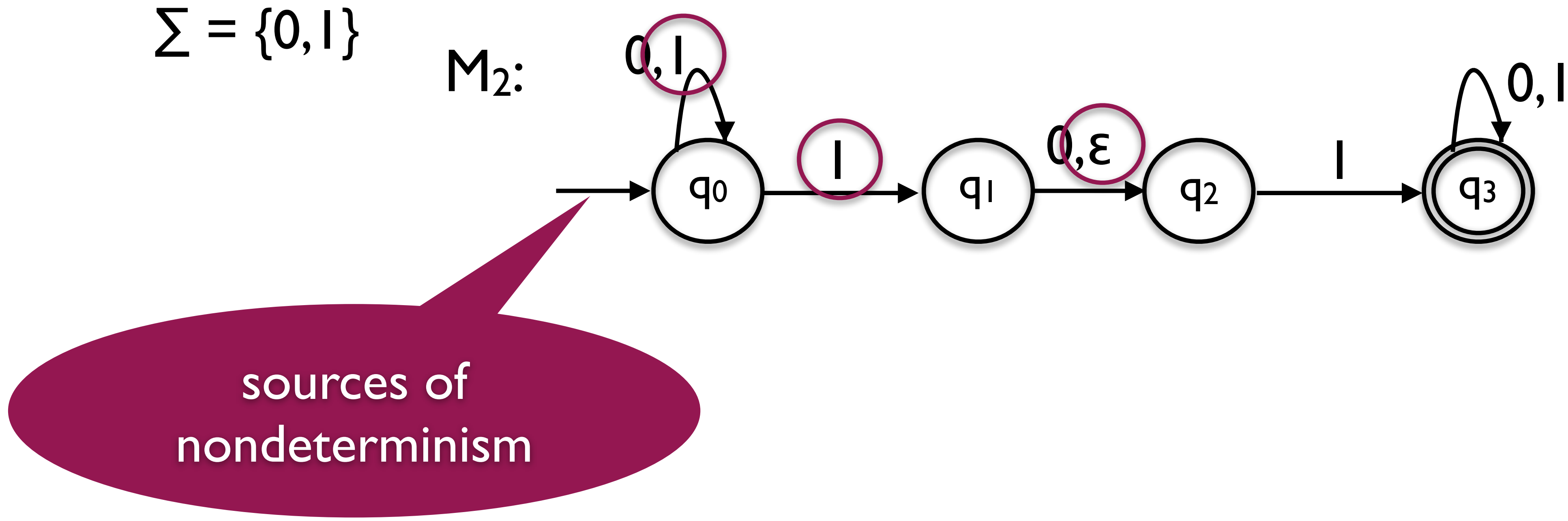
$\Sigma = \{0,1\}$



Nondeterministic Automata (NFA)

Informal example

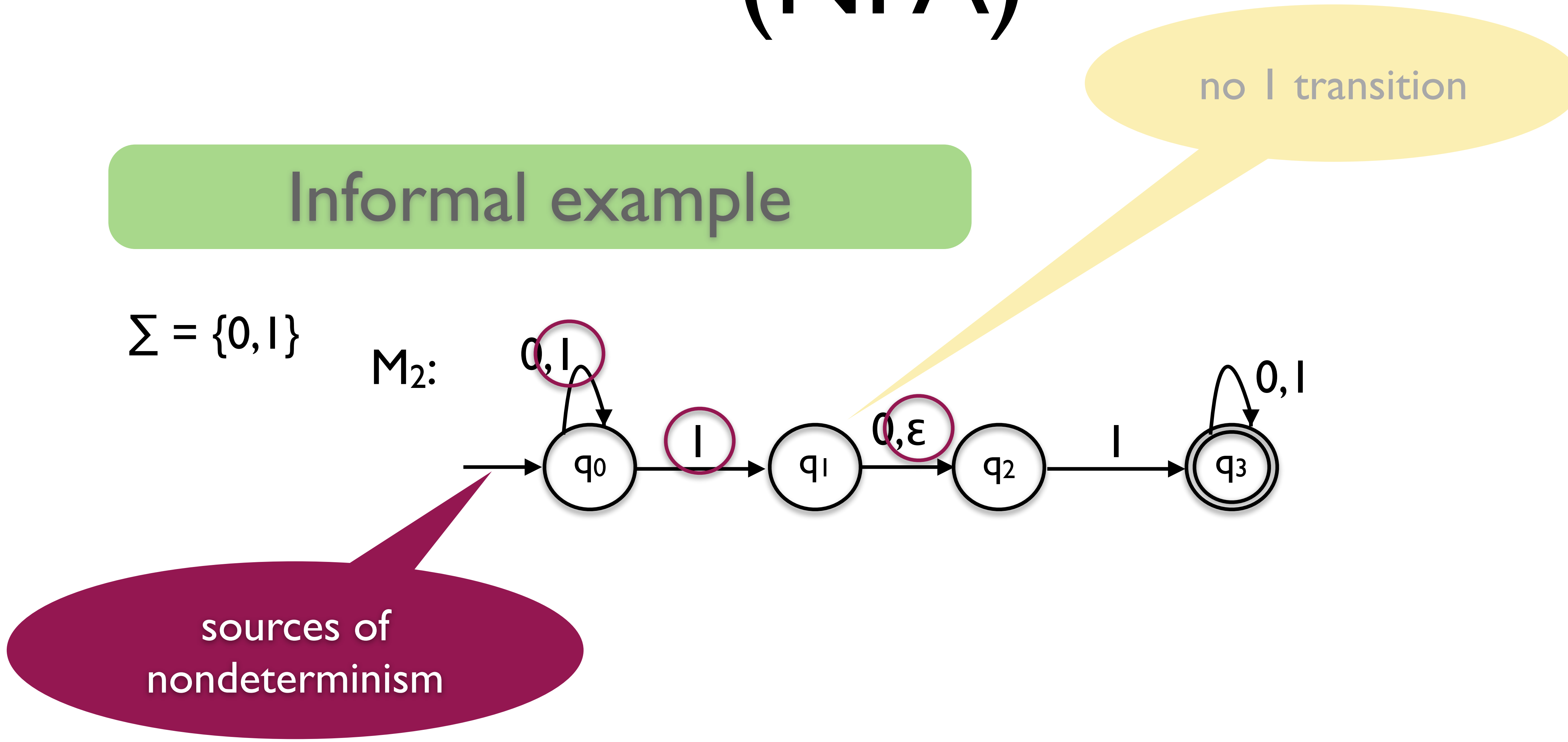
$\Sigma = \{0,1\}$



Nondeterministic Automata (NFA)

Informal example

$\Sigma = \{0,1\}$



Nondeterministic Automata (NFA)

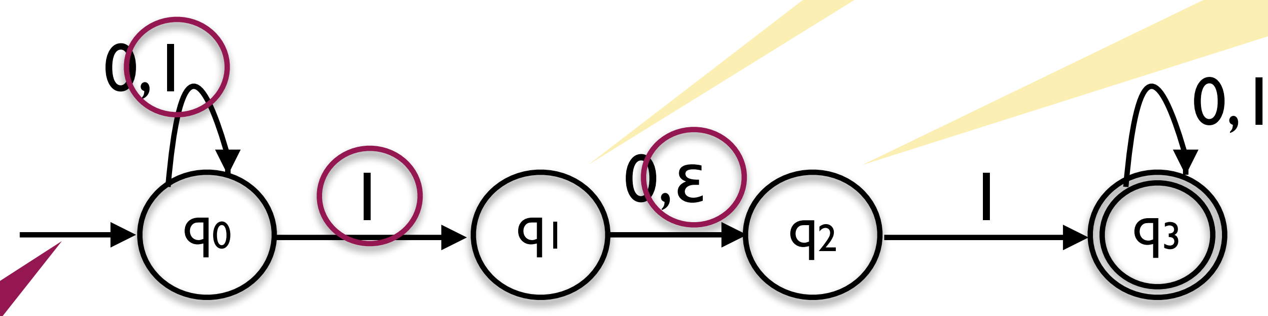
no 1 transition

Informal example

no 0 transition

$\Sigma = \{0,1\}$

M_2:



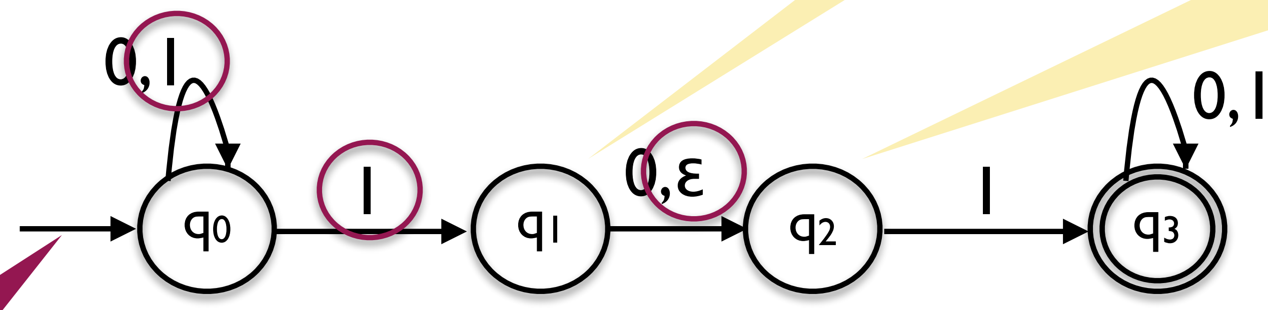
sources of nondeterminism

Nondeterministic Automata (NFA)

Informal example

$\Sigma = \{0,1\}$

M_2:



no 1 transition

no 0 transition

sources of nondeterminism

Accepts a word iff there exists an accepting run

NFA

Definition

A nondeterministic automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

NFA

Definition

A nondeterministic automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

NFA

Definition

A nondeterministic automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

In the example M

NFA

Definition

A nondeterministic automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

In the example M

$M_2 = (Q, \Sigma, \delta, q_0, F)$ for

NFA

Definition

A nondeterministic automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

In the example M

$M_2 = (Q, \Sigma, \delta, q_0, F)$ for

$Q = \{q_0, q_1, q_2, q_3\}$

NFA

Definition

A nondeterministic automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

In the example M

$M_2 = (Q, \Sigma, \delta, q_0, F)$ for

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$

NFA

Definition

A nondeterministic automaton M is a tuple M = $(Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

In the example M

$M_2 = (Q, \Sigma, \delta, q_0, F)$ for

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$ $F = \{q_3\}$

NFA

Definition

A nondeterministic automaton M is a tuple $M = (Q, \Sigma, \delta, q_0, F)$ where

Q is a finite set of states
Σ is a finite alphabet
$\delta: Q \times \Sigma_\varepsilon \longrightarrow \mathcal{P}(Q)$ is the transition function
q_0 is the initial state, $q_0 \in Q$
F is a set of final states, $F \subseteq Q$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

In the example M

$M_2 = (Q, \Sigma, \delta, q_0, F)$ for

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{0, 1\}$ $F = \{q_3\}$

$\delta(q_0, 0) = \{q_0\}$
$\delta(q_0, 1) = \{q_0, q_1\}$
$\delta(q_0, \varepsilon) = \varnothing$
…..