Automata Theory and Formal Languages

Sommer Term 2026

5th Lecture

Regular Languages and Finite Automata

Prof. Dr. Matthias Mnich *Institute for Algorithms and Complexity*

Regular Languages and Automata

Regular Languages

Regular languages (type 3) have been intensively investigated and most interesting theoretical questions have been answered.

Regular Languages

Regular languages (type 3) have been intensively investigated and most interesting theoretical questions have been answered.

They can be described by:

- regular expressions

Regular Languages

Regular languages (type 3) have been intensively investigated and most interesting theoretical questions have been answered.

They can be described by:

- regular expressions
- type 3 grammars

Regular Languages

Regular languages (type 3) have been intensively investigated and most interesting theoretical questions have been answered.

They can be described by:

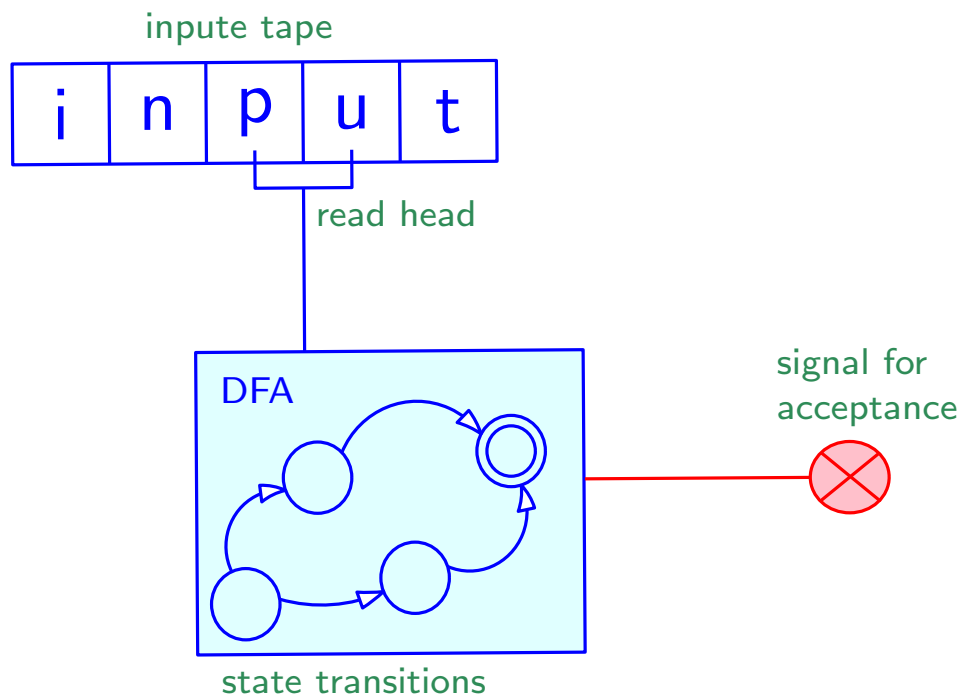
- regular expressions
- type 3 grammars
- finite automata

Finite Automata

A deterministic finite automaton (DFA) obtains as input a string x and either accepts or rejects the word.

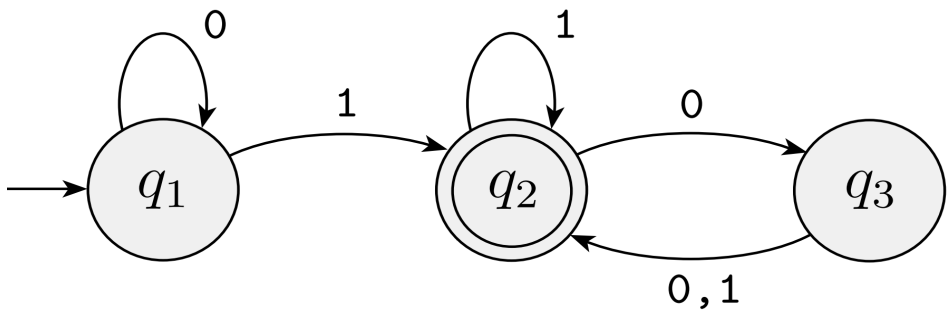
Finite Automata

A deterministic finite automaton (DFA) obtains as input a string x and either accepts or rejects the word.



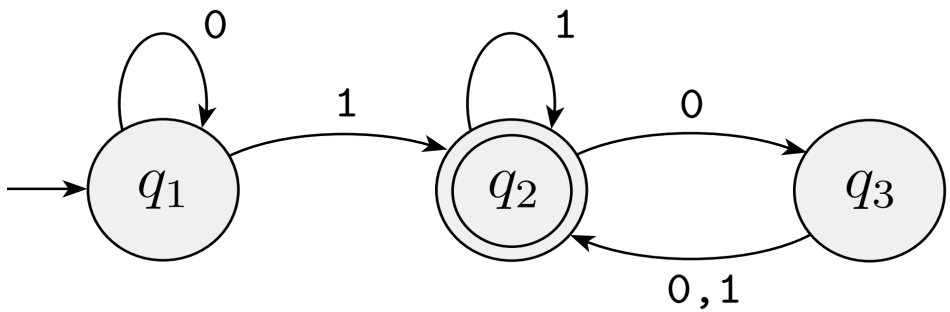
Finite Automata

Example



Finite Automata

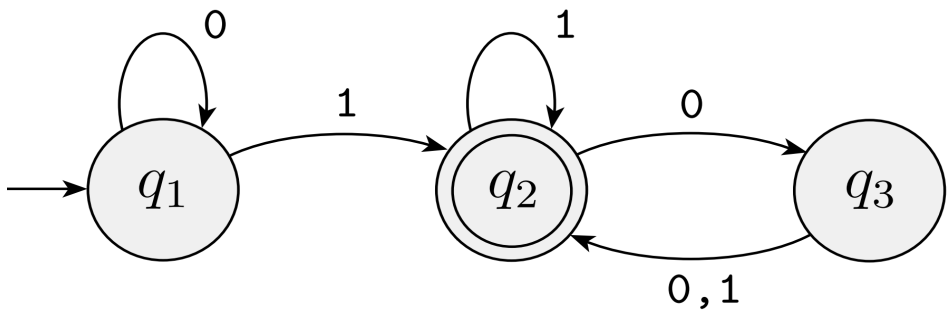
Example



On input 0100:

Finite Automata

Example

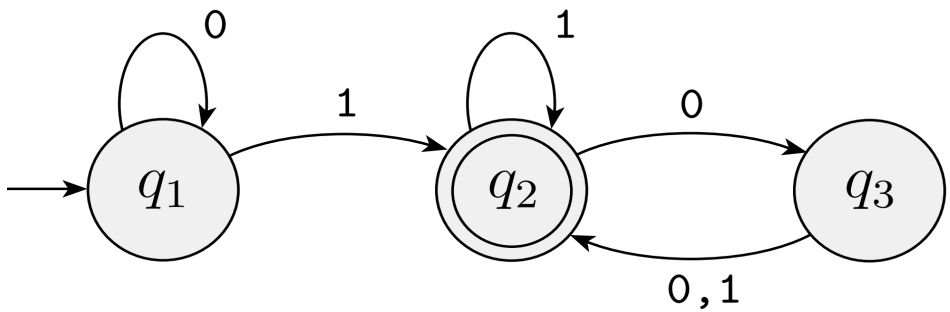


On input 0100:

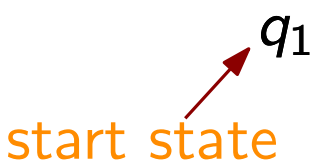
q_1

Finite Automata

Example

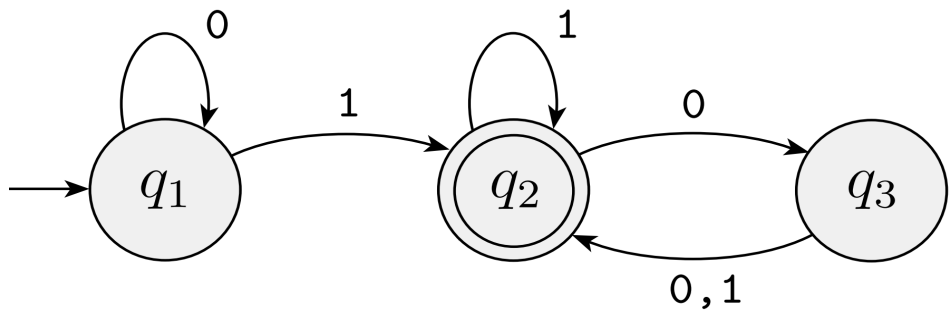


On input 0100:



Finite Automata

Example



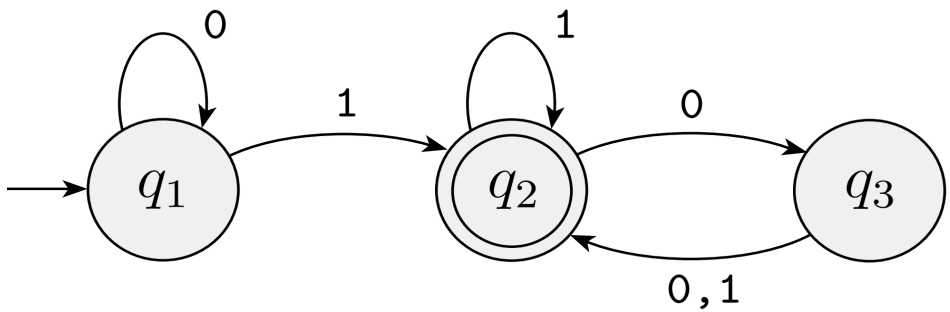
On input 0100:

$$q_1 \xrightarrow{0} q_1$$

start state

Finite Automata

Example



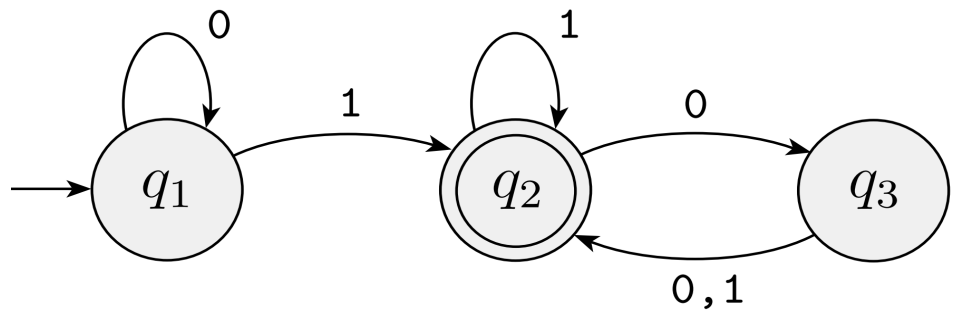
On input 0100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2$$

start state

Finite Automata

Example



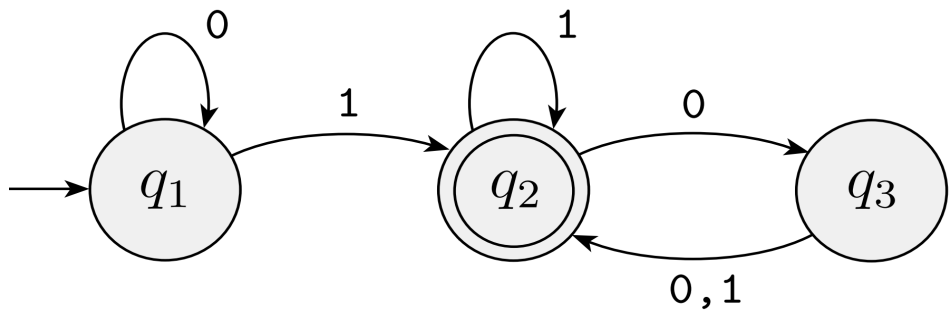
On input 0100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3$$

start state

Finite Automata

Example



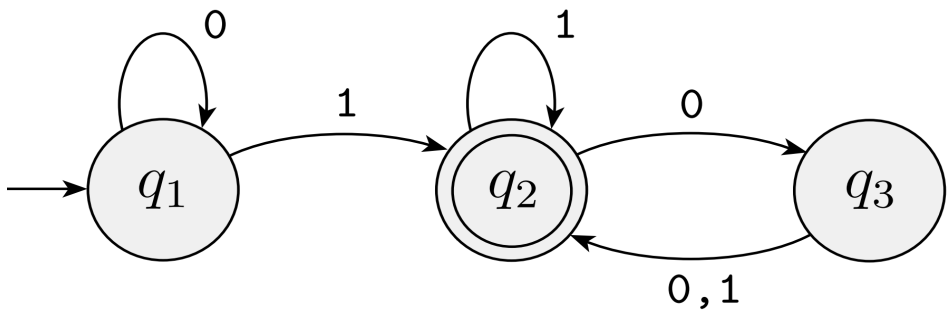
On input 0100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_2$$

start state

Finite Automata

Example



On input 0100:

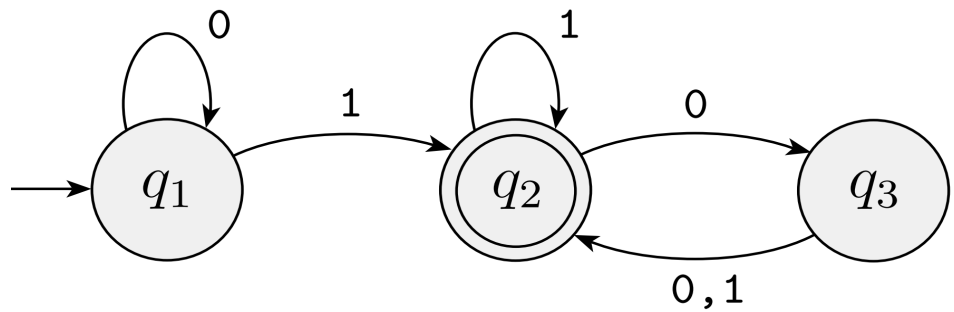
$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_2$$

start state

accepting state

Finite Automata

Example



On input 0100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_2$$

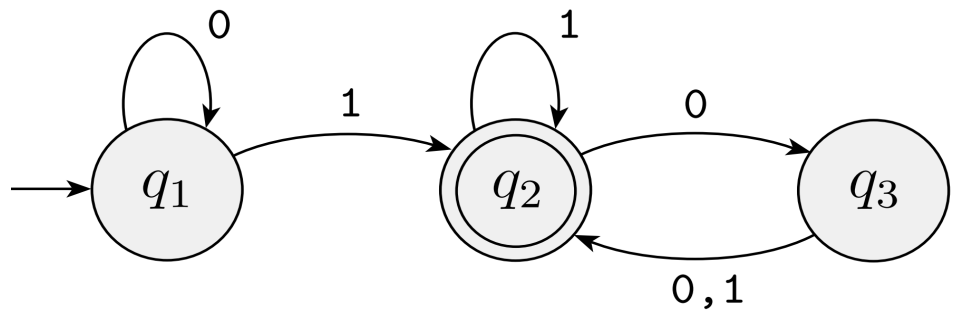
start state

accepting state

0100 is accepted.

Finite Automata

Example



On input 0100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_2$$

start state

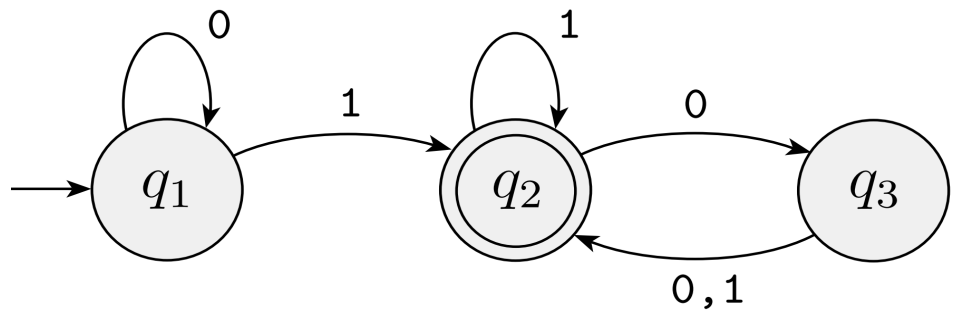
accepting state

0100 is accepted.

On input 0110:

Finite Automata

Example



On input 0100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_2$$

start state

accepting state

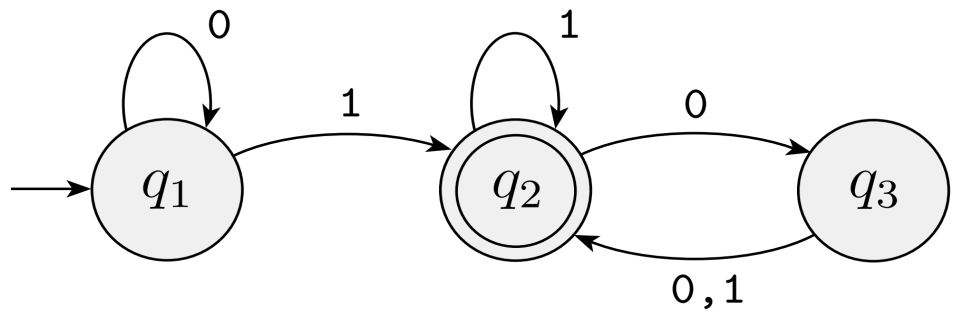
0100 is accepted.

On input 0110:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_2 \xrightarrow{0} q_3$$

Finite Automata

Example



On input 0100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_2$$

start state

accepting state

0100 is accepted.

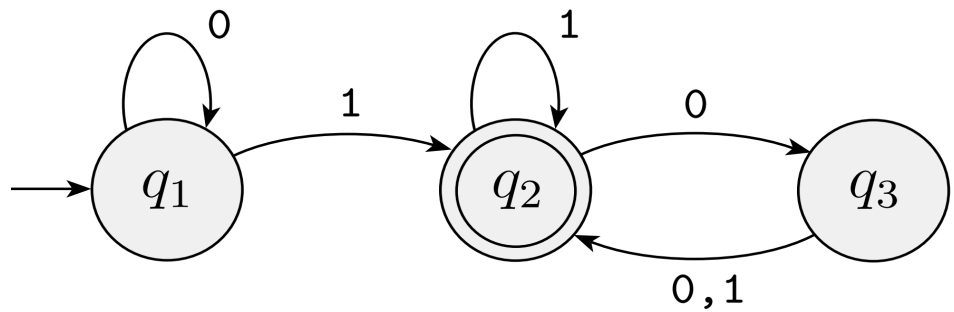
On input 0110:

non-accepting state

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_2 \xrightarrow{0} q_3$$

Finite Automata

Example



On input 0100:

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_3 \xrightarrow{0} q_2$$

start state

accepting state

0100 is accepted.

On input 0110:

non-accepting state

$$q_1 \xrightarrow{0} q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_2 \xrightarrow{0} q_3$$

0110 is rejected.

Finite Automata

Definition

A *deterministic finite automaton* M is a 5-tuple

$$M = (Z, \Sigma, \delta, z_0, E)$$

where

Finite Automata

Definition

A *deterministic finite automaton* M is a 5-tuple

$$M = (Z, \Sigma, \delta, z_0, E)$$

where

- Z: Finite set of *states*

Finite Automata

Definition

A *deterministic finite automaton* M is a 5-tuple

$$M = (Z, \Sigma, \delta, z_0, E)$$

where

- Z: Finite set of *states*
- Σ: Input alphabet with $Z \cap \Sigma = \emptyset$

Finite Automata

Definition

A *deterministic finite automaton* M is a 5-tuple

$$M = (Z, \Sigma, \delta, z_0, E)$$

where

- Z: Finite set of *states*
- Σ: Input alphabet with $Z \cap \Sigma = \emptyset$
- $z_0 \in Z$: Initial state

Finite Automata

Definition

A *deterministic finite automaton* M is a 5-tuple

$$M = (Z, \Sigma, \delta, z_0, E)$$

where

- Z: Finite set of *states*
- Σ: Input alphabet with $Z \cap \Sigma = \emptyset$
- $z_0 \in Z$: Initial state
- $E \subset Z$: Set of accepting / final states

Finite Automata

Definition

A *deterministic finite automaton* M is a 5-tuple

$$M = (Z, \Sigma, \delta, z_0, E)$$

where

- Z: Finite set of *states*
- Σ: Input alphabet with $Z \cap \Sigma = \emptyset$
- $z_0 \in Z$: Initial state
- $E \subset Z$: Set of accepting / final states
- δ: $Z \times \Sigma \to Z$: State transition function

Graphical Representation

State diagram

directed graph with nodes representing the states and labeled edges representing the state transitions

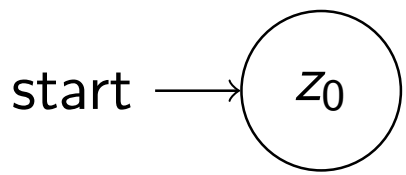
Graphical Representation

State diagram

directed graph with nodes representing the states and labeled edges representing the state transitions

Initial state

labeled with an arrow



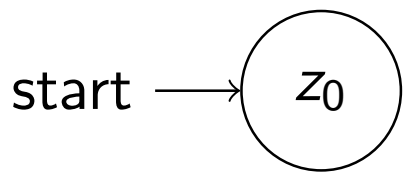
Graphical Representation

State diagram

directed graph with nodes representing the states and labeled edges representing the state transitions

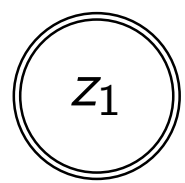
Initial state

labeled with an arrow



Final state

double circle



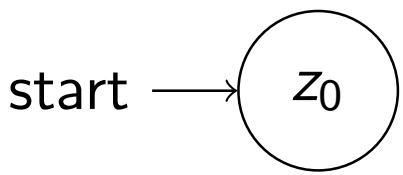
Graphical Representation

State diagram

directed graph with nodes representing the states and labeled edges representing the state transitions

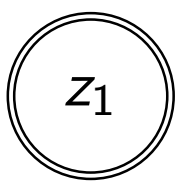
Initial state

labeled with an arrow



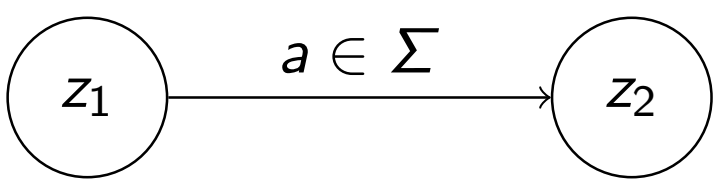
Final state

double circle

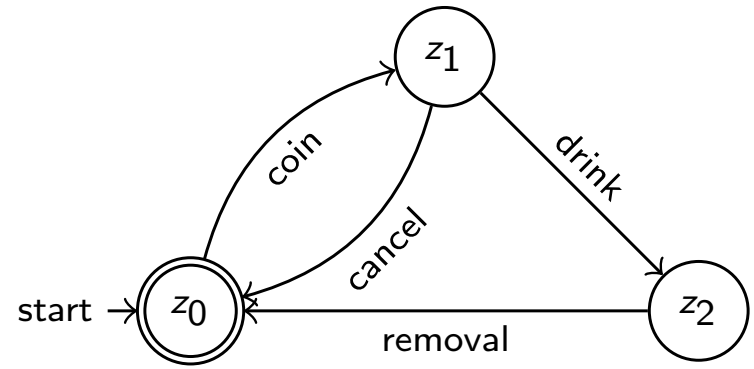


State transition

if $\delta(z_1, a) = z_2$

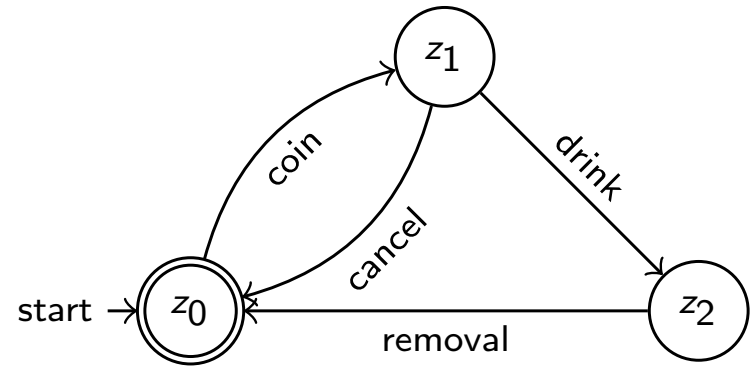


Example: Soda Machine



Example: Soda Machine

States: $Z = \{z_0, z_1, z_2\}$

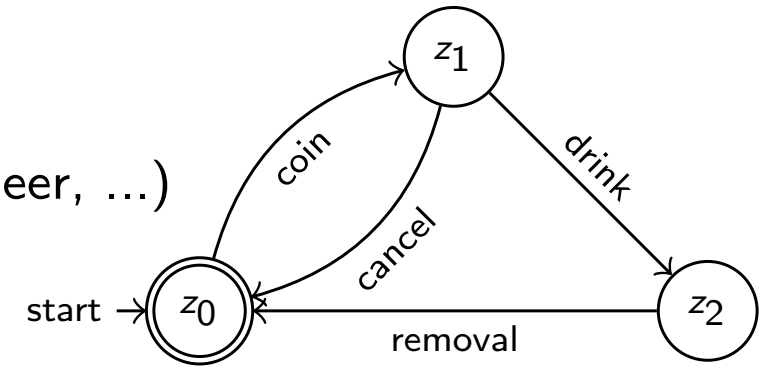


Example: Soda Machine

States: $Z = \{z_0, z_1, z_2\}$

z_0: machine waits for coins
z_1: machine waits for drink choice (Cola, Fanta, Beer, ...)
z_2: machine waits for drink removal

Initial state: z_0
Final state: z_0

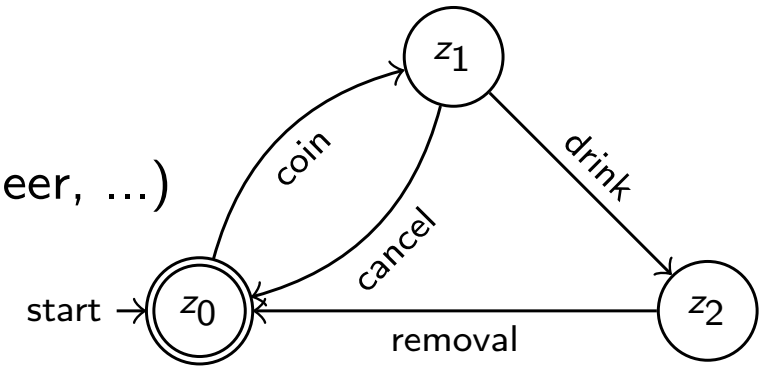


Example: Soda Machine

States: $Z = \{z_0, z_1, z_2\}$

z_0: machine waits for coins
z_1: machine waits for drink choice (Cola, Fanta, Beer, ...)
z_2: machine waits for drink removal

Initial state: z_0
Final state: z_0



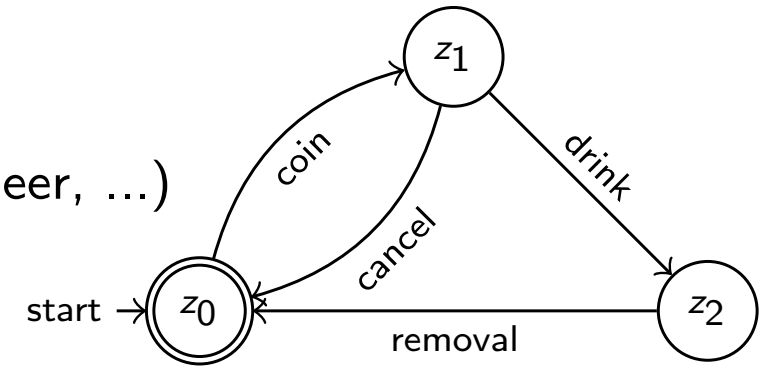
Input alphabet: $\Sigma = \{\text{coin}, \text{drink}, \text{cancel}, \text{removal}\}$

Example: Soda Machine

States: $Z = \{z_0, z_1, z_2\}$

z_0: machine waits for coins
z_1: machine waits for drink choice (Cola, Fanta, Beer, ...)
z_2: machine waits for drink removal

Initial state: z_0
Final state: z_0

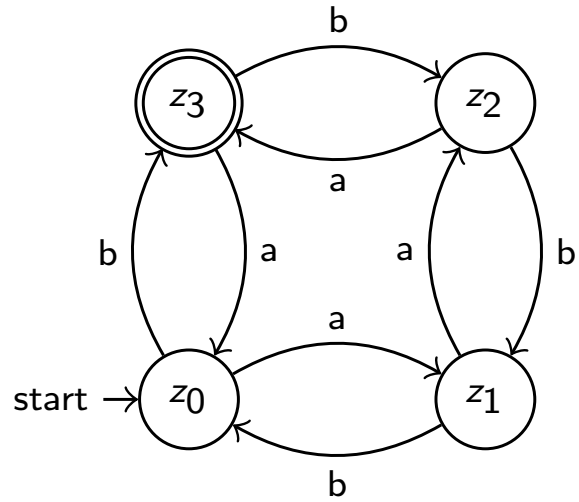


Input alphabet: $\Sigma = \{\text{coin}, \text{drink}, \text{cancel}, \text{removal}\}$

State transitions

$$\delta(z_0, \text{coin}) = z_1$$
$$\delta(z_1, \text{drink}) = z_2$$
$$\delta(z_2, \text{removal}) = z_0$$
$$\delta(z_1, \text{cancel}) = z_0$$

Further Example

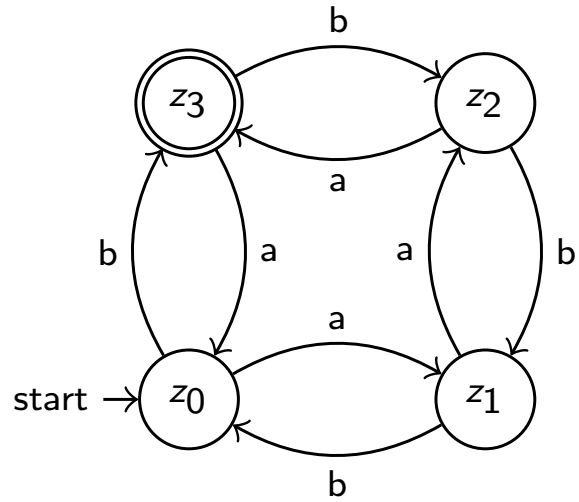


Further Example

States: $Z = \{z_0, z_1, z_2, z_3\}$

Initial state: z_0

Final state: z_3



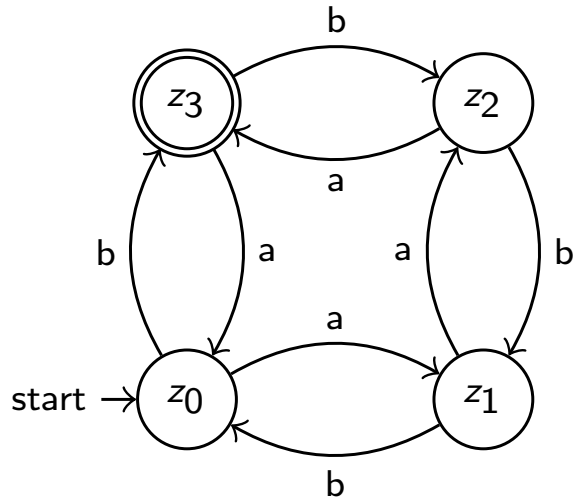
Further Example

States: $Z = \{z_0, z_1, z_2, z_3\}$

Initial state: z_0

Final state: z_3

Input alphabet: $\Sigma = \{a, b\}$



Further Example

States: $Z = \{z_0, z_1, z_2, z_3\}$

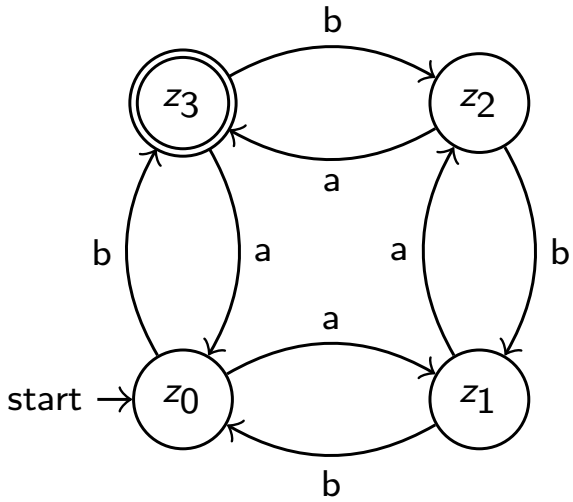
Initial state: z_0

Final state: z_3

Input alphabet: $\Sigma = \{a, b\}$

State transitions

$\delta(z_0, a) = z_1,\ \delta(z_0, b) = z_3,\ \delta(z_1, a) = z_2$
$\delta(z_1, b) = z_0,\ \delta(z_2, a) = z_3,\ \delta(z_2, b) = z_1$
$\delta(z_3, a) = z_0,\ \delta(z_3, b) = z_2$

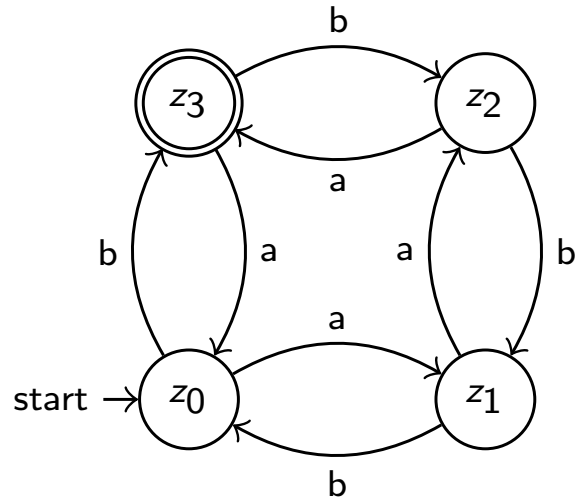


Further Example

States: $Z = \{z_0, z_1, z_2, z_3\}$

Initial state: z_0

Final state: z_3



Input alphabet: $\Sigma = \{a, b\}$

State transitions

$\delta(z_0, a) = z_1,\ \delta(z_0, b) = z_3,\ \delta(z_1, a) = z_2$
$\delta(z_1, b) = z_0,\ \delta(z_2, a) = z_3,\ \delta(z_2, b) = z_1$
$\delta(z_3, a) = z_0,\ \delta(z_3, b) = z_2$

State transitions - Table representation

δ	a	b
z_0	z_1	z_3
z_1	z_2	z_0
z_2	z_3	z_1
z_3	z_0	z_2

Language Accepted by a DFA

Definition

Let $M = (Z, \Sigma, \delta, z_0, E)$ be a deterministic finite automaton. Then we define the function $\hat{\delta} : Z \times \Sigma^* \to Z$ inductively:

$$\hat{\delta}(z, \varepsilon) = z$$

$$\hat{\delta}(z, ax) = \hat{\delta}(\delta(z, a), x)$$

for $a \in \Sigma$, $x \in \Sigma^*$, $z \in Z$.

Language Accepted by a DFA

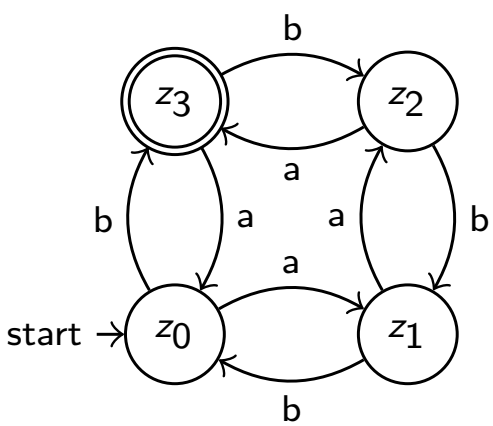
Definition

Let $M = (Z, \Sigma, \delta, z_0, E)$ be a deterministic finite automaton. Then we define the function $\hat{\delta} : Z \times \Sigma^* \to Z$ inductively:

$$\hat{\delta}(z, \varepsilon) = z$$
$$\hat{\delta}(z, ax) = \hat{\delta}(\delta(z, a), x)$$

for $a \in \Sigma$, $x \in \Sigma^*$, $z \in Z$.

Example



Language Accepted by a DFA

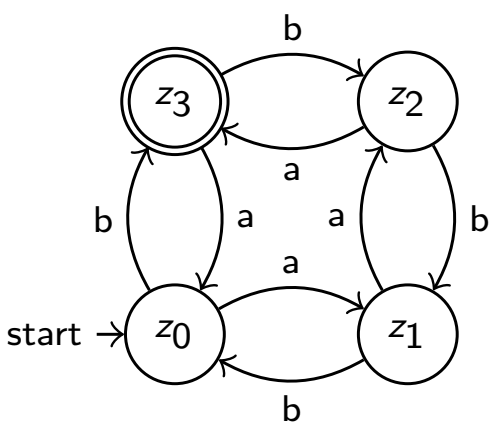
Definition

Let $M = (Z, \Sigma, \delta, z_0, E)$ be a deterministic finite automaton. Then we define the function $\hat{\delta} : Z \times \Sigma^* \to Z$ inductively:

$$\hat{\delta}(z, \varepsilon) = z$$
$$\hat{\delta}(z, ax) = \hat{\delta}(\delta(z, a), x)$$

for $a \in \Sigma$, $x \in \Sigma^*$, $z \in Z$.

Example



$$\hat{\delta}(z_0, aab) = z_1$$
$$\hat{\delta}(z_2, bbbaaa) = z_2$$

Language Accepted by a DFA

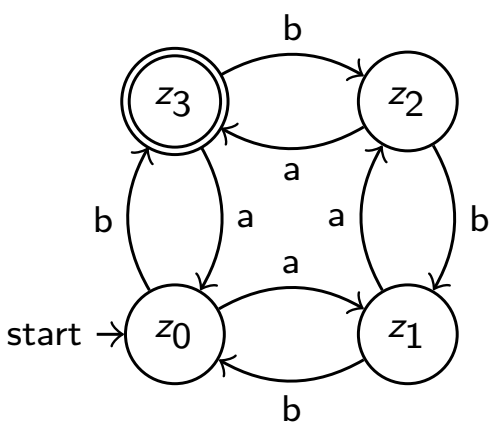
Definition

Let $M = (Z, \Sigma, \delta, z_0, E)$ be a deterministic finite automaton. Then we define the function $\hat{\delta} : Z \times \Sigma^* \to Z$ inductively:

$$\hat{\delta}(z, \varepsilon) = z$$
$$\hat{\delta}(z, ax) = \hat{\delta}(\delta(z, a), x)$$

for $a \in \Sigma$, $x \in \Sigma^*$, $z \in Z$.

Example



$$\hat{\delta}(z_0, aab) = z_1$$
$$\hat{\delta}(z_2, bbbaaa) = z_2$$
$$\hat{\delta}(z_i, b^n a^n) = z_i$$

Language Accepted by a DFA

Language of a DFA

The language accepted or recognized by M is then given by

$$L(M) := \{x \in \Sigma^* \mid \hat{\delta}(z_0, x) \in E\}.$$

Language Accepted by a DFA

Language of a DFA

The language accepted or recognized by M is then given by

$$L(M) := \{x \in \Sigma^* \mid \hat{\delta}(z_0, x) \in E\}.$$

Remarks

- $\hat{\delta}$ extends δ from characters to words

Language Accepted by a DFA

Language of a DFA

The language accepted or recognized by M is then given by

$$L(M) := \{x \in \Sigma^* \mid \hat{\delta}(z_0, x) \in E\}.$$

Remarks

- $\hat{\delta}$ extends δ from characters to words
- For characters we have $\hat{\delta}(z, a) = \delta(z, a)$
  (in this case we do not differentiate between length 1 words and characters)

Language Accepted by a DFA

Language of a DFA

The language accepted or recognized by M is then given by

$$L(M) := \{x \in \Sigma^* \mid \hat{\delta}(z_0, x) \in E\}.$$

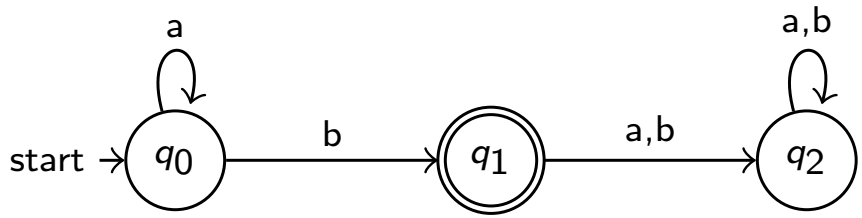
Remarks

- $\hat{\delta}$ extends δ from characters to words
- For characters we have $\hat{\delta}(z, a) = \delta(z, a)$
  (in this case we do not differentiate between length 1 words and characters)
- Further we have: $\hat{\delta}(z, a_1 a_2 \dots a_n) = \delta(\dots \delta(\delta(z, a_1), a_2) \dots, a_n)$

Language Accepted by a DFA

Example

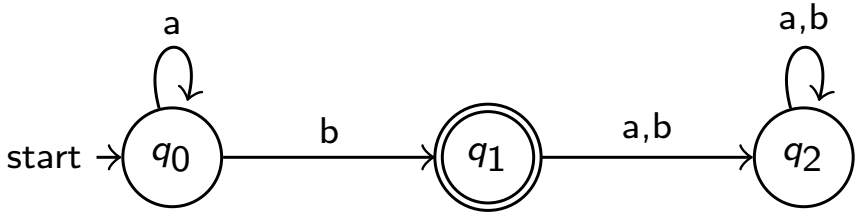
Automaton M



Language Accepted by a DFA

Example

Automaton M



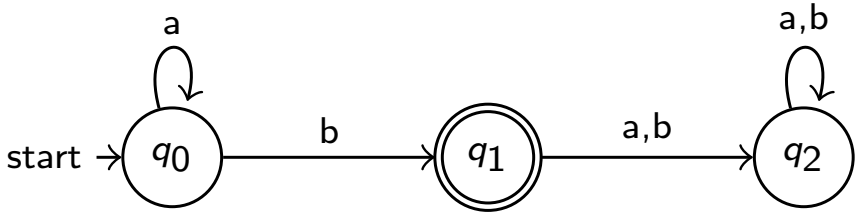
Claim

$$L(M) = \{a\}^*\{b\}$$

Language Accepted by a DFA

Example

Automaton M



Claim

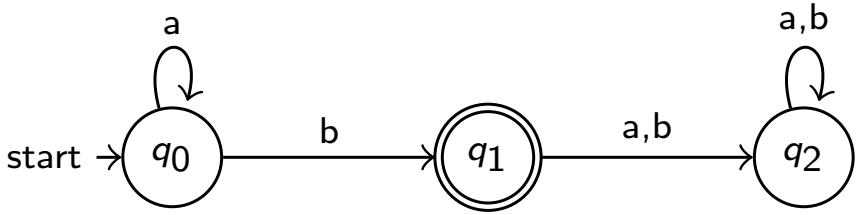
$$L(M) = \{a\}^*\{b\}$$

How to prove this?

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

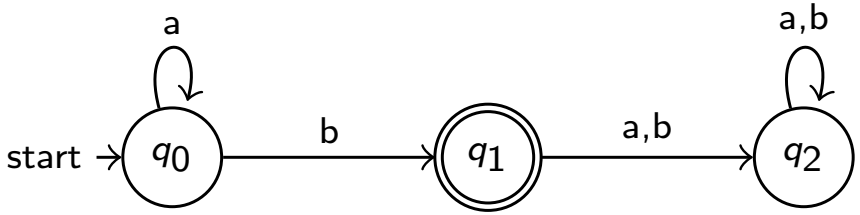
How to prove this?

Step 1: $L(M) \subseteq \{a\}^*\{b\}$

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

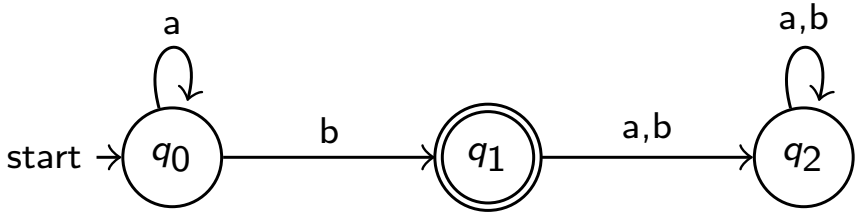
Step 1: $L(M) \subseteq \{a\}^*\{b\}$

Let $x \in L(M)$, i.e. x is accepted by M.

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

Step 1: $L(M) \subseteq \{a\}^*\{b\}$

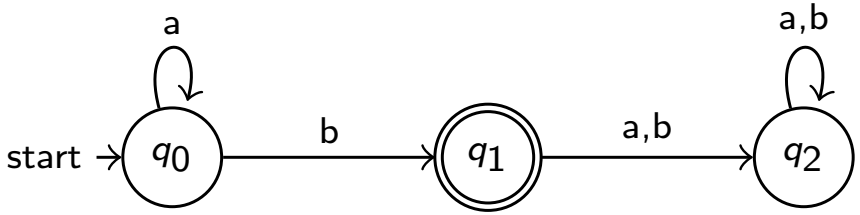
Let $x \in L(M)$, i.e. x is accepted by M.

As the accepting state, q_1 is only reachable through a b-edge, x must end with b.

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

Step 1: $L(M) \subseteq \{a\}^*\{b\}$

Let $x \in L(M)$, i.e. x is accepted by M.

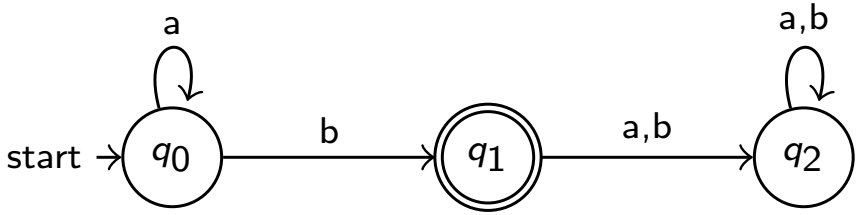
As the accepting state, q_1 is only reachable through a b-edge, x must end with b.

Before that, M was in state q_0. In that state, only a's can be read.

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

Step 1: $L(M) \subseteq \{a\}^*\{b\}$

Let $x \in L(M)$, i.e. x is accepted by M.

As the accepting state, q_1 is only reachable through a b-edge, x must end with b.

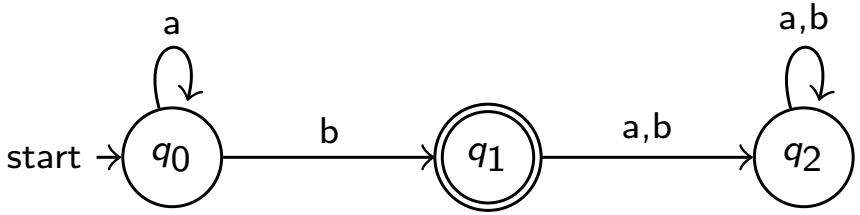
Before that, M was in state q_0. In that state, only a's can be read.

That means x must be of the form a^*b, i.e. $x \in \{a\}^*\{b\}$.

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

Step 1: $L(M) \subseteq \{a\}^*\{b\}$

Let $x \in L(M)$, i.e. x is accepted by M.

As the accepting state, q_1 is only reachable through a b-edge, x must end with b.

Before that, M was in state q_0. In that state, only a's can be read.

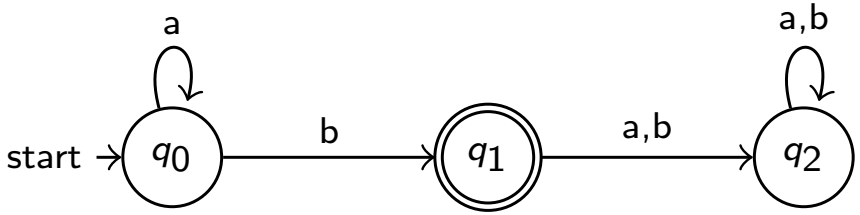
That means x must be of the form a^*b, i.e. $x \in \{a\}^*\{b\}$.

So we have proved that $L(M) \subseteq \{a\}^*\{b\}$

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

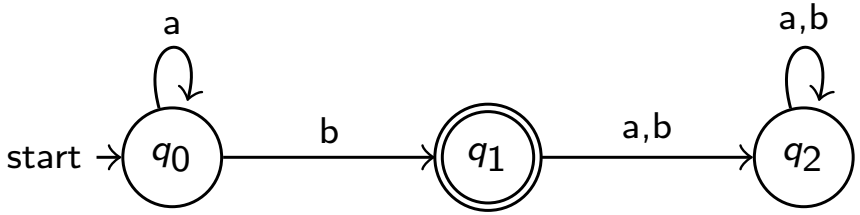
How to prove this?

Step 2: $\{a\}^*\{b\} \subseteq L(M)$

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

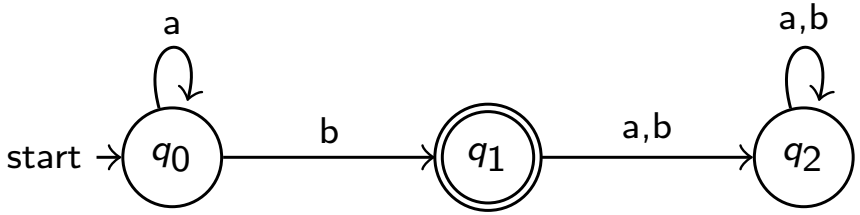
Step 2: $\{a\}^*\{b\} \subseteq L(M)$

Let $x \in \{a\}^*\{b\}$, i.e. x is of the form a^*b.

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

Step 2: $\{a\}^*\{b\} \subseteq L(M)$

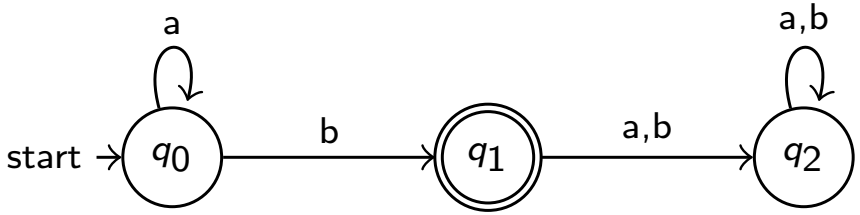
Let $x \in \{a\}^*\{b\}$, i.e. x is of the form a^*b.

We have $\hat{\delta}(q_0, a^*) = q_0$ and $\delta(q_0, b) = q_1$, so $\hat{\delta}(q_0, a^*b) = q_1$.

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

Step 2: $\{a\}^*\{b\} \subseteq L(M)$

Let $x \in \{a\}^*\{b\}$, i.e. x is of the form a^*b.

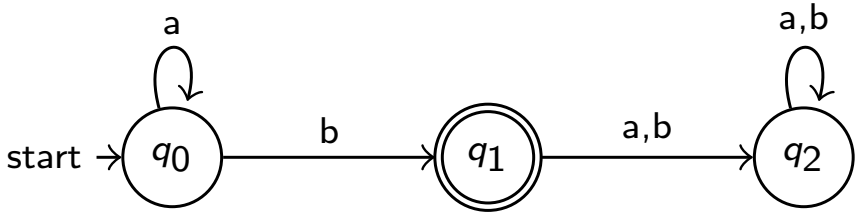
We have $\hat{\delta}(q_0, a^*) = q_0$ and $\delta(q_0, b) = q_1$, so $\hat{\delta}(q_0, a^*b) = q_1$.

That means x is accepted by M, i.e. $x \in L(M)$.

Language Accepted by a DFA

Example

Automaton M



Claim

$$L(M) = \{a\}^*\{b\}$$

How to prove this?

Step 2: $\{a\}^*\{b\} \subseteq L(M)$

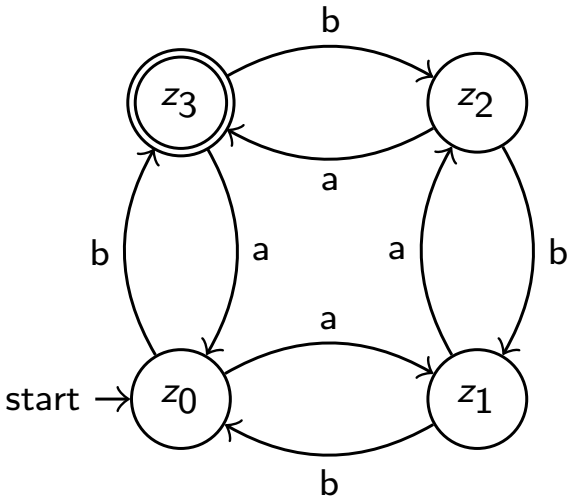
Let $x \in \{a\}^*\{b\}$, i.e. x is of the form a^*b.

We have $\hat{\delta}(q_0, a^*) = q_0$ and $\delta(q_0, b) = q_1$, so $\hat{\delta}(q_0, a^*b) = q_1$.

That means x is accepted by M, i.e. $x \in L(M)$.

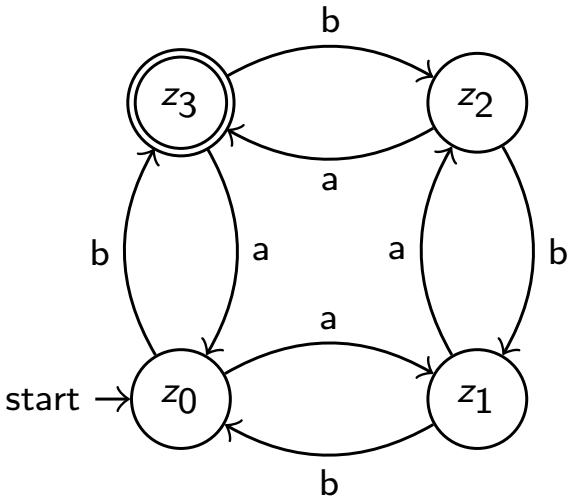
So we have proved that $\{a\}^*\{b\} \subseteq L(M)$.

Further Example



Accepting language?

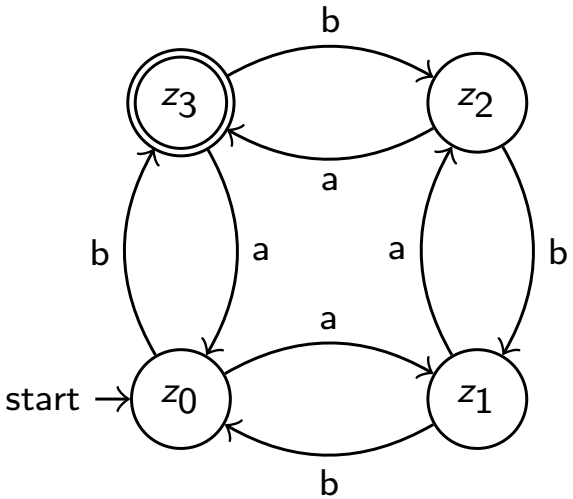
Further Example



Accepting language?

$$L = \{x \in \{a, b\}^* \mid (\#a \text{ in } x) - (\#b \text{ in } x) \equiv 3 \pmod 4\}$$

Further Example



Accepting language?

$$L = \{x \in \{a, b\}^* \mid (\#a \text{ in } x) - (\#b \text{ in } x) \equiv 3 \pmod 4\}$$

Try to prove it formally.

Constructing DFA for a Language

Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$

Constructing DFA for a Language

Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$

We need to keep track of the parity of the number of a's and b's by reading characters of the input x.

Constructing DFA for a Language

Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$

We need to keep track of the parity of the number of a's and b's by reading characters of the input x.

To achieve this, consider following 4 states for M:

- q_{00}: M is in this state when it has read an even # of a's and even # b's of x.

Constructing DFA for a Language

Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$

We need to keep track of the parity of the number of a's and b's by reading characters of the input x.

To achieve this, consider following 4 states for M:

- q_{00}: M is in this state when it has read an even # of a's and even # b's of x.
- q_{01}: even # of a's and odd # b's
- q_{10}: odd # of a's and even # b's
- q_{11}: odd # of a's and odd # b's

Constructing DFA for a Language

Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$

We need to keep track of the parity of the number of a's and b's by reading characters of the input x.

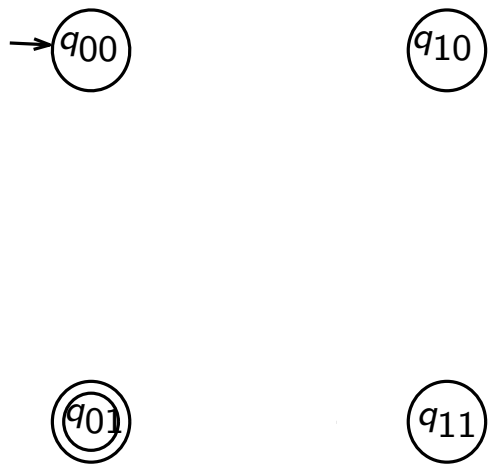
To achieve this, consider following 4 states for M:

- q_{00}: M is in this state when it has read an even # of a's and even # b's of x.
- q_{01}: even # of a's and odd # b's
- q_{10}: odd # of a's and even # b's
- q_{11}: odd # of a's and odd # b's

Constructing DFA for a Language

Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$



We need to keep track of the parity of the number of a's and b's by reading characters of the input x.

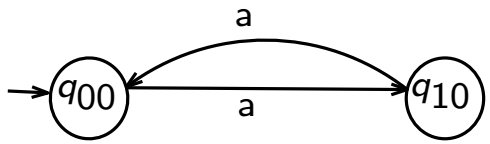
To achieve this, consider following 4 states for M:

- q_{00}: M is in this state when it has read an even # of a's and even # b's of x.
- q_{01}: even # of a's and odd # b's
- q_{10}: odd # of a's and even # b's
- q_{11}: odd # of a's and odd # b's

Constructing DFA for a Language

Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$



We need to keep track of the parity of the number of a's and b's by reading characters of the input x.

To achieve this, consider following 4 states for M:

- q_{00}: M is in this state when it has read an even # of a's and even # b's of x.
- q_{01}: even # of a's and odd # b's
- q_{10}: odd # of a's and even # b's
- q_{11}: odd # of a's and odd # b's

Constructing DFA for a Language

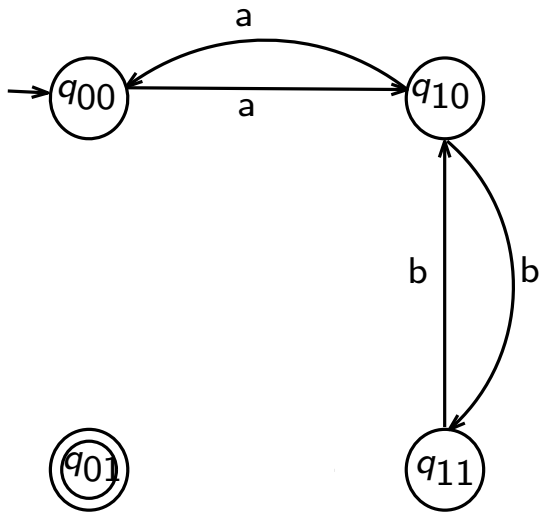
Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$

We need to keep track of the parity of the number of a's and b's by reading characters of the input x.

To achieve this, consider following 4 states for M:

- q_{00}: M is in this state when it has read an even # of a's and even # b's of x.
- q_{01}: even # of a's and odd # b's
- q_{10}: odd # of a's and even # b's
- q_{11}: odd # of a's and odd # b's



Constructing DFA for a Language

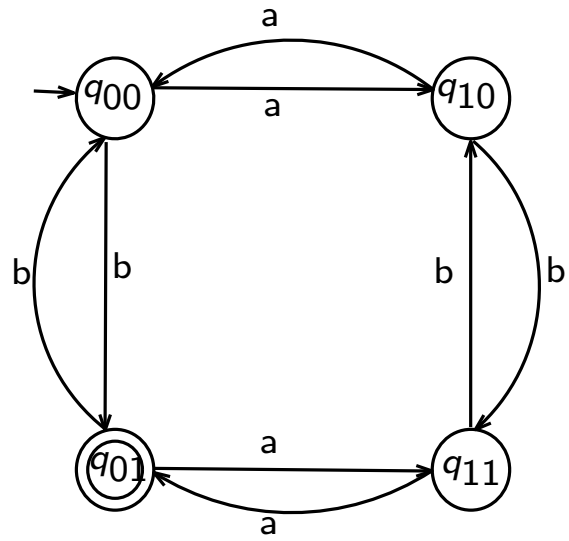
Example

Construct an automaton M recognizing
$A = \{x \in \{\mathsf{a}, \mathsf{b}\}^* : |x|_{\mathsf{a}}$ is even and $|x|_{\mathsf{b}}$ is odd$\}$

We need to keep track of the parity of the number of a's and b's by reading characters of the input x.

To achieve this, consider following 4 states for M:

- q_{00}: M is in this state when it has read an even # of a's and even # b's of x.
- q_{01}: even # of a's and odd # b's
- q_{10}: odd # of a's and even # b's
- q_{11}: odd # of a's and odd # b's



DFA: Regular Languages

Theorem

Every language that can be accepted by a DFA is generated by a regular (type 3) grammar.

DFA: Regular Languages

Theorem

Every language that can be accepted by a DFA is generated by a regular (type 3) grammar.

Grammar	Languages	Production rules (constraints)
Type 3	Regular	$A \rightarrow \mathrm{a}$, $A \rightarrow \mathrm{a}B$

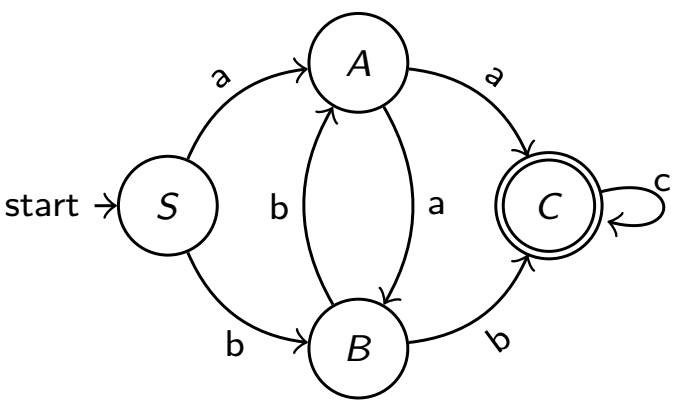
DFA: Regular Languages

Theorem

Every language that can be accepted by a DFA is generated by a regular (type 3) grammar.

Grammar	Languages	Production rules (constraints)
Type 3	Regular	$A \longrightarrow$ a, $A \longrightarrow$ aB

Example



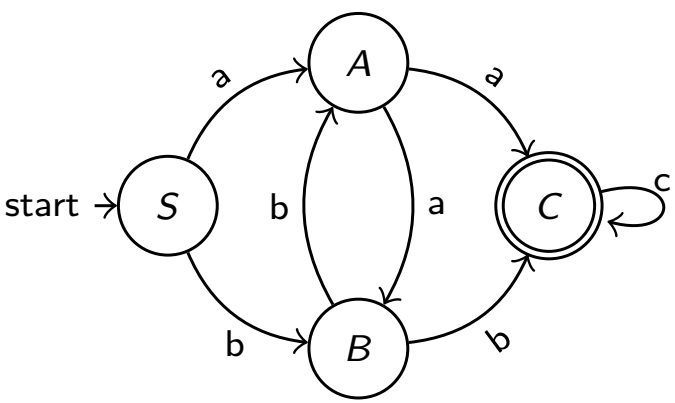
DFA: Regular Languages

Theorem

Every language that can be accepted by a DFA is generated by a regular (type 3) grammar.

Grammar	Languages	Production rules (constraints)
Type 3	Regular	$A \to \mathrm{a}$, $A \to \mathrm{a}B$

Example



For every transition $q \xrightarrow{\mathrm{a}} q'$ put the rule $q \to \mathrm{a}q'$:

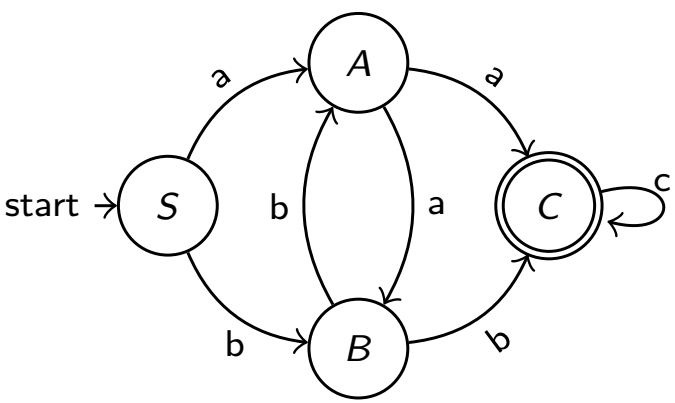
DFA: Regular Languages

Theorem

Every language that can be accepted by a DFA is generated by a regular (type 3) grammar.

Grammar	Languages	Production rules (constraints)
Type 3	Regular	$A \to \mathrm{a}$, $A \to \mathrm{a}B$

Example



For every transition $q \xrightarrow{\mathrm{a}} q'$ put the rule $q \to \mathrm{a}q'$:

$S \to \mathrm{a}A \mid \mathrm{b}B$

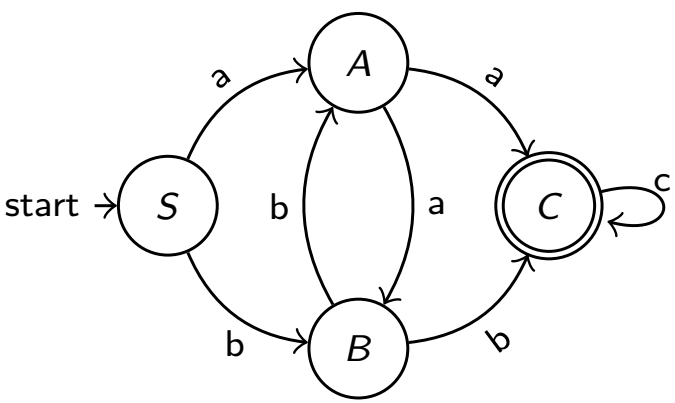
DFA: Regular Languages

Theorem

Every language that can be accepted by a DFA is generated by a regular (type 3) grammar.

Grammar	Languages	Production rules (constraints)
Type 3	Regular	$A \to \mathrm{a}$, $A \to \mathrm{a}B$

Example



For every transition $q \xrightarrow{\mathrm{a}} q'$ put the rule $q \to \mathrm{a}q'$:

$S \to \mathrm{a}A \mid \mathrm{b}B$

$A \to \mathrm{a}B \mid \mathrm{a}C$

$B \to \mathrm{b}A \mid \mathrm{b}C$

$C \to \mathrm{c}C$

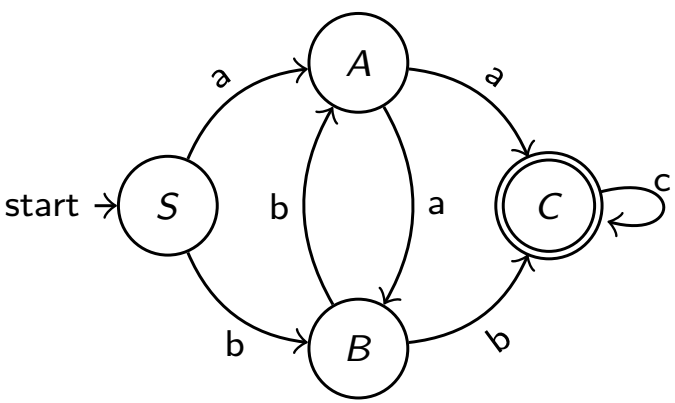
DFA: Regular Languages

Theorem

Every language that can be accepted by a DFA is generated by a regular (type 3) grammar.

Grammar	Languages	Production rules (constraints)
Type 3	Regular	$A \to \mathrm{a}$, $A \to \mathrm{a}B$

Example



For every transition $q \xrightarrow{\mathrm{a}} q'$ put the rule $q \to \mathrm{a}q'$:

For every $q \xrightarrow{\mathrm{a}} q_{\text{accept}}$ put the rule $q \to \mathrm{a}$:

$S \to \mathrm{a}A \mid \mathrm{b}B$

$A \to \mathrm{a}B \mid \mathrm{a}C$

$B \to \mathrm{b}A \mid \mathrm{b}C$

$C \to \mathrm{c}C$

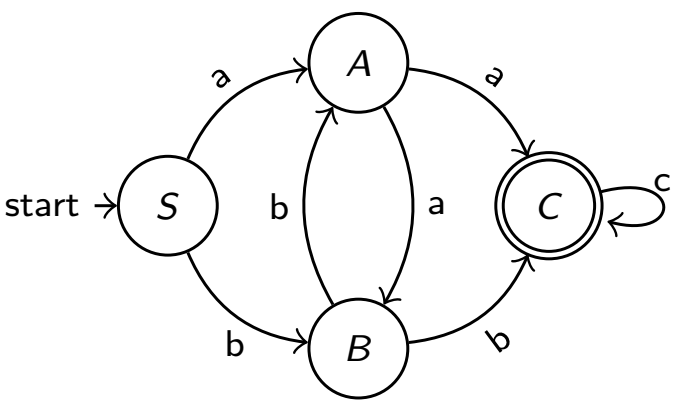
DFA: Regular Languages

Theorem

Every language that can be accepted by a DFA is generated by a regular (type 3) grammar.

Grammar	Languages	Production rules (constraints)
Type 3	Regular	$A \to \mathrm{a}$, $A \to \mathrm{a}B$

Example



For every transition $q \xrightarrow{\mathrm{a}} q'$ put the rule $q \to \mathrm{a}q'$:

For every $q \xrightarrow{\mathrm{a}} q_{\text{accept}}$ put the rule $q \to \mathrm{a}$:

$S \to \mathrm{a}A \mid \mathrm{b}B$

$A \to \mathrm{a}B \mid \mathrm{a}C$

$B \to \mathrm{b}A \mid \mathrm{b}C$

$C \to \mathrm{c}C$

$A \to \mathrm{a}$

$B \to \mathrm{b}$

$C \to \mathrm{c}$

DFA: Regular Languages

Proof of Theorem

Let $M = (Z, \Sigma, \delta, z_0, E)$ be an automaton.

DFA: Regular Languages

Proof of Theorem

Let $M = (Z, \Sigma, \delta, z_0, E)$ be an automaton.

We construct a regular grammar $G = (V, \Sigma_G, P, S)$ with

$$V = Z, \;\; \Sigma_G = \Sigma, \;\; S = z_0,$$

DFA: Regular Languages

Proof of Theorem

Let $M = (Z, \Sigma, \delta, z_0, E)$ be an automaton.

We construct a regular grammar $G = (V, \Sigma_G, P, S)$ with

$$V = Z, \ \Sigma_G = \Sigma, \ S = z_0,$$

and rules P:

- if $z_0 \in E$ then $(z_0 \to \varepsilon) \in P$

DFA: Regular Languages

Proof of Theorem

Let $M = (Z, \Sigma, \delta, z_0, E)$ be an automaton.

We construct a regular grammar $G = (V, \Sigma_G, P, S)$ with

$$V = Z, \ \Sigma_G = \Sigma, \ S = z_0,$$

and rules P:

- if $z_0 \in E$ then $(z_0 \to \varepsilon) \in P$
- for each transition $\delta(z_1, a) = z_2$, put the following rule:

$$(z_1 \to a z_2) \in P$$

DFA: Regular Languages

Proof of Theorem

Let $M = (Z, \Sigma, \delta, z_0, E)$ be an automaton.

We construct a regular grammar $G = (V, \Sigma_G, P, S)$ with

$$V = Z, \ \Sigma_G = \Sigma, \ S = z_0,$$

and rules P:

- if $z_0 \in E$ then $(z_0 \to \varepsilon) \in P$
- for each transition $\delta(z_1, a) = z_2$, put the following rule:

$$(z_1 \to a z_2) \in P$$

and if $z_2 \in E$:

$$(z_1 \to a) \in P$$

DFA: Regular Languages

Proof of Theorem (cont'd)

$x = a_1 a_2 \ldots a_n \in L(M)$

DFA: Regular Languages

Proof of Theorem (cont'd)

$x = a_1 a_2 \ldots a_n \in L(M)$

$\Leftrightarrow$ $\exists$ sequence $z_0, z_1, \ldots, z_n$ of states of M with z_0 the initial state and $z_n \in E$ such that

$$\delta(z_0, a_1) = z_1,\ \delta(z_1, a_2) = z_2,\ \ldots,\ \delta(z_{n-1}, a_n) = z_n.$$

DFA: Regular Languages

Proof of Theorem (cont'd)

$x = a_1 a_2 \ldots a_n \in L(M)$

$\Leftrightarrow$ $\exists$ sequence $z_0, z_1, \ldots, z_n$ of states of M with z_0 the initial state and $z_n \in E$ such that

$$\delta(z_0, a_1) = z_1,\ \delta(z_1, a_2) = z_2,\ \ldots,\ \delta(z_{n-1}, a_n) = z_n.$$

$\Leftrightarrow$ $\exists$ sequence $z_0, z_1 \ldots, z_n$ of variables of G with z_0 the initial variable such that

$$z_0 \Rightarrow a_1 z_1 \Rightarrow a_1 a_2 z_2 \Rightarrow \cdots \Rightarrow a_1 a_2 \ldots a_{n-1} z_{n-1} \Rightarrow a_1 a_2 \ldots a_{n-1} a_r$$

DFA: Regular Languages

Proof of Theorem (cont'd)

$x = a_1 a_2 \ldots a_n \in L(M)$

$\Leftrightarrow$ $\exists$ sequence $z_0, z_1, \ldots, z_n$ of states of M with z_0 the initial state and $z_n \in E$ such that

$$\delta(z_0, a_1) = z_1,\ \delta(z_1, a_2) = z_2,\ \ldots,\ \delta(z_{n-1}, a_n) = z_n.$$

$\Leftrightarrow$ $\exists$ sequence $z_0, z_1 \ldots, z_n$ of variables of G with z_0 the initial variable such that

$$z_0 \Rightarrow a_1 z_1 \Rightarrow a_1 a_2 z_2 \Rightarrow \cdots \Rightarrow a_1 a_2 \ldots a_{n-1} z_{n-1} \Rightarrow a_1 a_2 \ldots a_{n-1} a_r$$

$\Leftrightarrow$ $x \in L(G)$

DFA: Total State Transition

Remark

The state transition function of a DFA might be a partial funcation, i.e. δ does not need to be defined for each element in the domain of definition.

DFA: Total State Transition

Remark

The state transition function of a DFA might be a partial funcation, i.e. δ does not need to be defined for each element in the domain of definition.

Mathematically one can associate a total function $f' : X \to Y \cup \{\bot\}$ for a partial function $f : X \to Y$ with:

$$f'(x) = \begin{cases} f(x) & \text{if } x \in \mathrm{Def}(f) \\ \bot & \text{else} \end{cases}$$

DFA: Total State Transition

Construction of a DFA with total state transitions

- Introduce a new state $\perp$ (error state) with $\perp \notin \Sigma \cup Z$ and $\perp \notin E$

DFA: Total State Transition

Construction of a DFA with total state transitions

- Introduce a new state $\perp$ (error state) with $\perp \notin \Sigma \cup Z$ and $\perp \notin E$
- Let all missing transitions point to $\perp$.

DFA: Total State Transition

Construction of a DFA with total state transitions

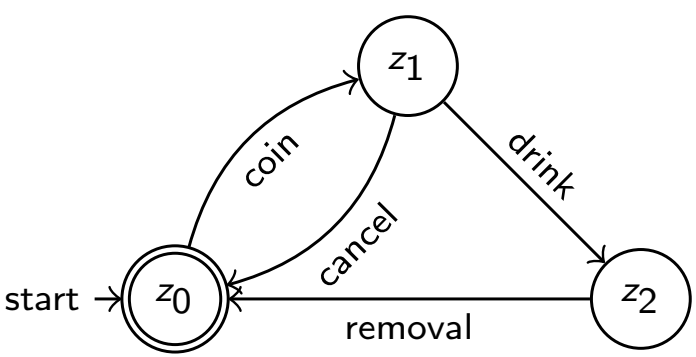
- Introduce a new state $\perp$ (error state) with $\perp \notin \Sigma \cup Z$ and $\perp \notin E$
- Let all missing transitions point to $\perp$.
- Add $\forall x \in \Sigma : \delta(\perp, x) = \perp$ to δ

DFA: Total State Transition

Construction of a DFA with total state transitions

- Introduce a new state $\perp$ (error state) with $\perp \notin \Sigma \cup Z$ and $\perp \notin E$
- Let all missing transitions point to $\perp$.
- Add $\forall x \in \Sigma : \delta(\perp, x) = \perp$ to δ

Example: Soda Machine



DFA: Total State Transition

Construction of a DFA with total state transitions

- Introduce a new state $\perp$ (error state) with $\perp \notin \Sigma \cup Z$ and $\perp \notin E$
- Let all missing transitions point to $\perp$.
- Add $\forall x \in \Sigma : \delta(\perp, x) = \perp$ to δ

Example: Soda Machine