Theory of Computer Science

C2. Regular Languages: Finite Automata

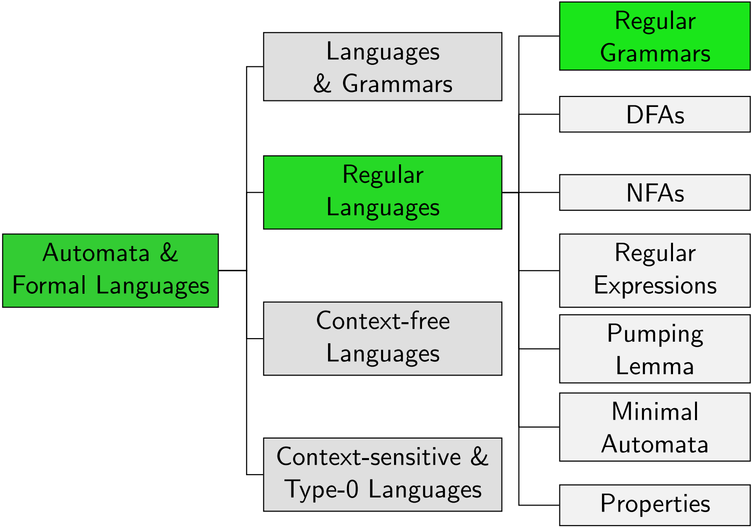
Gabriele Röger

University of Basel

March 23, 2020

Regular Grammars

Overview


Automata &
Formal Languages
Languages
& Grammars
Regular
Languages
Context-free
Languages
Context-sensitive &
Type-0 Languages
Regular
Grammars
DFAs
NFAs
Regular
Expressions
Pumping
Lemma
Minimal
Automata
Properties

Repetition: Regular Grammars

Definition (Regular Grammars)

A regular grammar is a 4-tuple $\langle \Sigma, V, P, S \rangle$ with

1. Σ finite alphabet of terminals
2. V finite set of variables (with $V \cap \Sigma = \emptyset$)
3. $P \subseteq (V \times (\Sigma \cup \Sigma V)) \cup \{\langle S, \varepsilon \rangle\}$ finite set of rules
4. if $S \to \varepsilon \in P$, there is no $X \in V, y \in \Sigma$ with $X \to yS \in P$
5. $S \in V$ start variable.

Repetition: Regular Grammars

Definition (Regular Grammars)

A regular grammar is a 4-tuple $\langle \Sigma, V, P, S\rangle$ with

1. Σ finite alphabet of terminals
2. V finite set of variables (with $V \cap \Sigma = \emptyset$)
3. $P \subseteq (V \times (\Sigma \cup \Sigma V)) \cup \{\langle S, \varepsilon\rangle\}$ finite set of rules
4. if $S \to \varepsilon \in P$, there is no $X \in V, y \in \Sigma$ with $X \to yS \in P$
5. $S \in V$ start variable.

Rule $X \to \varepsilon$ is only allowed if $X = S$ and
S never occurs in the right-hand side of a rule.

Repetition: Regular Grammars

Definition (Regular Grammars)

A regular grammar is a 4-tuple $\langle \Sigma, V, P, S \rangle$ with

1. Σ finite alphabet of terminals
2. V finite set of variables (with $V \cap \Sigma = \emptyset$)
3. $P \subseteq (V \times (\Sigma \cup \Sigma V)) \cup \{\langle S, \varepsilon \rangle\}$ finite set of rules
4. if $S \to \varepsilon \in P$, there is no $X \in V, y \in \Sigma$ with $X \to yS \in P$
5. $S \in V$ start variable.

Rule $X \to \varepsilon$ is only allowed if $X = S$ and
S never occurs in the right-hand side of a rule.
How restrictive is this?

Start Variable in Right-Hand Side of Rules

For every type-0 language L there is a grammar where the start variable does not occur on the right-hand side of any rule.

Theorem

For every grammar $G = \langle \Sigma, V, P, S \rangle$ *there is a grammar* $G' = \langle \Sigma, V', P', S \rangle$ *with rules* $P' \subseteq (V' \cup \Sigma)^+ \times (V' \setminus \{S\} \cup \Sigma)^*$ *such that* $\mathcal{L}(G) = \mathcal{L}(G')$.

Start Variable in Right-Hand Side of Rules: Proof

Proof.

Let $G = \langle \Sigma, V, P, S \rangle$ be a grammar and $S' \notin V$ be a new variable. Construct rule set P' from P as follows:

- for every rule $r \in P$, add a rule r' to P', where r' is the result of replacing all occurences of S in r with S'.
- for every rule $S \rightarrow w \in P$, add a rule $S \rightarrow w'$ to P', where w' is the result of replacing all occurences of S in w with S'.

Then $\mathcal{L}(G) = \mathcal{L}(\langle \Sigma, V \cup \{S'\}, P', S \rangle)$. □

Start Variable in Right-Hand Side of Rules: Proof

Proof.

Let $G = \langle \Sigma, V, P, S \rangle$ be a grammar and $S' \notin V$ be a new variable. Construct rule set P' from P as follows:

- for every rule $r \in P$, add a rule r' to P', where r' is the result of replacing all occurences of S in r with S'.
- for every rule $S \to w \in P$, add a rule $S \to w'$ to P', where w' is the result of replacing all occurences of S in w with S'.

Then $\mathcal{L}(G) = \mathcal{L}(\langle \Sigma, V \cup \{S'\}, P', S \rangle)$. □

Note that the rules in P' are not fundamentally different from the rules in P. In particular:

- If $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$ then $P' \subseteq V' \times (\Sigma \cup \Sigma V' \cup \{\varepsilon\})$.
- If $P \subseteq V \times (V \cup \Sigma)^*$ then $P' \subseteq V' \times (V' \cup \Sigma)^*$.

Start Variable in Right-Hand Side of Rules: Example

Epsilon Rules

Theorem

For every grammar G with rules $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$ there is a regular grammar G' with $\mathcal{L}(G) = \mathcal{L}(G')$.

Proof.

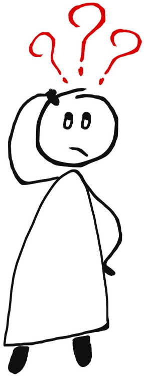
Let $G = \langle \Sigma, V, P, S \rangle$ be a grammar s.t. $P \subseteq V \times (\Sigma \cup \Sigma V \cup \{\varepsilon\})$.
Use the previous proof to construct grammar $G' = \langle \Sigma, V', P', S \rangle$ s.t. $P' \subseteq V' \times (\Sigma \cup \Sigma(V' \setminus \{S\}) \cup \{\varepsilon\})$.
Let $V_\varepsilon = \{A \mid A \to \varepsilon \in P'\}$.

Let P'' be the rule set that is created from P' by removing all rules of the form $A \to \varepsilon$ $(A \neq S)$. Additionally, for every rule of the form $B \to xA$ with $A \in V_\varepsilon, B \in V', x \in \Sigma$ we add a rule $B \to x$ to P''.

Then $G'' = \langle \Sigma, V', P'', S \rangle$ is regular and $\mathcal{L}(G) = \mathcal{L}(G'')$. □

Epsilon Rules: Example

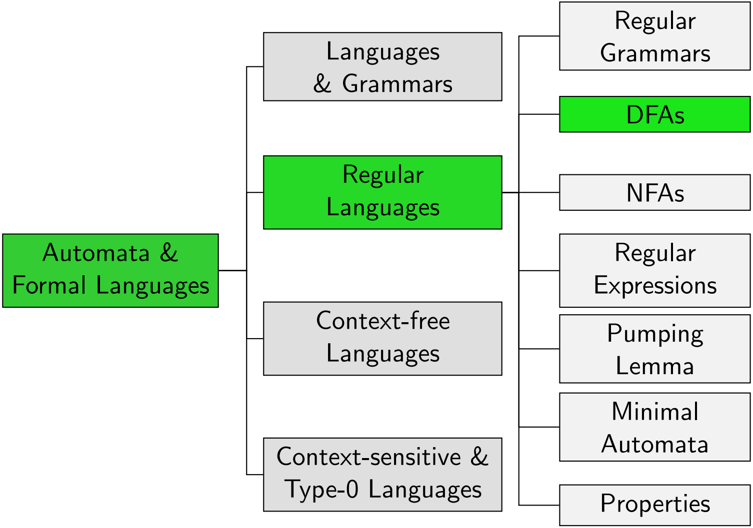
Questions



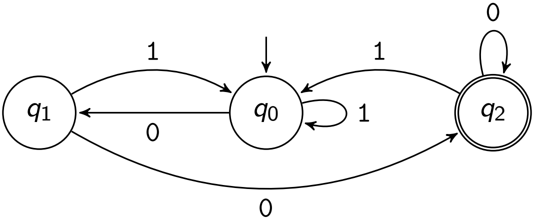
Questions?

DFAs

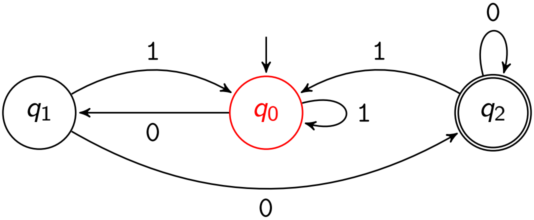
Overview


Automata & Formal Languages
Languages & Grammars
Regular Languages
Context-free Languages
Context-sensitive & Type-0 Languages
Regular Grammars
DFAs
NFAs
Regular Expressions
Pumping Lemma
Minimal Automata
Properties

Finite Automata: Example

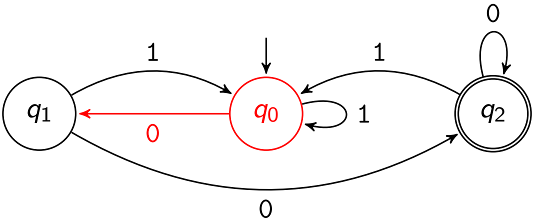


Finite Automata: Example



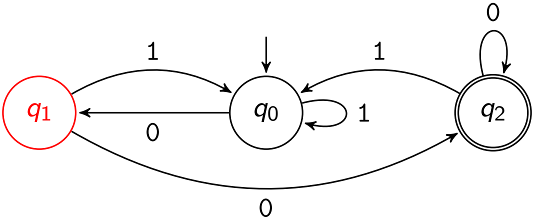
When reading the input 01100 the automaton visits the states q_0,

Finite Automata: Example



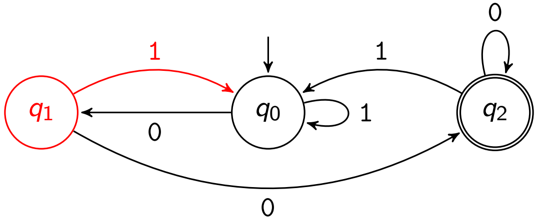
When reading the input 01100 the automaton visits the states q_0,

Finite Automata: Example



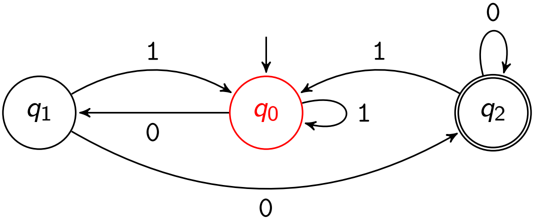
When reading the input 01100 the automaton visits the states
q_0, q_1,

Finite Automata: Example



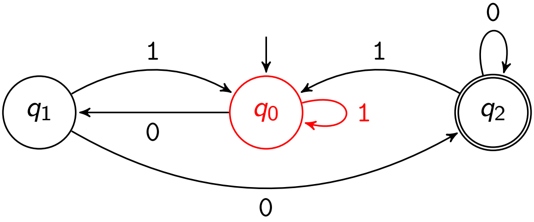
When reading the input 01100 the automaton visits the states q_0, q_1,

Finite Automata: Example



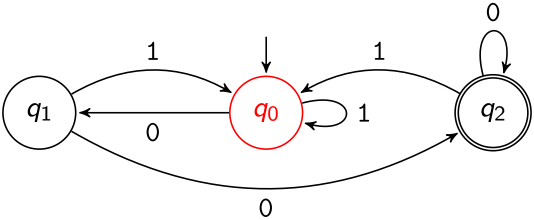
When reading the input 01100 the automaton visits the states q_0, q_1, q_0,

Finite Automata: Example



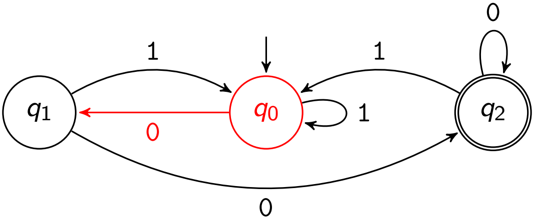
When reading the input 01100 the automaton visits the states q_0, q_1, q_0,

Finite Automata: Example



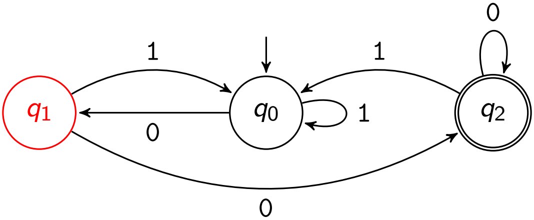
When reading the input 01100 the automaton visits the states
q_0, q_1, q_0, q_0,

Finite Automata: Example



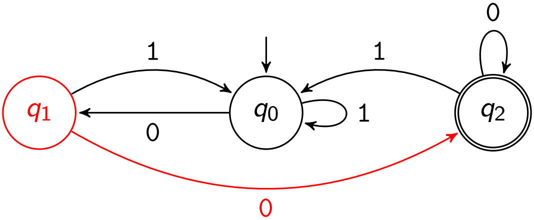
When reading the input 01100 the automaton visits the states
q_0, q_1, q_0, q_0,

Finite Automata: Example



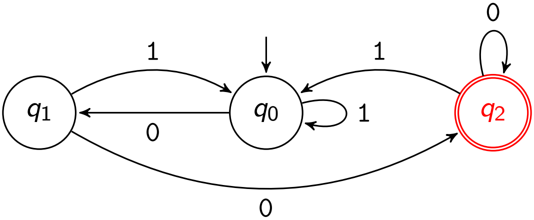
When reading the input 01100 the automaton visits the states q_0, q_1, q_0, q_0, q_1,

Finite Automata: Example



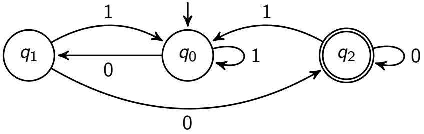
When reading the input 01100 the automaton visits the states q_0, q_1, q_0, q_0, q_1,

Finite Automata: Example

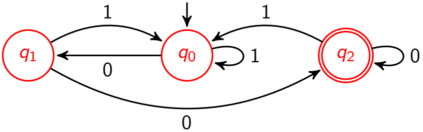


When reading the input 01100 the automaton visits the states q_0, q_1, q_0, q_0, q_1, q_2.

Finite Automata: Terminology and Notation

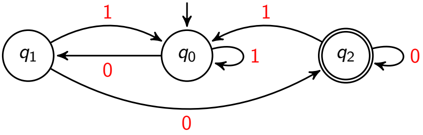


Finite Automata: Terminology and Notation



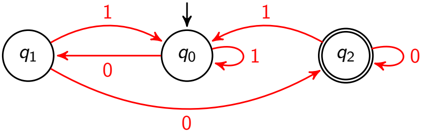
- states $Q = \{q_0, q_1, q_2\}$

Finite Automata: Terminology and Notation



- states $Q = \{q_0, q_1, q_2\}$
- input alphabet $\Sigma = \{0, 1\}$

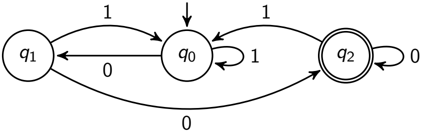
Finite Automata: Terminology and Notation



- states $Q = \{q_0, q_1, q_2\}$
- input alphabet $\Sigma = \{0, 1\}$
- transition function δ

$$\delta(q_0, 0) = q_1$$
$$\delta(q_0, 1) = q_0$$
$$\delta(q_1, 0) = q_2$$
$$\delta(q_1, 1) = q_0$$
$$\delta(q_2, 0) = q_2$$
$$\delta(q_2, 1) = q_0$$

Finite Automata: Terminology and Notation



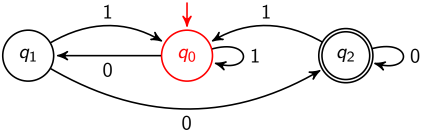
- states $Q = \{q_0, q_1, q_2\}$
- input alphabet $\Sigma = \{0, 1\}$
- transition function δ

$\delta(q_0, 0) = q_1$
$\delta(q_0, 1) = q_0$
$\delta(q_1, 0) = q_2$
$\delta(q_1, 1) = q_0$
$\delta(q_2, 0) = q_2$
$\delta(q_2, 1) = q_0$

δ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

table form of δ

Finite Automata: Terminology and Notation



- states $Q = \{q_0, q_1, q_2\}$
- input alphabet $\Sigma = \{0, 1\}$
- transition function δ
- start state q_0

$\delta(q_0, 0) = q_1$

$\delta(q_0, 1) = q_0$

$\delta(q_1, 0) = q_2$

$\delta(q_1, 1) = q_0$

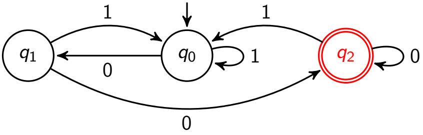
$\delta(q_2, 0) = q_2$

$\delta(q_2, 1) = q_0$

δ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

table form of δ

Finite Automata: Terminology and Notation



- states $Q = \{q_0, q_1, q_2\}$
- input alphabet $\Sigma = \{0, 1\}$
- transition function δ
- start state q_0
- end states $\{q_2\}$

$\delta(q_0, 0) = q_1$
$\delta(q_0, 1) = q_0$
$\delta(q_1, 0) = q_2$
$\delta(q_1, 1) = q_0$
$\delta(q_2, 0) = q_2$
$\delta(q_2, 1) = q_0$

δ	0	1
q_0	q_1	q_0
q_1	q_2	q_0
q_2	q_2	q_0

table form of δ

Deterministic Finite Automaton: Definition

Definition (Deterministic Finite Automata)

A deterministic finite automaton (DFA) is a 5-tuple $M = \langle Q, \Sigma, \delta, q_0, E \rangle$ where

- Q is the finite set of states
- Σ is the input alphabet (with $Q \cap \Sigma = \emptyset$)
- $\delta : Q \times \Sigma \to Q$ is the transition function
- $q_0 \in Q$ is the start state
- $E \subseteq Q$ is the set of end states

German: deterministischer endlicher Automat, Zustände, Eingabealphabet, Überführungs-/Übergangsfunktion, Startzustand, Endzustände

DFA: Recognized Words

Definition (Words Recognized by a DFA)

DFA $M = \langle Q, \Sigma, \delta, q_0, E \rangle$ recognizes the word $w = a_1 \dots a_n$ if there is a sequence of states $q'_0, \dots, q'_n \in Q$ with

1. $q'_0 = q_0$,
2. $\delta(q'_{i-1}, a_i) = q'_i$ for all $i \in \{1, \dots, n\}$ and
3. $q'_n \in E$.

German: DFA erkennt das Wort

DFA: Recognized Words

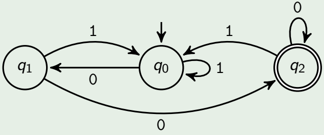
Definition (Words Recognized by a DFA)

DFA $M = \langle Q, \Sigma, \delta, q_0, E\rangle$ recognizes the word $w = a_1 \dots a_n$ if there is a sequence of states $q'_0, \dots, q'_n \in Q$ with

1. $q'_0 = q_0$,
2. $\delta(q'_{i-1}, a_i) = q'_i$ for all $i \in \{1, \dots, n\}$ and
3. $q'_n \in E$.

German: DFA erkennt das Wort

Example



recognizes:	does not recognize:
00	ε
10010100	1001010
01000	010001

DFA: Accepted Language

Definition (Language Accepted by a DFA)

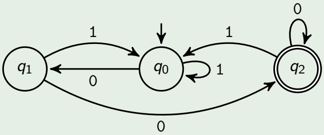
Let M be a deterministic finite automaton.
The language accepted by M is defined as
$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is recognized by } M\}$.

DFA: Accepted Language

Definition (Language Accepted by a DFA)

Let M be a deterministic finite automaton.
The language accepted by M is defined as
$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is recognized by } M\}$.

Example

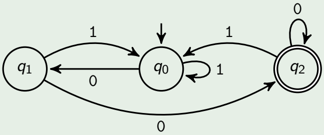


DFA: Accepted Language

Definition (Language Accepted by a DFA)

Let M be a deterministic finite automaton.
The language accepted by M is defined as
$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is recognized by } M\}$.

Example



The DFA accepts the language $\{w \in \{0,1\}^* \mid w \text{ ends with } 00\}$.

Languages Accepted by DFAs are Regular

Theorem

Every language accepted by a DFA is regular (type 3).

Languages Accepted by DFAs are Regular

Theorem

Every language accepted by a DFA is regular (type 3).

Proof.

Let $M = \langle Q, \Sigma, \delta, q_0, E \rangle$ be a DFA.
We define a regular grammar G with $\mathcal{L}(G) = \mathcal{L}(M)$.

Define $G = \langle \Sigma, Q, P, q_0 \rangle$ where P contains

- a rule $q \to aq'$ for every $\delta(q, a) = q'$, and
- a rule $q \to \varepsilon$ for every $q \in E$.

(We can eliminate forbidden epsilon rules
as described at the start of the chapter.) ...

Languages Accepted by DFAs are Regular

Theorem

Every language accepted by a DFA is regular (type 3).

Proof (continued).

For every $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$

iff there is a sequence of states $q'_0, q'_1, \dots, q'_n$ with
$q'_0 = q_0$, $q'_n \in E$ and $\delta(q'_{i-1}, a_i) = q'_i$ for all $i \in \{1, \dots, n\}$

iff there is a sequence of variables $q'_0, q'_1, \dots, q'_n$ with
q'_0 is start variable and we have $q'_0 \Rightarrow a_1 q'_1 \Rightarrow a_1 a_2 q'_2 \Rightarrow \dots \Rightarrow a_1 a_2 \dots a_n q'_n \Rightarrow a_1 a_2 \dots a_n$.

iff $w \in \mathcal{L}(G)$ □

Languages Accepted by DFAs are Regular

Theorem

Every language accepted by a DFA is regular (type 3).

Proof (continued).

For every $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$

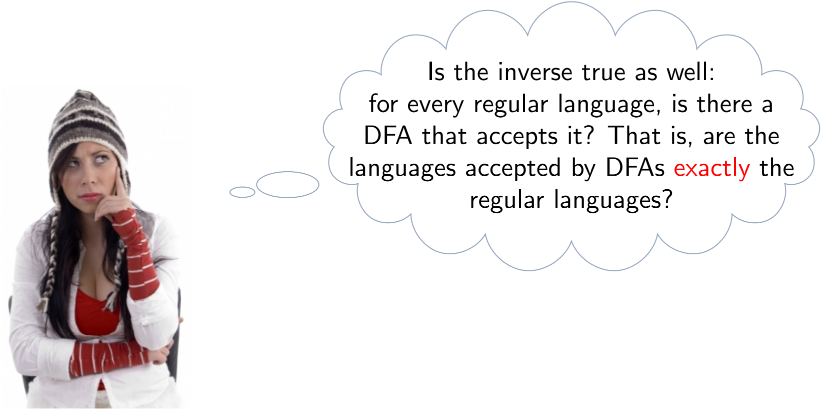
iff there is a sequence of states $q'_0, q'_1, \dots, q'_n$ with
$q'_0 = q_0$, $q'_n \in E$ and $\delta(q'_{i-1}, a_i) = q'_i$ for all $i \in \{1, \dots, n\}$

iff there is a sequence of variables $q'_0, q'_1, \dots, q'_n$ with
q'_0 is start variable and we have $q'_0 \Rightarrow a_1 q'_1 \Rightarrow a_1 a_2 q'_2 \Rightarrow \dots \Rightarrow a_1 a_2 \dots a_n q'_n \Rightarrow a_1 a_2 \dots a_n$.

iff $w \in \mathcal{L}(G)$ □

Example: blackboard

Question



Picture courtesy of imagerymajestic / FreeDigitalPhotos.net

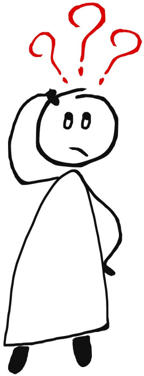
Question

Is the inverse true as well:
for every regular language, is there a DFA that accepts it? That is, are the languages accepted by DFAs exactly the regular languages?

Yes!
We will prove this later (via a detour).

Picture courtesy of imagerymajestic / FreeDigitalPhotos.net

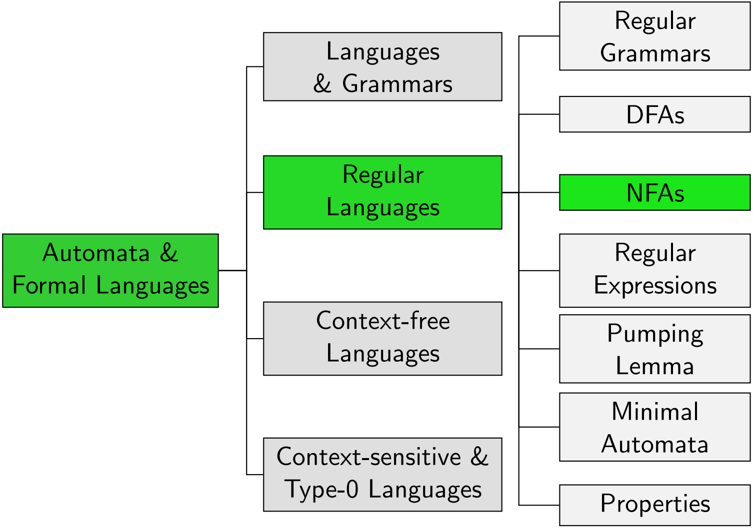
Questions



Questions?

NFAs

Overview

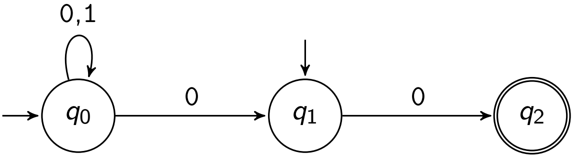

Automata &
Formal Languages
Languages
& Grammars
Regular
Languages
Context-free
Languages
Context-sensitive &
Type-0 Languages
Regular
Grammars
DFAs
NFAs
Regular
Expressions
Pumping
Lemma
Minimal
Automata
Properties

Nondeterministic Finite Automata

Why are DFAs called deterministic automata? What are nondeterministic automata, then?

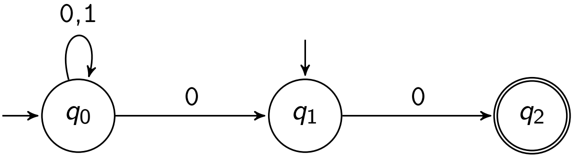
Picture courtesy of stockimages / FreeDigitalPhotos.net

Nondeterministic Finite Automata: Example



differences to DFAs:

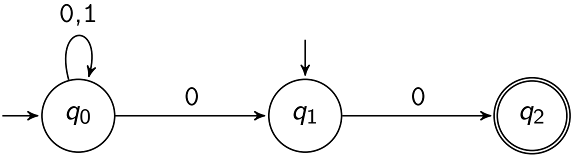
Nondeterministic Finite Automata: Example



differences to DFAs:

- multiple start states possible

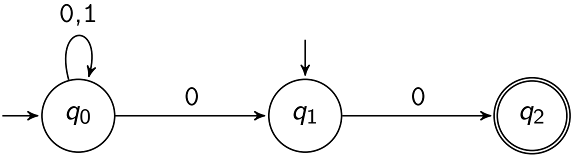
Nondeterministic Finite Automata: Example



differences to DFAs:

- multiple start states possible
- transition function δ can lead to zero or more successor states for the same $a \in \Sigma$

Nondeterministic Finite Automata: Example



differences to DFAs:

- multiple start states possible
- transition function δ can lead to zero or more successor states for the same $a \in \Sigma$
- automaton recognizes a word if there is at least one accepting sequence of states

Nondeterministic Finite Automaton: Definition

Definition (Nondeterministic Finite Automata)

A nondeterministic finite automaton (NFA) is a 5-tuple $M = \langle Q, \Sigma, \delta, S, E \rangle$ where

- Q is the finite set of states
- Σ is the input alphabet (with $Q \cap \Sigma = \emptyset$)
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the transition function (mapping to the power set of Q)
- $S \subseteq Q$ is the set of start states
- $E \subseteq Q$ is the set of end states

German: nichtdeterministischer endlicher Automat

Nondeterministic Finite Automaton: Definition

Definition (Nondeterministic Finite Automata)

A nondeterministic finite automaton (NFA) is a 5-tuple $M = \langle Q, \Sigma, \delta, S, E \rangle$ where

- Q is the finite set of states
- Σ is the input alphabet (with $Q \cap \Sigma = \emptyset$)
- $\delta : Q \times \Sigma \to \mathcal{P}(Q)$ is the transition function (mapping to the power set of Q)
- $S \subseteq Q$ is the set of start states
- $E \subseteq Q$ is the set of end states

German: nichtdeterministischer endlicher Automat

DFAs are (essentially) a special case of NFAs.

NFA: Recognized Words

Definition (Words Recognized by an NFA)

NFA $M = \langle Q, \Sigma, \delta, S, E \rangle$ recognizes the word $w = a_1 \dots a_n$ if there is a sequence of states $q'_0, \dots, q'_n \in Q$ with

1. $q'_0 \in S$,
2. $q'_i \in \delta(q'_{i-1}, a_i)$ for all $i \in \{1, \dots, n\}$ and
3. $q'_n \in E$.

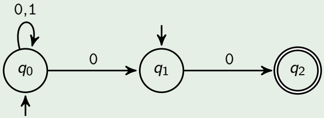
NFA: Recognized Words

Definition (Words Recognized by an NFA)

NFA $M = \langle Q, \Sigma, \delta, S, E \rangle$ recognizes the word $w = a_1 \dots a_n$ if there is a sequence of states $q'_0, \dots, q'_n \in Q$ with

1. $q'_0 \in S$,
2. $q'_i \in \delta(q'_{i-1}, a_i)$ for all $i \in \{1, \dots, n\}$ and
3. $q'_n \in E$.

Example



recognizes:
0
10010100
01000

does not recognize:
ε
1001010
010001

NFA: Accepted Language

Definition (Language Accepted by an NFA)

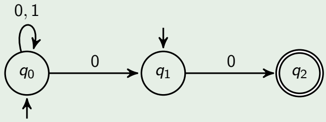
Let $M = \langle Q, \Sigma, \delta, S, E \rangle$ be a nondeterministic finite automaton.
The language accepted by M is defined as
$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is recognized by } M\}$.

NFA: Accepted Language

Definition (Language Accepted by an NFA)

Let $M = \langle Q, \Sigma, \delta, S, E \rangle$ be a nondeterministic finite automaton.
The language accepted by M is defined as
$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is recognized by } M\}$.

Example

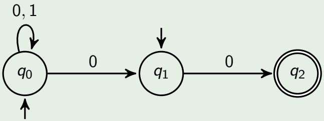


NFA: Accepted Language

Definition (Language Accepted by an NFA)

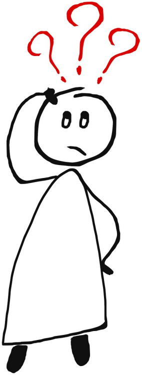
Let $M = \langle Q, \Sigma, \delta, S, E \rangle$ be a nondeterministic finite automaton.
The language accepted by M is defined as
$\mathcal{L}(M) = \{w \in \Sigma^* \mid w \text{ is recognized by } M\}$.

Example



The NFA accepts the language
$\{w \in \{0,1\}^* \mid w = 0 \text{ or } w \text{ ends with } 00\}$.

Questions



Questions?

NFAs are No More Powerful than DFAs

Theorem (Rabin, Scott)

Every language accepted by an NFA is also accepted by a DFA.

NFAs are No More Powerful than DFAs

Theorem (Rabin, Scott)

Every language accepted by an NFA is also accepted by a DFA.

Proof.

For every NFA $M = \langle Q, \Sigma, \delta, S, E \rangle$ we can construct a DFA $M' = \langle Q', \Sigma, \delta', q_0', E' \rangle$ with $\mathcal{L}(M) = \mathcal{L}(M')$.
Here M' is defined as follows:

- $Q' := \mathcal{P}(Q)$ (the power set of Q)
- $q_0' := S$
- $E' := \{\mathcal{Q} \subseteq Q \mid \mathcal{Q} \cap E \neq \emptyset\}$
- For all $\mathcal{Q} \in Q'$: $\delta'(\mathcal{Q}, a) := \bigcup_{q \in \mathcal{Q}} \delta(q, a)$

...

NFAs are No More Powerful than DFAs

Theorem (Rabin, Scott)

Every language accepted by an NFA is also accepted by a DFA.

Proof (continued).

For every $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$
iff there is a sequence of states $q_0, q_1, \dots, q_n$ with
$q_0 \in S$, $q_n \in E$ and $q_i \in \delta(q_{i-1}, a_i)$ for all $i \in \{1, \dots, n\}$
iff there is a sequence of subsets $\mathcal{Q}_0, \mathcal{Q}_1, \dots, \mathcal{Q}_n$ with
$\mathcal{Q}_0 = q_0'$, $\mathcal{Q}_n \in E'$ and $\delta'(\mathcal{Q}_{i-1}, a_i) = \mathcal{Q}_i$ for all $i \in \{1, \dots, n\}$
iff $w \in \mathcal{L}(M')$ □

NFAs are No More Powerful than DFAs

Theorem (Rabin, Scott)

Every language accepted by an NFA is also accepted by a DFA.

Proof (continued).

For every $w = a_1 a_2 \dots a_n \in \Sigma^*$:

$w \in \mathcal{L}(M)$
iff there is a sequence of states $q_0, q_1, \dots, q_n$ with
$q_0 \in S$, $q_n \in E$ and $q_i \in \delta(q_{i-1}, a_i)$ for all $i \in \{1, \dots, n\}$
iff there is a sequence of subsets $\mathcal{Q}_0, \mathcal{Q}_1, \dots, \mathcal{Q}_n$ with
$\mathcal{Q}_0 = q_0'$, $\mathcal{Q}_n \in E'$ and $\delta'(\mathcal{Q}_{i-1}, a_i) = \mathcal{Q}_i$ for all $i \in \{1, \dots, n\}$
iff $w \in \mathcal{L}(M')$ □

Example: blackboard

NFAs are More Compact than DFAs

Example

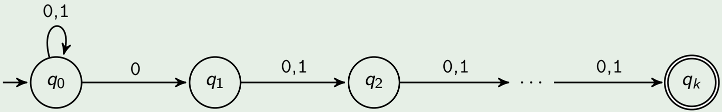
For $k \geq 1$ consider the language
$L_k = \{w \in \{0,1\}^* \mid |w| \geq k$ and the k-th last symbol of w is $0\}$.

NFAs are More Compact than DFAs

Example

For $k \geq 1$ consider the language
$L_k = \{w \in \{0,1\}^* \mid |w| \geq k$ and the k-th last symbol of w is $0\}$.

The language L_k can be accepted by an NFA with $k+1$ states:

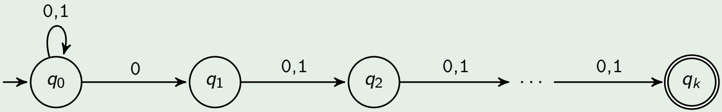


NFAs are More Compact than DFAs

Example

For $k \geq 1$ consider the language
$L_k = \{w \in \{0, 1\}^* \mid |w| \geq k \text{ and the } k\text{-th last symbol of } w \text{ is } 0\}$.

The language L_k can be accepted by an NFA with $k + 1$ states:



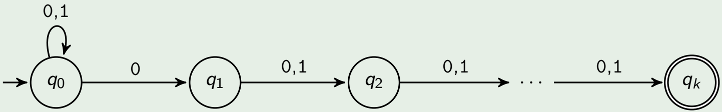
There is no DFA with less than 2^k states that accepts L_k (without proof).

NFAs are More Compact than DFAs

Example

For $k \geq 1$ consider the language
$L_k = \{w \in \{0, 1\}^* \mid |w| \geq k$ and the k-th last symbol of w is $0\}$.

The language L_k can be accepted by an NFA with $k + 1$ states:



There is no DFA with less than 2^k states that accepts L_k (without proof).

NFAs can often represent languages more compactly than DFAs.

Regular Grammars are No More Powerful than NFAs

Theorem

For every regular grammar G there is an NFA M with $\mathcal{L}(G) = \mathcal{L}(M)$.

Regular Grammars are No More Powerful than NFAs

Theorem

For every regular grammar G there is an NFA M with $\mathcal{L}(G) = \mathcal{L}(M)$.

Proof.

Let $G = \langle \Sigma, V, P, S \rangle$ be a regular grammar.
Define NFA $M = \langle Q, \Sigma, \delta, S', E \rangle$ with

$$Q = V \cup \{X\}, \quad X \notin V$$

$$S' = \{S\}$$

$$E = \begin{cases} \{S, X\} & \text{if } S \to \varepsilon \in P \\ \{X\} & \text{if } S \to \varepsilon \notin P \end{cases}$$

$$B \in \delta(A, a) \text{ if } A \to aB \in P$$

$$X \in \delta(A, a) \text{ if } A \to a \in P$$

Regular Grammars are No More Powerful than NFAs

Theorem

For every regular grammar G there is an NFA M with $\mathcal{L}(G) = \mathcal{L}(M)$.

Proof (continued).

For every $w = a_1 a_2 \dots a_n \in \Sigma^*$ with $n \geq 1$:

$w \in \mathcal{L}(G)$

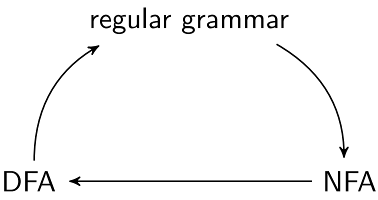
iff there is a sequence on variables $A_1, A_2, \dots, A_{n-1}$ with
$S \Rightarrow a_1 A_1 \Rightarrow a_1 a_2 A_2 \Rightarrow \cdots \Rightarrow a_1 a_2 \dots a_{n-1} A_{n-1} \Rightarrow a_1 a_2 \dots a_n$.

iff there is a sequence of variables $A_1, A_2, \dots, A_{n-1}$ with
$A_1 \in \delta(S, a_1), A_2 \in \delta(A_1, a_2), \dots, X \in \delta(A_{n-1}, a_n)$.

iff $w \in \mathcal{L}(M)$.

Case $w = \varepsilon$ is also covered because $S \in E$ iff $S \to \varepsilon \in P$. □

Finite Automata and Regular Languages

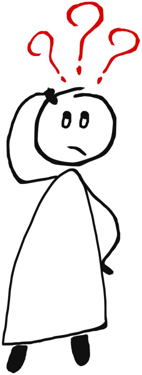


In particular, this implies:

Corollary

$\mathcal{L}$ regular $\iff$ $\mathcal{L}$ is accepted by a DFA.
$\mathcal{L}$ regular $\iff$ $\mathcal{L}$ is accepted by an NFA.

Questions



Questions?

Summary

Summary

- We now know three formalisms that all describe exactly the regular languages: regular grammars, DFAs and NFAs
- We will get to know a fourth formalism in the next chapter.
- DFAs are automata where every state transition is uniquely determined.
- NFAs recognize a word if there is at least one accepting sequence of states.