Automata and Formal Language Theory

Stefan Hetzl
Institute of Discrete Mathematics and Geometry
Vienna University of Technology

9th International Tbilisi Summer School in Logic and Language

Tbilisi, Georgia

September 2013

Introduction

- Formal and natural languages

Introduction

- Formal and natural languages
- How to specify a formal language?
  - Automata
  - Grammars

Introduction

- Formal and natural languages
- How to specify a formal language?
  - Automata
  - Grammars
- Strong connections to:
  - Computability theory
  - Complexity theory

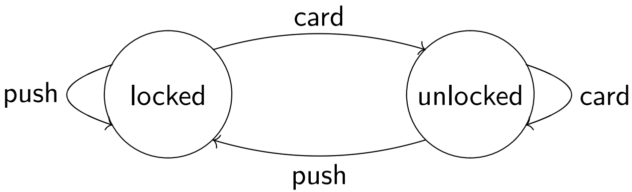
Introduction

- Formal and natural languages
- How to specify a formal language?
  - Automata
  - Grammars
- Strong connections to:
  - Computability theory
  - Complexity theory
- Applications in computer science:
  - Verification
  - Compiler construction
  - Data formats

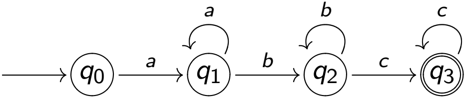
Outline

- Deterministic finite automata
- Nondeterministic finite automata
- Automata with ε-transitions
- The class of regular languages
- The pumping lemma for regular languages
- Context-free grammars and languages
- Right linear grammars
- Pushdown Automata
- The pumping lemma for context-free languages
- Grammars in computer science
- Further topics

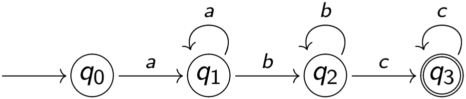
Finite Automata – A First Example



Finite Automata – A More Abstract Example

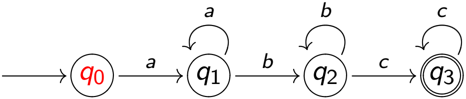


Finite Automata – A More Abstract Example



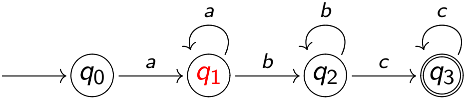
abbbcc

Finite Automata – A More Abstract Example



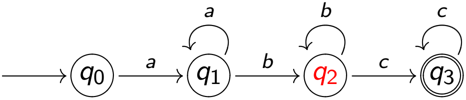
$abbbcc$

Finite Automata – A More Abstract Example



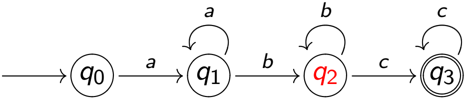
$abbbcc$

Finite Automata – A More Abstract Example



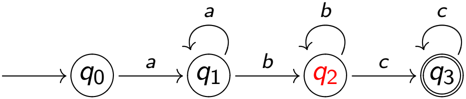
$abbbcc$

Finite Automata – A More Abstract Example



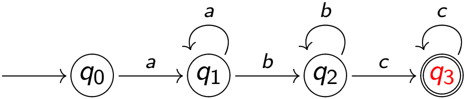
$abbbcc$

Finite Automata – A More Abstract Example



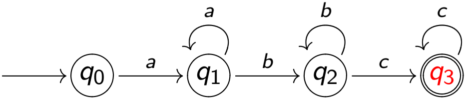
$abbbcc$

Finite Automata – A More Abstract Example



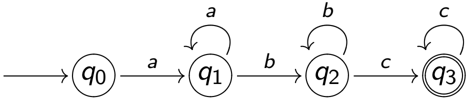
$abbbcc$

Finite Automata – A More Abstract Example



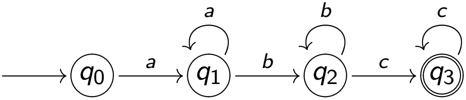
$abbbcc$

Finite Automata – A More Abstract Example



$abbbcc$ ✓

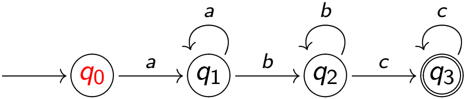
Finite Automata – A More Abstract Example



$abbbcc$ ✓

aab

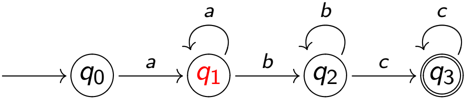
Finite Automata – A More Abstract Example



$abbbcc$ ✓

aab

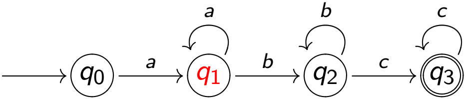
Finite Automata – A More Abstract Example



$abbbcc$ ✓

aab

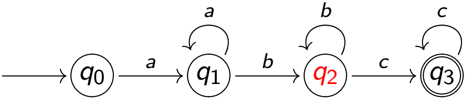
Finite Automata – A More Abstract Example



$abbbcc$ ✓

aab

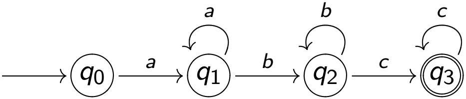
Finite Automata – A More Abstract Example



abbbcc ✓

aab

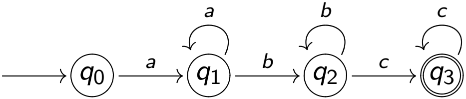
Finite Automata – A More Abstract Example



$abbbcc$ ✓

aab ✗

Finite Automata – A More Abstract Example

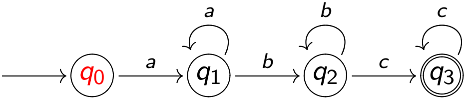


$abbbcc$ ✓

aab ✗

ac

Finite Automata – A More Abstract Example

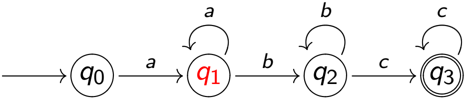


$abbbcc$ ✓

aab ✗

ac

Finite Automata – A More Abstract Example

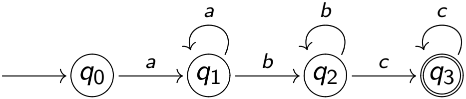


$abbbcc$ ✓

aab ✗

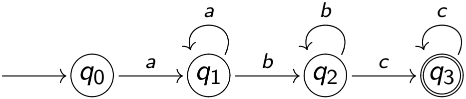
ac

Finite Automata – A More Abstract Example



abbbcc	✓
aab	✗
ac	✗

Finite Automata – A More Abstract Example

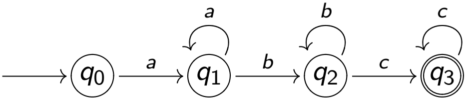


$abbbcc$ ✓
aab ✗
ac ✗

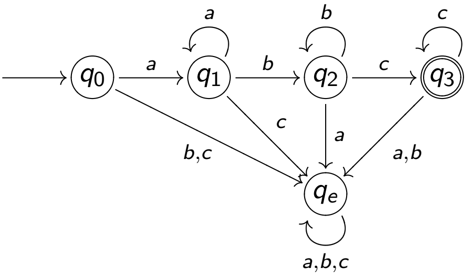
The language *accepted* by this automaton is

$$L = \{a^k b^n c^m \mid k, n, m \geq 1\}$$

The Error State



The Error State



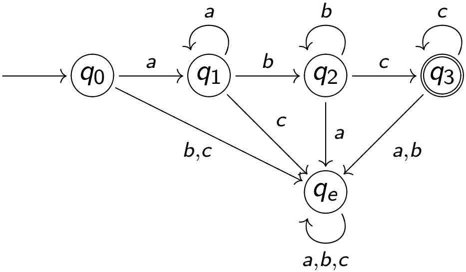
Deterministic Finite Automata – Definition

Definition

A *deterministic finite automaton (DFA)* is a tuple $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ where:

1. Q is a finite set (the *states*).
2. Σ is a finite set (the *input symbols*).
3. $\delta : Q \times \Sigma \to Q$ (the *transition function*).
4. $q_0 \in Q$ (the *starting state*)
5. $F \subseteq Q$ (the *final states*).

DFA – Example



as tuple: BB1

The Language of a DFA

Definition

Extend $\delta : Q \times \Sigma \to Q$ to $\hat{\delta} : Q \times \Sigma^* \to Q$ as follows.

$$\hat{\delta}(q, w) = \begin{cases} q & \text{if } w = \varepsilon \\ \delta(\hat{\delta}(q, v), x) & \text{if } w = vx \text{ for } v \in \Sigma^*, x \in \Sigma \end{cases}$$

The Language of a DFA

Definition

Extend $\delta : Q \times \Sigma \to Q$ to $\hat{\delta} : Q \times \Sigma^* \to Q$ as follows.

$$\hat{\delta}(q, w) = \begin{cases} q & \text{if } w = \varepsilon \\ \delta(\hat{\delta}(q, v), x) & \text{if } w = vx \text{ for } v \in \Sigma^*, x \in \Sigma \end{cases}$$

Example

$\hat{\delta}(abc, q_0) = q_3, \ \hat{\delta}(aba, q_0) = q_e$

The Language of a DFA

Definition

Extend $\delta : Q \times \Sigma \to Q$ to $\hat{\delta} : Q \times \Sigma^* \to Q$ as follows.

$$\hat{\delta}(q, w) = \begin{cases} q & \text{if } w = \varepsilon \\ \delta(\hat{\delta}(q, v), x) & \text{if } w = vx \text{ for } v \in \Sigma^*, x \in \Sigma \end{cases}$$

Example

$\hat{\delta}(abc, q_0) = q_3,\ \hat{\delta}(aba, q_0) = q_e$

Definition

Let $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ be a DFA. The language accepted by A is

$$\mathrm{L}(A) = \{w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F\}.$$

Designing an DFA

$$L = \{w \in \{a, b\}^* \mid w \text{ contains an even number of } a\text{'s and an even number of } b\text{'s}\}$$

BB2

Outline

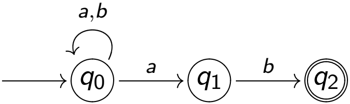
- ✓ Deterministic finite automata
- ⇒ Nondeterministic finite automata
- Automata with ε-transitions
- The class of regular languages
- The pumping lemma for regular languages
- Context-free grammars and languages
- Right linear grammars
- Pushdown Automata
- The pumping lemma for context-free languages
- Grammars in computer science
- Further topics

Nondeterministic Finite Automata – A Motivating Example

Automaton for accepting $L = \{wab \mid w \in \{a, b\}^*\}$?

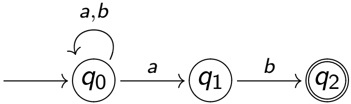
Nondeterministic Finite Automata – A Motivating Example

Automaton for accepting $L = \{wab \mid w \in \{a, b\}^*\}$?



Nondeterministic Finite Automata – A Motivating Example

Automaton for accepting $L = \{wab \mid w \in \{a, b\}^*\}$?



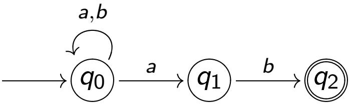
Nondeterminism $\implies$ consider *all* possible runs

Definition

A *nondeterministic finite automaton (NFA)* is a tuple $A = \langle Q, \Sigma, \Delta, q_0, F \rangle$ where:

1. Q is a finite set (the *states*).
2. Σ is a finite set (the *input symbols*).
3. $\Delta \subseteq Q \times \Sigma \times Q$ (the *transition* **relation**).
4. $q_0 \in Q$ (the *starting state*)
5. $F \subseteq Q$ (the *final states*).

NFA – Example



as tuple: BB3

The Language of an NFA

Definition

Extend $\Delta \subseteq Q \times \Sigma \times Q$ to $\hat{\Delta} : Q \times \Sigma^* \times Q$ as follows.

$$
\begin{aligned}
(q, w, q) \in \hat{\Delta} &\quad \text{if } w = \varepsilon \\
(q, w, q^*) \in \hat{\Delta} &\quad \text{if } w = vx \text{ for } v \in \Sigma^*, x \in \Sigma, \\
&\qquad \text{and } (q, v, q') \in \hat{\Delta}, \text{and } (q', x, q) \in \Delta
\end{aligned}
$$

The Language of an NFA

Definition

Extend $\Delta \subseteq Q \times \Sigma \times Q$ to $\hat{\Delta} : Q \times \Sigma^* \times Q$ as follows.

$$
\begin{aligned}
(q, w, q) \in \hat{\Delta} \quad & \text{if } w = \varepsilon \\
(q, w, q^*) \in \hat{\Delta} \quad & \text{if } w = vx \text{ for } v \in \Sigma^*, x \in \Sigma, \\
& \qquad \text{and } (q, v, q') \in \hat{\Delta}, \text{and } (q', x, q) \in \Delta
\end{aligned}
$$

Example

$(q_0, ab, q_0), (q_0, ab, q_2) \in \hat{\Delta},\ (q_0, bb, q_0) \in \hat{\Delta}$

The Language of an NFA

Definition

Extend $\Delta \subseteq Q \times \Sigma \times Q$ to $\hat{\Delta} : Q \times \Sigma^* \times Q$ as follows.

$$\begin{aligned} (q, w, q) \in \hat{\Delta} \quad & \text{if } w = \varepsilon \\ (q, w, q^*) \in \hat{\Delta} \quad & \text{if } w = vx \text{ for } v \in \Sigma^*, x \in \Sigma, \\ & \quad \text{and } (q, v, q') \in \hat{\Delta}, \text{and } (q', x, q) \in \Delta \end{aligned}$$

Example

$(q_0, ab, q_0), (q_0, ab, q_2) \in \hat{\Delta},\ (q_0, bb, q_0) \in \hat{\Delta}$

Definition

Let $A = \langle Q, \Sigma, \Delta, q_0, F\rangle$ be a NFA. The language accepted by A is

$$\mathrm{L}(A) = \{w \in \Sigma^* \mid \exists q \in F \text{ s.t. } (q_0, w, q) \in \hat{\Delta}\}.$$

Theorem

Let $L \subseteq \Sigma^*$, then there is a DFA D with $\mathrm{L}(D) = L$ iff there is a NFA N with $\mathrm{L}(N) = L$.

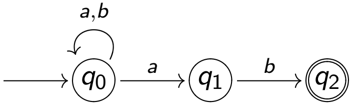
Proof (BB4).

1. Converting D to N: easy.
2. Converting N to D: subset construction.

$\square$

The Subset Construction – Example

Automaton for accepting $L = \{wab \mid w \in \{a, b\}^*\}$:



Conversion to DFA: BB5

In practice: only construct *reachable* states

Theorem

There are $L_n \subseteq \Sigma^*, n \geq 1$ and NFA N_n with $n + 1$ states with $\mathsf{L}(N_n) = L_n$ s.t. all DFA D_n with $\mathsf{L}(D_n) = L_n$ have at least 2^n states.

Proof.

Let $\Sigma = \{a, b\}$ and for $n \geq 1$ define

$$L_n = \{wav \mid w, v \in \Sigma^*, |v| = n - 1\}$$

BB6 $\square$

Outline

- ✓ Deterministic finite automata
- ✓ Nondeterministic finite automata
- ⇒ Automata with ε-transitions
- ▶ The class of regular languages
- ▶ The pumping lemma for regular languages
- ▶ Context-free grammars and languages
- ▶ Right linear grammars
- ▶ Pushdown Automata
- ▶ The pumping lemma for context-free languages
- ▶ Grammars in computer science
- ▶ Further topics

Epsilon-Transitions – A Motivating Example

Automaton for accepting decimal representations of integers:

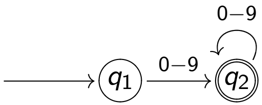
$$\Sigma = \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$L = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^{+} \\ \cup \{-w \mid w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^{+}\}$$

Epsilon-Transitions – A Motivating Example

Automaton for accepting decimal representations of integers:

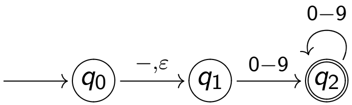
$$
\begin{aligned}
\Sigma &= \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
L &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+ \\
&\quad \cup \{-w \mid w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+\}
\end{aligned}
$$



Epsilon-Transitions – A Motivating Example

Automaton for accepting decimal representations of integers:

$$
\begin{aligned}
\Sigma &= \{-, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
L &= \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^{+} \\
&\quad \cup \{-w \mid w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^{+}\}
\end{aligned}
$$



Definition

A *nondeterministic finite automaton with ε-transitions (ε-NFA)* is a tuple $A = \langle Q, \Sigma, \Delta, q_0, F \rangle$ where:

1. Q is a finite set (the *states*).
2. Σ is a finite set (the *input symbols*).
3. $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times Q$ (the *transition relation*).
4. $q_0 \in Q$ (the *starting state*)
5. $F \subseteq Q$ (the *final states*).

ε-NFA – Properties

Definition

Transition relation $\hat{\Delta}$: includes ε-transitions.

Definition

Let $A = \langle Q, \Sigma, \Delta, q_0, F \rangle$ be a ε-NFA. The language accepted by A is

$$\mathrm{L}(A) = \{w \in \Sigma^* \mid \exists q \in F \text{ s.t. } (q_0, w, q) \in \hat{\Delta}\}.$$

Theorem

Let $L \subseteq \Sigma^*$, then there is a DFA D with $\mathrm{L}(D) = L$ iff there is a ε-NFA N with $\mathrm{L}(N) = L$.

Proof.

Modified subset construction (ε-closed subsets). $\square$

Outline

- ✓ Deterministic finite automata
- ✓ Nondeterministic finite automata
- ✓ Automata with ε-transitions
- ⇒ The class of regular languages
- The pumping lemma for regular languages
- Context-free grammars and languages
- Right linear grammars
- Pushdown Automata
- The pumping lemma for context-free languages
- Grammars in computer science
- Further topics

The Class of Regular Languages

Corollary

Let $L \subseteq \Sigma^*$. *The following are equivalent:*

- *There is a DFA* D *with* $\mathrm{L}(D) = L$.
- *There is a NFA* N *with* $\mathrm{L}(N) = L$.
- *There is a* ε*-NFA* N' *with* $\mathrm{L}(N') = L$.

Definition

$L \subseteq \Sigma^*$ is called *regular language* if there is a finite automaton A with $\mathrm{L}(A) = L$.

Closure Properties of Regular Languages

Theorem

If $L_1, L_2 \subseteq \Sigma^*$ are regular, then $L_1 \cup L_2$ is regular.

Proof.

BB7 $\square$

Theorem

If $L_1, L_2 \subseteq \Sigma^*$ are regular, then $L_1 \cup L_2$ is regular.

Proof.

BB7 $\square$

Theorem

If $L \subseteq \Sigma^*$ is regular, then $L^{\mathrm{c}} = \Sigma^* \setminus L$ is regular.

Proof.

BB8 $\square$

Theorem

If $L_1, L_2 \subseteq \Sigma^*$ are regular, then $L_1 \cup L_2$ is regular.

Proof.

BB7 $\square$

Theorem

If $L \subseteq \Sigma^*$ is regular, then $L^{\mathrm{c}} = \Sigma^* \setminus L$ is regular.

Proof.

BB8 $\square$

Theorem

If $L_1, L_2 \subseteq \Sigma^*$ are regular, then $L_1 \cap L_2$ is regular.

Proof.

$L_1 \cap L_2 = (L_1^{\mathrm{c}} \cup L_2^{\mathrm{c}})^{\mathrm{c}}$. $\square$

Outline

- ✓ Deterministic finite automata
- ✓ Nondeterministic finite automata
- ✓ Automata with ε-transitions
- ✓ The class of regular languages
- $\Rightarrow$ The pumping lemma for regular languages
- Context-free grammars and languages
- Right linear grammars
- Pushdown Automata
- The pumping lemma for context-free languages
- Grammars in computer science
- Further topics

The Pumping Lemma for Regular Languages

Lemma (Pumping Lemma)

Let L be a regular language. Then there is an $n \in \mathbb{N}$ s.t. for every $w \in L$ with $|w| \geq n$ we have $w = v_1 v_2 v_3$ with

1. $v_2 \neq \varepsilon$,
2. $|v_1 v_2| \leq n$, *and*
3. *for all* $k \geq 0$ *also* $v_1 v_2^k v_3 \in L$.

Proof.

BB9 $\square$

Lemma (Pumping Lemma)

Let L be a regular language. Then there is an $n \in \mathbb{N}$ s.t. for every $w \in L$ with $|w| \geq n$ we have $w = v_1 v_2 v_3$ with

1. $v_2 \neq \varepsilon$,
2. $|v_1 v_2| \leq n$, *and*
3. *for all* $k \geq 0$ *also* $v_1 v_2^k v_3 \in L$.

Example

$L = \{a^m b^m \mid m \geq 1\}$ is not regular (BB10).

Outline

- ✓ Deterministic finite automata
- ✓ Nondeterministic finite automata
- ✓ Automata with ε-transitions
- ✓ The class of regular languages
- ✓ The pumping lemma for regular languages
- $\Rightarrow$ Context-free grammars and languages
- Right linear grammars
- Pushdown Automata
- The pumping lemma for context-free languages
- Grammars in computer science
- Further topics

Context-Free Grammars – A First Example

How can we specify the set of all arithmetical expressions?
E.g. $12, 30 + 21 \cdot 6, (123 + 7) \cdot 15 + 88, \ldots$

$$\begin{aligned}
E &\to N \mid E + E \mid E \cdot E \mid (E) \\
N &\to D \mid DN \\
D &\to 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}$$

Context-Free Grammars – Definition

Definition

A *context-free grammar (CFG)* is a tuple $G = \langle N, T, P, S \rangle$ where

1. N is a finite set of symbols (the *nonterminals*),
2. T is a finite set of symbols (the *terminals*),
3. P is a finite set of *production rules* of the form:

$$A \rightarrow w \text{ where } A \in N \text{ and } w \in (N \cup T)^*$$

4. $S \in N$ (the *start symbol*).

$$
\begin{aligned}
G &= \langle NT, T, P, S\rangle \\
NT &= \{N, D, E\} \\
T &= \{+, \cdot, (,), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \\
P &= \{E \to N, E \to E + E, E \to E \cdot E, E \to (E), \\
&\qquad N \to D, N \to DN, \\
&\qquad D \to 0, D \to 1, D \to 2, D \to 3, D \to 4, \\
&\qquad D \to 5, D \to 6, D \to 7, D \to 8, D \to 9\} \\
S &= E
\end{aligned}
$$

The Language of a Grammar

Let $G = \langle N, T, P, S \rangle$ be a CFG.

Definition

For every $A \to w \in P$ and every $uAv \in (N \cup T)^*$ define

$$uAv \Rightarrow_G uwv.$$

The *derivation relation* $\Rightarrow_G^*$ is the reflexive and transitive closure of $\Rightarrow_G$.

Definition

The language of G is $\mathrm{L}(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$.

Definition

$L \subseteq \Sigma^*$ is called *context-free* if there is a context-free grammar G with $\mathrm{L}(G) = L$.

Context-Free Grammars – Example Derivation

$$
\begin{aligned}
E &\rightarrow N \mid E + E \mid E \cdot E \mid (E) \\
N &\rightarrow D \mid DN \\
D &\rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{aligned}
$$

Example derivation: BB11

Formalisms

	Automata	Grammars
Regular Languages	DFAs, NFAs	?
Context-Free Languages	?	CFG

Outline

- ✓ Deterministic finite automata
- ✓ Nondeterministic finite automata
- ✓ Automata with ε-transitions
- ✓ The class of regular languages
- ✓ The pumping lemma for regular languages
- ✓ Context-free grammars and languages
- $\Rightarrow$ Right linear grammars
- Pushdown Automata
- The pumping lemma for context-free languages
- Grammars in computer science
- Further topics

Right Linear Grammars

Definition

A grammar $G = \langle N, T, P, S \rangle$ is called *right linear* if all productions are of one of the following forms:

$$A \to xB \text{ where } x \in T, B \in N$$
$$A \to x \text{ where } x \in T$$
$$A \to \varepsilon$$

Theorem

Let $L \subseteq \Sigma^*$. Then L is regular iff L has a right linear grammar.

Proof (BB12).

1. From right linear grammar to ε-NFA.
2. From NFA to right linear grammar.

□

Remark: notion of left linear grammars with analogous result

Right Linear Grammars – Example

Let $L = \{a^m b^n \mid m, n \geq 0\}$ (BB13)

Right linear grammar

Automaton

Formalisms

	Automata	Grammars
Regular Languages	DFA, NFA	LLG, RLG
Context-Free Languages	?	CFG

Regular vs. Context-Free Languages

Theorem

Every regular language is context-free.

Proof.

Every regular language has a right linear grammar. Every right linear grammar is context-free. □

Theorem

There is a context-free language which is not regular.

Proof.

$L = \{a^n b^n \mid n \geq 1\}$ is not regular (pumping lemma), but $S \to ab \mid aSb$ is a context-free grammar for L. □

Outline

- ✓ Deterministic finite automata
- ✓ Nondeterministic finite automata
- ✓ Automata with ε-transitions
- ✓ The class of regular languages
- ✓ The pumping lemma for regular languages
- ✓ Context-free grammars and languages
- ✓ Right linear grammars
- $\Rightarrow$ Pushdown Automata
- ▸ The pumping lemma for context-free languages
- ▸ Grammars in computer science
- ▸ Further topics

Automata for Context-Free Languages ?

- Well-balanced strings of parentheses, e.g.
  $(), (()()), ((()())()) \in W$ but
  $((),)(() \notin W$
- Context-free grammar for W: $E \rightarrow EE \mid (E) \mid \varepsilon$
- W is not regular (by pumping lemma, BB14)
- Generating a language vs. accepting a language
- How to accept a well-balanced string? (BB15)

Automata for Context-Free Languages ?

- Well-balanced strings of parentheses, e.g.
  $(), (()()), ((()())()) \in W$ but
  $((),)(() \notin W$
- Context-free grammar for W: $E \to EE \mid (E) \mid \varepsilon$
- W is not regular (by pumping lemma, BB14)
- Generating a language vs. accepting a language
- How to accept a well-balanced string? (BB15)
  $\Longrightarrow$ Need more than constant memory
  $\Longrightarrow$ For context-free languages: automaton with a stack

Pushdown Automata – Definition

Definition

A *pushdown automaton* (PDA) is a tuple $A = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F\rangle$ where:

1. Q is a finite set (the *states*).
2. Σ is a finite set (the *input symbols*).
3. Γ is a finite set (the *stack symbols*).
4. $\Delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$ (the *transition relation*). where for each $(q, x, z) \in Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma$ there are only finitely many $(q', w) \in Q \times \Gamma^*$ s.t. $(q, x, z, q', w) \in \Delta$.
5. $q_0 \in Q$ (the *starting state*).
6. $Z_0 \in \Gamma$ (the *starting stack symbol*).
7. $F \subseteq Q$ (the *final states*).

Computation of a PDA

Let $A = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ be a PDA.

Definition

A *configuration* of A is a triple $(q, w, v) \in Q \times \Sigma^* \times \Gamma^*$ where

- q is the *current state*,
- w is the *remaining input*, and
- v is the *current stack contents*.

Convention: top of the stack is on the left

Definition

Define the binary *step relation* $\vdash_A$ on configurations of A as:

$$(q, xw, yv) \vdash_A (p, w, uv) \;\text{ if } (q, x, y, p, u) \in \Delta$$

Define $\vdash_A^*$ as reflexive and transitive closure of $\vdash_A$.

The Language of a PDA

Definition

Let $A = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ be a PDA, then the *language accepted by A by* **final state** is defined as:

$$\mathrm{L}(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_A^* (q, \varepsilon, v) \text{ for some } q \in F \text{ and any } v\}$$

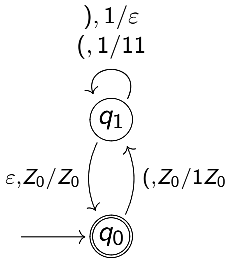
Definition

Let $A = \langle Q, \Sigma, \Gamma, \delta, q_0, Z_0, F \rangle$ be a PDA, then the *language accepted by A by* **empty stack** is defined as:

$$\mathrm{N}(A) = \{w \in \Sigma^* \mid (q_0, w, Z_0) \vdash_A^* (q, \varepsilon, \varepsilon) \text{ for any } q\}$$

Example

$\Sigma = \{(,)\}, \Gamma = \{Z_0, 1\}$



Derivation of (()()): BB16

Theorem

Let $L \subseteq \Sigma^*$, then the following are equivalent:

1. *There is a PDA A_N with $\mathrm{N}(A_N) = L$.*
2. *There is a PDA A_F with $\mathrm{L}(A_F) = L$.*

Proof (BB17).

$1 \Rightarrow 2$

$2 \Rightarrow 1$ $\square$

Theorem

A language L has a context-free grammar iff it has a PDA.

Proof (BB18).

1. Given grammar G, construct PDA A.
2. Given PDA A, construct grammar G.

$\square$

Formalisms

	Automata	Grammars
Regular Languages	DFA, NFA	LLG, RLG
Context-Free Languages	PDA	CFG

Outline

- ✓ Deterministic finite automata
- ✓ Nondeterministic finite automata
- ✓ Automata with ε-transitions
- ✓ The class of regular languages
- ✓ The pumping lemma for regular languages
- ✓ Context-free grammars and languages
- ✓ Right linear grammars
- ✓ Pushdown Automata
- $\Rightarrow$ The pumping lemma for context-free languages
- ▶ Grammars in computer science
- ▶ Further topics

Lemma (Pumping Lemma)

Let L be a context-free language. Then there is an $n \in \mathbb{N}$ s.t. for every $w \in L$ with $|w| \geq n$ we have $w = v_1 v_2 v_3 v_4 v_5$ with

1. $|v_2 v_3 v_4| \leq n$,
2. $v_2 v_4 \neq \varepsilon$, *and*
3. *for all* $k \geq 0$ *also* $v_1 v_2^k v_3 v_4^k v_5 \in L$.

Proof Sketch (BB19).

$\square$

Lemma (Pumping Lemma)

Let L be a context-free language. Then there is an $n \in \mathbb{N}$ s.t. for every $w \in L$ with $|w| \geq n$ we have $w = v_1 v_2 v_3 v_4 v_5$ with

1. $|v_2 v_3 v_4| \leq n$,
2. $v_2 v_4 \neq \varepsilon$, *and*
3. *for all* $k \geq 0$ *also* $v_1 v_2^k v_3 v_4^k v_5 \in L$.

Example

$L = \{a^m b^m c^m \mid m \geq 1\}$ is not context-free (BB20).

Outline

- ✓ Deterministic finite automata
- ✓ Nondeterministic finite automata
- ✓ Automata with ε-transitions
- ✓ The class of regular languages
- ✓ The pumping lemma for regular languages
- ✓ Context-free grammars and languages
- ✓ Right linear grammars
- ✓ Pushdown Automata
- ✓ The pumping lemma for context-free languages
- ⇒ Grammars in computer science
- ▶ Further topics

HTML is a Context-free Language

Source of website

$$
\begin{aligned}
\text{Char} &\to \textbf{a} \mid \textbf{A} \mid \textbf{b} \mid \textbf{B} \mid \cdots \\
\text{String} &\to \varepsilon \mid \text{Char String} \\
\text{Element} &\to \text{Heading} \mid \text{Paragraph} \mid \text{Link} \mid \text{String} \mid <\textbf{br}> \mid \cdots \\
\text{Elements} &\to \varepsilon \mid \text{Element Elements} \\
\text{Heading} &\to <\textbf{h3}> \text{ String } </\textbf{h3}> \mid \cdots \\
\text{Paragraph} &\to <\textbf{p}> \text{ Elements } </\textbf{p}> \\
\text{Link} &\to <\textbf{a href} = \text{“String“}> \text{ String } </\textbf{a}> \\
&\vdots
\end{aligned}
$$

XML

Generalization: XML (Extensible Markup Language)

- DTD (Document Type Definition) is a grammar
- There are DTDs for:
  HTML, office formats, mathematical formulas, address data, vector graphics, cooking recipes, formal proofs, ...
- Very rich infrastructure available

Further Topics

- Context-sensitive languages: $uAv \to uwv$
- Regular expressions
- Decidability/complexity of, e.g., membership, emptyness, ...
- Parser generators
- ...

Introductory Textbook:
J. E. Hopcroft, R. Motwani, J. D. Ullman: *Introduction to Automata Theory, Languages, and Computation*, 2nd edition, 2001, Addison-Wesley.