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 $\varepsilon$-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
The language accepted by this automaton is $L = \{a^k b^n c^m | k, n, m \geq 1\}$.
Finite Automata – A More Abstract Example

The language accepted by this automaton is

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

\textit{abbbcc}
Finite Automata – A More Abstract Example

The language accepted by this automaton is

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

$abbbcc$
The language accepted by this automaton is \( L = \{ a^k b^n c^m \mid k, n, m \geq 1 \} \)
Finite Automata – A More Abstract Example

The language accepted by this automaton is $L = \{a^k b^n c^m | k, n, m \geq 1\}$.

$abbbcc$
The language accepted by this automaton is \( L = \{ a^k b^n c^m \mid k, n, m \geq 1 \} \)

\( aabbcc \)
Finite Automata – A More Abstract Example

The language accepted by this automaton is $L = \{a^k b^n c^m \mid k, n, m \geq 1\}$.

$abbbcc$
The language accepted by this automaton is $L = \{a^k b^n c^m \mid k, n, m \geq 1\}$.
Finite Automata – A More Abstract Example

The language accepted by this automaton is $L = \{a^k b^n c^m | k, n, m \geq 1\}$.

$abbbcc$
Finite Automata – A More Abstract Example

The language accepted by this automaton is

\[ L = \{ a^k b^n c^m | k, n, m \geq 1 \} \]

\[ abbbcc \quad \checkmark \]
The language accepted by this automaton is \( L = \{ a^k b^n c^m \mid k, n, m \geq 1 \} \).

\[ aabbc \quad \checkmark \]

\[ ab 

\[ aabbc \quad \checkmark \]

\[ aab \]
The language accepted by this automaton is $L = \{a^k b^n c^m | k, n, m \geq 1\}$.

The strings $abbbcc$ and $aab$ are accepted.
Finite Automata – A More Abstract Example

The language accepted by this automaton is \( L = \{ a^k b^n c^m \mid k, n, m \geq 1 \} \)

\( a b b b c c \quad \checkmark \)

\( a a b \)
Finite Automata – A More Abstract Example

The language accepted by this automaton is $L = \{a^k b^n c^m | k, n, m \geq 1\}$

$abbbcc \checkmark$

$aab$
Finite Automata – A More Abstract Example

The language accepted by this automaton is

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

- \textbf{abbbcc}  \checkmark
- \textbf{aab}
Finite Automata – A More Abstract Example

The language accepted by this automaton is

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

- $abbbcc$ is accepted: ✓
- $aab$ is not accepted: ×
The language accepted by this automaton is:

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

The strings that are accepted by the automaton are:

- `abbbcc`
- `aab`
- `ac`
Finite Automata – A More Abstract Example

The language accepted by this automaton is

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

$abbbcc \quad \checkmark$

$aab \quad \times$

$ac$
Finite Automata – A More Abstract Example

The language accepted by this automaton is $L = \{a_k b_n c_m | k, n, m \geq 1\}$.

- $abbbcc$ \(\checkmark\)
- $aab$ \(\times\)
- $ac$
The language accepted by this automaton is $L = \{ a^k b^n c^m | k, n, m \geq 1 \}$.

- The string $abbbcc$ is accepted. (✓)
- The string $aab$ is not accepted. (✗)
- The string $ac$ is not accepted. (✗)
The language accepted by this automaton is

\[ L = \{a^k b^n c^m \mid k, n, m \geq 1\} \]
The Error State

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{c} q_3 \]
The Error State
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. \( \Sigma \) is a finite set (the input symbols).
3. \( \delta : Q \times \Sigma \rightarrow Q \) (the transition function).
4. \( q_0 \in Q \) (the starting state)
5. \( F \subseteq Q \) (the final states).
as tuple: BB1
Definition
Extend $\delta : Q \times \Sigma \rightarrow Q$ to $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ 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 $\delta : Q \times \Sigma \rightarrow Q$ to $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ 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
Extend $\delta : Q \times \Sigma \rightarrow Q$ to $\hat{\delta} : Q \times \Sigma^* \rightarrow Q$ 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

$L(A) = \{ w \in \Sigma^* \mid \hat{\delta}(q_0, w) \in F \}$. 
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
✓ Deterministic finite automata
⇒ Nondeterministic finite automata
  ▶ Automata with $\varepsilon$-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
Automaton for accepting $L = \{wab \mid w \in \{a, b\}^*\}$ ?
Automaton for accepting \( L = \{wab \mid w \in \{a, b\}^*\} \)?
Automaton for accepting $L = \{wab \mid w \in \{a, b\}^*\}$?

Nondeterminism $\Rightarrow$ 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. $\Sigma$ 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).
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.

$(q, w, q) \in \hat{\Delta}$ if $w = \varepsilon$

$(q, w, q^*) \in \hat{\Delta}$ if $w = vx$ for $v \in \Sigma^*$, $x \in \Sigma$,
and $(q, v, q') \in \hat{\Delta}$, and $(q', x, q) \in \Delta$
The Language of an NFA

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

$$(q, w, q) \in \hat{\Delta} \text{ if } w = \varepsilon$$

$$(q, w, q^*) \in \hat{\Delta} \text{ if } w = vx \text{ for } v \in \Sigma^*, x \in \Sigma,$$

and $$(q, v, q') \in \hat{\Delta}, \text{ and } (q', x, q) \in \Delta$$

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.

$$(q, w, q) \in \hat{\Delta} \quad \text{if} \quad w = \varepsilon$$

$$(q, w, q^*) \in \hat{\Delta} \quad \text{if} \quad w = vx \quad \text{for} \quad v \in \Sigma^*, x \in \Sigma,$$

and $$(q, v, q') \in \hat{\Delta}, \text{ and } (q', x, q) \in \Delta$$

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

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

$$L(A) = \{ w \in \Sigma^* \mid \exists q \in F \text{ s.t. } (q_0, w, q) \in \hat{\Delta} \}. $$
Theorem
Let $L \subseteq \Sigma^*$, then there is a DFA $D$ with $L(D) = L$ iff there is a NFA $N$ with $L(N) = L$.

Proof (BB4).

1. Converting $D$ to $N$: easy.
2. Converting $N$ to $D$: subset construction.
Automaton for accepting $L = \{wab \mid w \in \{a, b\}^*\}$:

![Automaton Diagram]

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 $L(N_n) = L_n$ s.t. all DFA $D_n$ with $L(D_n) = L_n$ have at least $2^n$ states.

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

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

✓ Deterministic finite automata
✓ Nondeterministic finite automata
⇒ Automata with $\varepsilon$-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
Automaton for accepting decimal representations of integers:

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

\[ L = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+ \]
\[ \cup \{-w \mid w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+\} \]
Automaton for accepting decimal representations of integers:

\[ \Sigma = \{ -, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\} \]
\[ L = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+ \]
\[ \cup \{-w \mid w \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}^+\} \]
Automaton for accepting decimal representations of integers:

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

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

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

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

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

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

**Proof.**
Modified subset construction ($\varepsilon$-closed subsets).
✓ Deterministic finite automata
✓ Nondeterministic finite automata
✓ Automata with $\varepsilon$-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
Corollary

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

- There is a DFA $D$ with $L(D) = L$.
- There is a NFA $N$ with $L(N) = L$.
- There is a $\varepsilon$-NFA $N'$ with $L(N') = L$.

Definition

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

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

Proof.

BB7
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

**Theorem**

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

**Proof.**

BB8
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

Theorem
If \( L \subseteq \Sigma^* \) is regular, then \( L^c = \Sigma^* \setminus L \) is regular.

Proof.
BB8

Theorem
If \( L_1, L_2 \subseteq \Sigma^* \) are regular, then \( L_1 \cap L_2 \) is regular.

Proof.
\[ L_1 \cap L_2 = (L_1^c \cup L_2^c)^c. \]
✓ 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
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
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 $w = v_1v_2v_3$ with

1. $v_2 \neq \epsilon,$
2. $|v_1v_2| \leq n,$ and
3. for all $k \geq 0$ also $v_1v_2^kv_3 \in L.$

Example

$L = \{a^mb^m \mid m \geq 1\}$ is not regular (BB10).
✓ Deterministic finite automata
✓ Nondeterministic finite automata
✓ Automata with \( \varepsilon \)-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
How can we specify the set of all arithmetical expressions?
E.g. $12$, $30 + 21 \cdot 6$, $(123 + 7) \cdot 15 + 88$, ...

\[
\begin{align*}
E & \rightarrow N \mid E + E \mid E \cdot E \mid (E) \\
N & \rightarrow D \mid DN \\
D & \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\end{align*}
\]
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).
Context-Free Grammars – Example

\[ G = \langle NT, T, P, S \rangle \]
\[ NT = \{ N, D, E \} \]
\[ T = \{ +, \cdot, (, ), 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \} \]
\[ P = \{ E \rightarrow N, E \rightarrow E + E, E \rightarrow E \cdot E, E \rightarrow (E), N \rightarrow D, N \rightarrow DN, D \rightarrow 0, D \rightarrow 1, D \rightarrow 2, D \rightarrow 3, D \rightarrow 4, D \rightarrow 5, D \rightarrow 6, D \rightarrow 7, D \rightarrow 8, D \rightarrow 9 \} \]
\[ S = E \]
Let $G = \langle N, T, P, S \rangle$ be a CFG.

**Definition**
For every $A \rightarrow w \in P$ 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 $L(G) = \{ w \in T^* \mid S \Rightarrow^*_G w \}$.

**Definition**
$L \subseteq \Sigma^*$ is called *context-free* if there is a context-free grammar $G$ with $L(G) = L$. 
Context-Free Grammars – Example Derivation

\[
E \rightarrow N \mid E + E \mid E \cdot E \mid (E)
\]

\[
N \rightarrow D \mid DN
\]

\[
D \rightarrow 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9
\]

Example derivation: BB11
## Formalisms

<table>
<thead>
<tr>
<th></th>
<th>Automata</th>
<th>Grammars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Languages</td>
<td>DFAs, NFAs</td>
<td>?</td>
</tr>
<tr>
<td>Context-Free Languages</td>
<td>?</td>
<td>CFG</td>
</tr>
</tbody>
</table>
✓ Deterministic finite automata
✓ Nondeterministic finite automata
✓ Automata with $\varepsilon$-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
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:

\[
A \rightarrow xB \text{ where } x \in T, B \in N \\
A \rightarrow x \text{ where } x \in T \\
A \rightarrow \varepsilon
\]

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

Proof (BB12).

1. From right linear grammar to \( \varepsilon \)-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

<table>
<thead>
<tr>
<th></th>
<th>Automata</th>
<th>Grammars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Languages</td>
<td>DFA, NFA</td>
<td>LLG, RLG</td>
</tr>
<tr>
<td>Context-Free Languages</td>
<td>?</td>
<td>CFG</td>
</tr>
</tbody>
</table>
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.

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


✓ Deterministic finite automata
✓ Nondeterministic finite automata
✓ Automata with $\varepsilon$-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
Well-balanced strings of parentheses, e.g. 

\((\), (())\), (((())())()) \in W \text{ but } 
\(((), )(() \notin W \\
\text{Context-free grammar for } W: E \rightarrow EE \mid (E) \mid \varepsilon \\
W \text{ is not regular (by pumping lemma, BB14)} \\
\text{Generating a language vs. accepting a language} \\
\text{How to accept a well-balanced string? (BB15)}
Automata for Context-Free Languages?

- Well-balanced strings of parentheses, e.g.
  
  \((\)), (((())), (((())))()()) \in W \text{ 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)
  
  \(\Rightarrow\) Need more than constant memory

  \(\Rightarrow\) For context-free languages: automaton with a stack
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. \( \Sigma \) is a finite set (the input symbols).
3. \( \Gamma \) 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).
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 $\vdash_A$. 
### 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:

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

### 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:

$$N(A) = \{ w \in \Sigma^* | (q_0, w, Z_0) \vdash_A^* (q, \varepsilon, \varepsilon) \text{ for any } q \}$$
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 $N(A_N) = L$.
2. There is a PDA $A_F$ with $L(A_F) = L$.

Proof (BB17).

$1 \Rightarrow 2$

$2 \Rightarrow 1$
Theorem

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

Proof (BB18).

2. Given PDA $A$, construct grammar $G$. 
<table>
<thead>
<tr>
<th>Languages</th>
<th>Automata</th>
<th>Grammars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Languages</td>
<td>DFA, NFA</td>
<td>LLG, RLG</td>
</tr>
<tr>
<td>Context-Free Languages</td>
<td>PDA</td>
<td>CFG</td>
</tr>
</tbody>
</table>
✓ Deterministic finite automata
✓ Nondeterministic finite automata
✓ Automata with $\varepsilon$-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 Pumping Lemma for Context-Free Languages

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).
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).
✓ Deterministic finite automata
✓ Nondeterministic finite automata
✓ Automata with $\varepsilon$-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

\[
\begin{align*}
\text{Char} & \to a \mid A \mid b \mid B \mid \cdots \\
\text{String} & \to \varepsilon \mid \text{Char String} \\
\text{Element} & \to \text{Heading} \mid \text{Paragraph} \mid \text{Link} \mid \text{String} \mid <\text{br}> \mid \cdots \\
\text{Elements} & \to \varepsilon \mid \text{Element Elements} \\
\text{Heading} & \to <\text{h3}> \text{String} </\text{h3}> \mid \cdots \\
\text{Paragraph} & \to <\text{p}> \text{Elements} </\text{p}> \\
\text{Link} & \to <\text{a href} = "\text{String}"> \text{String} </\text{a}>
\end{align*}
\]
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: $uAv \rightarrow uvw$
* Regular expressions
* Decidability/complexity of, e.g., membership, emptiness, ...
* Parser generators
* ...

Introductory Textbook: