Automata Theory and Formal Languages

Summer Term 2026

9th Lecture

Context-Free Languages

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

Context-Free Languages

Context-Free Languages

Recall:

Type 2 (context free) languages

A grammar is said to be context free if for each rule $w_1 \rightarrow w_2$ the string w_1 contains only a single variable (i.e. $w_1 \in V$). A language L is context free if a grammar G exists with $L(G) = L$. We know how to prove that a language is regular.

Context-Free Languages

Recall:

Type 2 (context free) languages

A grammar is said to be context free if for each rule $w_1 \rightarrow w_2$ the string w_1 contains only a single variable (i.e. $w_1 \in V$). A language L is context free if a grammar G exists with $L(G) = L$. We know how to prove that a language is regular.

- The context-free (Chomsky type 2) languages are very relevant for the definition and handling of programming languages.

Context-Free Languages

Recall:

Type 2 (context free) languages

A grammar is said to be context free if for each rule $w_1 \rightarrow w_2$ the string w_1 contains only a single variable (i.e. $w_1 \in V$). A language L is context free if a grammar G exists with $L(G) = L$. We know how to prove that a language is regular.

- The context-free (Chomsky type 2) languages are very relevant for the definition and handling of programming languages.
- They extend the regular languages by the possibility of dynamically *storing* information during the construction (grammar) or the acceptance (automaton) of a string. In this way we can store the n in an expression like $a^n b^n$.

Context-Free Languages

Recall:

Type 2 (context free) languages

A grammar is said to be context free if for each rule $w_1 \to w_2$ the string w_1 contains only a single variable (i.e. $w_1 \in V$). A language L is context free if a grammar G exists with $L(G) = L$. We know how to prove that a language is regular.

- The context-free (Chomsky type 2) languages are very relevant for the definition and handling of programming languages.
- They extend the regular languages by the possibility of dynamically *storing* information during the construction (grammar) or the acceptance (automaton) of a string. In this way we can store the n in an expression like $a^n b^n$.
- Context-free languages are so important since they allow expressing *bracket expressions*.

Example

The non-regular language

$$L = \{a^n b^n \mid n \geq 1\}$$

is context free

Example

The non-regular language

$$L = \{a^n b^n \mid n \geq 1\}$$

is context free since the grammar $G = (\{S\}, \{a, b\}, P, S)$ with productions

$$S \to ab \mid aSb$$

generates the language ($L(G) = L$).

Normal Forms

In context-free grammars, the right hand side of a rule may contain an arbitrary number of letters. For an easier theoretical handling of context-free grammars we introduce *normal forms*. A normal form does not restrict the language type and thus has to fulfill that any context-free grammar can be converted to the respective normal form.

Normal Forms

In context-free grammars, the right hand side of a rule may contain an arbitrary number of letters. For an easier theoretical handling of context-free grammars we introduce *normal forms*. A normal form does not restrict the language type and thus has to fulfill that any context-free grammar can be converted to the respective normal form.

Two important normal forms are

- Chomsky normal form
- Greibach normal form

We put our main focus on the first one.

Chomsky Normal Form

Definition

A context-free grammar $G = (V, \Sigma, P, S)$ is in *Chomsky normal form* (CNF) if all productions in P have the form

$$A \to BC \quad \text{or}$$
$$A \to a \quad \text{or}$$
$$S \to \varepsilon$$

where $A, B, C \in V$ and $a \in \Sigma$.

Algorithm for Conversion into CNF

Input: General context-free grammar $G = (V, \Sigma, P, S)$

Algorithm for Conversion into CNF

Input: General context-free grammar $G = (V, \Sigma, P, S)$

1. If $S \to \varepsilon$ is a production in P then we introduce a new start variable S' and add the rules

$$S' \to S \quad \text{and}$$
$$S' \to \varepsilon$$

Algorithm for Conversion into CNF

Input: General context-free grammar $G = (V, \Sigma, P, S)$

1. If $S \to \varepsilon$ is a production in P then we introduce a new start variable S' and add the rules

$$S' \to S \quad \text{and}$$
$$S' \to \varepsilon$$

2. For each terminal symbol $a \in \Sigma$ we introduce a new variable X_a. Each a in the productions is replaced by X_a and rules $X_a \to a$ are added. Now only variables on right hand side!

Algorithm for Conversion into CNF

Input: General context-free grammar $G = (V, \Sigma, P, S)$

1. If $S \to \varepsilon$ is a production in P then we introduce a new start variable S' and add the rules

$$S' \to S \quad \text{and}$$
$$S' \to \varepsilon$$

2. For each terminal symbol $a \in \Sigma$ we introduce a new variable X_a. Each a in the productions is replaced by X_a and rules $X_a \to a$ are added. Now only variables on right hand side!

3. For all productions having more than two variables on the right hand side (e.g. ABC) we introduce a new variable Y_{AB} and replace all AB with Y_{AB}. We then add the rule $Y_{AB} \to AB$. This step is iteratively done until no more than two variable are on the right hand side of the productions.

Algorithm for Conversion into CNF

4. Remove all rules $A \to \varepsilon$ except the rule $S' \to \varepsilon$ (if it exists). If $A \to \varepsilon$ is the only rule with A on the left hand side of the production we can remove all occurrences of A on the right hand side of the productions. Otherwise we introduce for any rule $C \to AB$ or $C \to BA$ the rule

Algorithm for Conversion into CNF

4. Remove all rules $A \to \varepsilon$ except the rule $S' \to \varepsilon$ (if it exists). If $A \to \varepsilon$ is the only rule with A on the left hand side of the production we can remove all occurrences of A on the right hand side of the productions. Otherwise we introduce for any rule $C \to AB$ or $C \to BA$ the rule

5. Remove all chain rules $(A \to B)$ and introduce for any $B \to w$, where w is a right hand side (one or two letters), the rule $A \to w$. Do this until no chain rules exist anymore.

Example

Let $G = (V, \Sigma, P, S)$ be a grammar over $\Sigma = \{a, b\}$ with rules

$$S \to ASA \mid aB$$
$$A \to B \mid S$$
$$B \to b \mid \varepsilon$$

We will now convert this grammar step by step until it is in Chomsky normal form.

Example

1. Nothing to do

Example

1. Nothing to do
2. Introduce new variables for terminals

$$S \to ASA \mid X_a B$$
$$A \to B \mid S$$
$$B \to X_b \mid \varepsilon$$
$$X_a \to a$$
$$X_b \to b$$

Example

3. Introduce new variable for AS

$$
\begin{aligned}
S &\to Y_{AS}A \mid X_aB \\
A &\to B \mid S \\
B &\to X_b \mid \varepsilon \\
X_a &\to a \\
X_b &\to b \\
Y_{AS} &\to AS
\end{aligned}
$$

Example

4. Remove ε, in two steps:

$$
\begin{aligned}
S &\to Y_{AS}A \mid X_aB \mid X_a \\
A &\to B \mid S \mid \varepsilon \\
B &\to X_b \\
X_a &\to a \\
X_b &\to b \\
Y_{AS} &\to AS
\end{aligned}
$$

Example

4. Remove ε, in two steps:

$$
\begin{aligned}
S &\to Y_{AS}A \mid X_aB \mid X_a \\
A &\to B \mid S \mid \varepsilon \\
B &\to X_b \\
X_a &\to a \\
X_b &\to b \\
Y_{AS} &\to AS
\end{aligned}
$$

$$
\begin{aligned}
S &\to Y_{AS}A \mid Y_{AS} \mid X_aB \mid X_a \\
A &\to B \mid S \\
B &\to X_b \\
X_a &\to a \\
X_b &\to b \\
Y_{AS} &\to AS
\end{aligned}
$$

Example

4. Remove ε, in two steps:

$$
\begin{aligned}
S &\to Y_{AS}A \mid X_aB \mid X_a \\
A &\to B \mid S \mid \varepsilon \\
B &\to X_b \\
X_a &\to a \\
X_b &\to b \\
Y_{AS} &\to AS
\end{aligned}
\qquad
\begin{aligned}
S &\to Y_{AS}A \mid Y_{AS} \mid X_aB \mid X_a \\
A &\to B \mid S \\
B &\to X_b \\
X_a &\to a \\
X_b &\to b \\
Y_{AS} &\to AS
\end{aligned}
$$

Note: S has not been added to last rule to avoid a cycle.

Example

5. Remove chain rules:

$$
\begin{aligned}
S &\to Y_{AS}A \mid AS \mid X_aB \mid a \\
A &\to B \mid S \\
B &\to b \\
X_a &\to a \\
Y_{AS} &\to AS
\end{aligned}
$$

Example

5. Remove chain rules:

$$
\begin{aligned}
S &\to Y_{AS}A \mid AS \mid X_aB \mid a \\
A &\to B \mid S \\
B &\to b \\
X_a &\to a \\
Y_{AS} &\to AS
\end{aligned}
$$

$$
\begin{aligned}
S &\to Y_{AS}A \mid AS \mid X_aB \mid a \\
A &\to b \mid Y_{AS}A \mid AS \mid X_aB \mid a \\
B &\to b \\
X_a &\to a \\
Y_{AS} &\to AS
\end{aligned}
$$

Greibach Normal Form

Definition

A context-free grammar $G = (V, \Sigma, P, S)$ with $\varepsilon \notin L(G)$ is in *Greibach normal form* (GNF) if all productions in P have the form

$$A \to aB_1 B_2 \dots B_k \quad k \geq 0$$

where $A, B_1, \dots, B_k \in V$ and $a \in \Sigma$.

Greibach Normal Form

Remark

The Greibach normal form is the natural extension of regular grammars since the later is a special case of the GNF where only $k = 0$ and $k = 1$ are allowed.

Greibach Normal Form

Remark

The Greibach normal form is the natural extension of regular grammars since the later is a special case of the GNF where only $k = 0$ and $k = 1$ are allowed.

Any context-free grammar that does not generate the empty string can be converted into GNF. An algorithm can be found in the book of *Schöning*.