Automata Theory and Formal Languages

Summer Term 2026

4th Lecture

Chomsky Hierarchy

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

Chomsky Hierarchy

Containment hierarchy of classes of formal grammars. Grammars are classified into four different types with different limitations

Chomsky Hierarchy

Containment hierarchy of classes of formal grammars. Grammars are classified into four different types with different limitations

Type 0

no restrictions

Chomsky Hierarchy

Containment hierarchy of classes of formal grammars. Grammars are classified into four different types with different limitations

Type 0

no restrictions

Type 1

Each rule $w_1 \rightarrow w_2$ satisfies $|w_1| \leq |w_2|$ with the exception that $S \rightarrow \varepsilon$ is allowed if S does not occur on any right hand side of the rules.
$|w_1| \leq |w_2|$ implies that no shortening rules are allowed.

Chomsky Hierarchy

Containment hierarchy of classes of formal grammars. Grammars are classified into four different types with different limitations

Chomsky Hierarchy

Containment hierarchy of classes of formal grammars. Grammars are classified into four different types with different limitations

Type 2 (context free)

Same restriction as type 1 and additionally for each rule $w_1 \rightarrow w_2$ the string w_1 contains only a single variable (i.e. $w_1 \in V$)

Chomsky Hierarchy

Containment hierarchy of classes of formal grammars. Grammars are classified into four different types with different limitations

Type 2 (context free)

Same restriction as type 1 and additionally for each rule $w_1 \rightarrow w_2$ the string w_1 contains only a single variable (i.e. $w_1 \in V$)

Type 3 (regular)

Same restriction as type 2 and additionally $w_2 \in \Sigma \cup \Sigma V$, i.e. the right hand side is either a single terminal symbol or a terminal symbol followed by a variable.

Chomsky Hierarchy

Remark

The order of the variable and terminal symbol is chosen arbitrarily. Important is just that within a regular grammar only a single order occurs. If for all rules $w_2 \in \Sigma \cup V\Sigma$ it would also be a regular grammar.

Chomsky Hierarchy

Definition

A language $L \subset \Sigma^*$ is said to be of type 0 (1,2,3) if there exist a type 0 (1,2,3) grammar G for which $L(G) = L$.

Chomsky Hierarchy

Definition

A language $L \subset \Sigma^*$ is said to be of type 0 (1,2,3) if there exist a type 0 (1,2,3) grammar G for which $L(G) = L$.

Remark

In order to show that a language L is of type 0 (1,2,3) one thus just has to find a corresponding grammar.

Chomsky Hierarchy

Definition

A language $L \subset \Sigma^*$ is said to be of type 0 (1,2,3) if there exist a type 0 (1,2,3) grammar G for which $L(G) = L$.

Remark

In order to show that a language L is of type 0 (1,2,3) one thus just has to find a corresponding grammar. In order to show that a grammar is not of type 0 (1,2,3) one has to prove that no type 0 (1,2,3) grammar can exist generating the corresponding language.

Chomsky Hierarchy

Remark

- For a *context-free rule* $A \to x$ one can replace A independently of the context by x. The *context* are the neighboring characters / substrings.

Chomsky Hierarchy

Remark

- For a *context-free rule* $A \to x$ one can replace A independently of the context by x. The *context* are the neighboring characters / substrings.
- For *context-sensitive languages* rules of the form

$$uAv \to uxv$$

are allowed, i.e. A can only be replaced by x if it occurs between a u and a v.

Chomsky Hierarchy

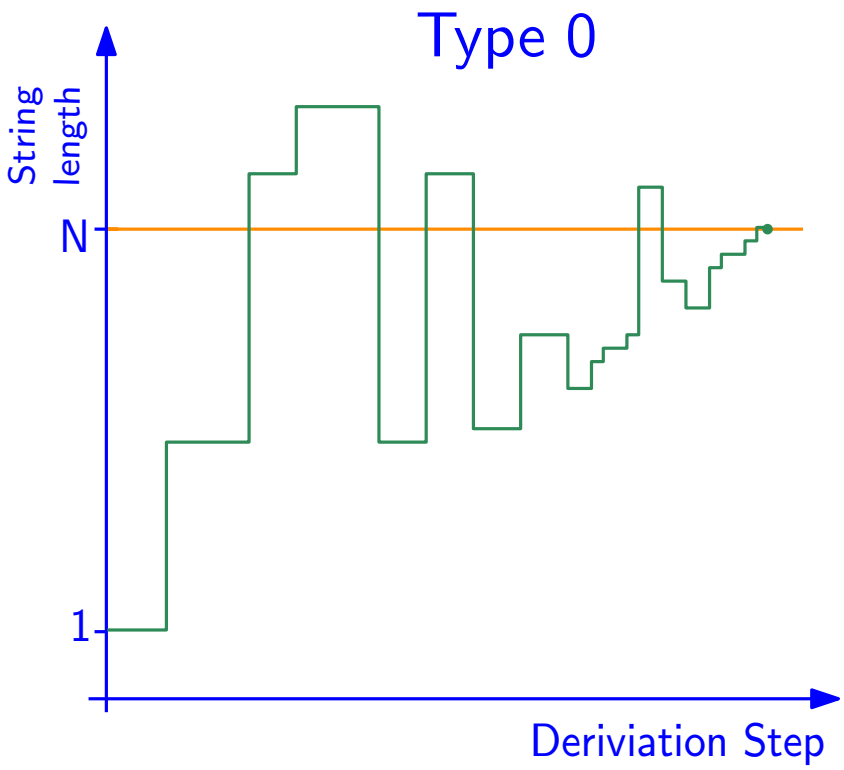
Difference between type 0 and type 1 languages:

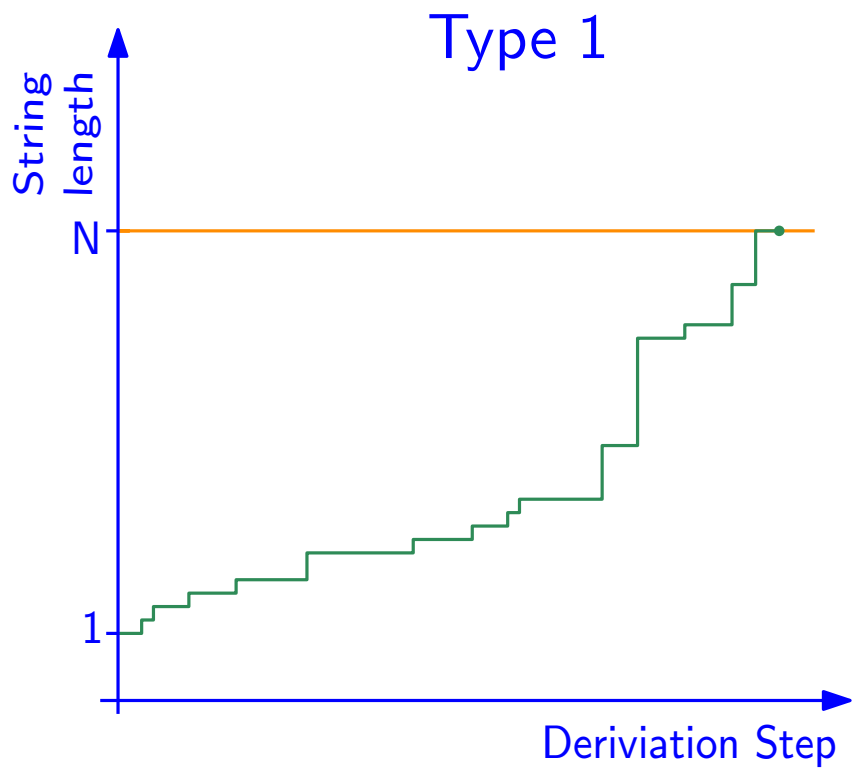
For type 0 languages the string length can decrease during the derivation process.

Chomsky Hierarchy

Difference between type 0 and type 1 languages:

For type 0 languages the string length can decrease during the derivation process.





Special ε rule

In some text books the special ε rule $S \to \varepsilon$ is not included for type 1 grammars. In that case the languages cannot generate the empty string. We include it directly in the Chomsky hierarchy to avoid this.

Special ε rule

In some text books the special ε rule $S \to \varepsilon$ is not included for type 1 grammars. In that case the languages cannot generate the empty string. We include it directly in the Chomsky hierarchy to avoid this.

For type 2/3 grammar one can additionally allow rules of the form $A \to \varepsilon$ where A is any variable. This does not change the power of the language class.

Containment hierarchy of Chomsky Hierarchy

type 3 languages $\subsetneq$ type 2 languages $\subsetneq$ type 1 languages $\subsetneq$ type 0 languages

Containment hierarchy of Chomsky Hierarchy

type 3 languages $\subsetneq$ type 2 languages $\subsetneq$ type 1 languages $\subsetneq$ type 0 languages

Difference sets

- type 2 but not type 3: $L = \{a^n b^n \mid n \geq 0\}$

Containment hierarchy of Chomsky Hierarchy

type 3 languages $\subsetneq$ type 2 languages $\subsetneq$ type 1 languages $\subsetneq$ type 0 languages

Difference sets

- type 2 but not type 3: $L = \{a^n b^n \mid n \geq 0\}$
- type 1 but not type 2: $L = \{a^n b^n c^n \mid n \geq 0\}$

Containment hierarchy of Chomsky Hierarchy

type 3 languages $\subsetneq$ type 2 languages $\subsetneq$ type 1 languages $\subsetneq$ type 0 languages

Difference sets

- type 2 but not type 3: $L = \{a^n b^n \mid n \geq 0\}$
- type 1 but not type 2: $L = \{a^n b^n c^n \mid n \geq 0\}$
- type 0 but not type 1:
  $L = \{w \mid w \text{ describes a terminating Turing machine}\}$

Decision Problem

Decision Problem

given: String x of length n and grammar G

Decision Problem

Decision Problem

given: String x of length n and grammar G

Problem: Is $x \in L(G)$?

Decision Problem

Decision Problem

given: String x of length n and grammar G

Problem: Is $x \in L(G)$?

Theorem

The decision problem for type 1 (2,3) languages is decidable (computable).

Decision Problem

Proof

If $x \in L(G)$ there exist a derivation

$$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k = x.$$

We consider the shortest existing derivation (there may be multiple).
Since G is of type 1 the string length does never decrease.

Decision Problem

Proof

If $x \in L(G)$ there exist a derivation

$$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k = x.$$

We consider the shortest existing derivation (there may be multiple). Since G is of type 1 the string length does never decrease.

Since the derivation is assumed to be the shortest it cannot contain any cycles. Since the number of strings of length $\leq n$ is finite the derivation contains a maximum of $\sum_{i=1}^{n} |\Sigma \cup V|^i$ steps.

Decision Problem

Proof

If $x \in L(G)$ there exist a derivation

$$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k = x.$$

We consider the shortest existing derivation (there may be multiple). Since G is of type 1 the string length does never decrease.

Since the derivation is assumed to be the shortest it cannot contain any cycles. Since the number of strings of length $\leq n$ is finite the derivation contains a maximum of $\sum_{i=1}^{n} |\Sigma \cup V|^i$ steps.

By systematically trying out all derivation branches an algorithm can determine if the grammar G can generate the string x.

Decision Problem

Proof

If $x \in L(G)$ there exist a derivation

$$S \Rightarrow x_1 \Rightarrow x_2 \Rightarrow \cdots \Rightarrow x_k = x.$$

We consider the shortest existing derivation (there may be multiple). Since G is of type 1 the string length does never decrease.

Since the derivation is assumed to be the shortest it cannot contain any cycles. Since the number of strings of length $\leq n$ is finite the derivation contains a maximum of $\sum_{i=1}^{n} |\Sigma \cup V|^i$ steps.

By systematically trying out all derivation branches an algorithm can determine if the grammar G can generate the string x.

The algorithm either terminates if all intermediate strings during the derivation process have reached a length of $n + 1$ or if the string x has been generated.

Decision Problem

Definition

The set of all strings of length $\leq n$ that can be derived in m steps can be inductively defined as

$$T_0^n := \{S\}$$
$$T_{m+1}^n := T_m^n \cup \{w' \in (\Sigma \cup V)^* \mid |w'| \leq n \wedge (\exists w'' \in T_m^n)[w'' \Rightarrow w']\}$$

Decision Problem

Definition

The set of all strings of length $\leq n$ that can be derived in m steps can be inductively defined as

$$T_0^n := \{S\}$$
$$T_{m+1}^n := T_m^n \cup \{w' \in (\Sigma \cup V)^* \mid |w'| \leq n \wedge (\exists w'' \in T_m^n)[w'' \Rightarrow w']\}$$

Since $|T_m^n| \leq \sum_{i=1}^{n} |\Sigma \cup V|^i$ there exists an m_0 so that

$$T_{m_0}^n = T_{m_0+1}^n = T_{m_0+2}^n = \cdots =: T^n$$

Decision Problem

Algorithm for the decision problem

$n := |w|$
$T := \{S\}$
$T' := \emptyset$
while $T \neq T'$ **do**
  $T' := T$
  $T := T \cup \{w' \in (\Sigma \cup V)^* \mid |w'| \leq n \wedge (\exists w'' \in T')[w'' \Rightarrow w']\}$
end while
if $w \in T$ **then**
  return true
else
  return false
end if

Decision Problem

Remark

The algorithm requires exponential time, and the decision problem is NP-hard.

Backus-Naur Form

Formalism for the expression of context-free (type 2) grammars (short BNF form)

Backus-Naur Form

Formalism for the expression of context-free (type 2) grammars (short BNF form)

Was used when the programming language ALGOL60 was designed

Backus-Naur Form

For rules with the same left hand side

$$
\begin{aligned}
A &\rightarrow \beta_1 \\
A &\rightarrow \beta_2 \\
&\vdots \\
A &\rightarrow \beta_n
\end{aligned}
$$

Backus-Naur Form

For rules with the same left hand side

$$A \to \beta_1$$
$$A \to \beta_2$$
$$\vdots$$
$$A \to \beta_n$$

one can use the meta rule

$$A \to \beta_1 | \beta_2 | \ldots | \beta_n.$$

Backus-Naur Form

For rules with the same left hand side

$$A \to \beta_1$$
$$A \to \beta_2$$
$$\vdots$$
$$A \to \beta_n$$

one can use the meta rule

$$A \to \beta_1 | \beta_2 | \ldots | \beta_n.$$

(Backus and Naur actually used ::= instead of $\to$)

Extended Backus-Naur Form (EBNF)

additionally

Optional string

$$A \to \alpha[\beta]\gamma$$

for

$$A \to \alpha\gamma$$
$$A \to \alpha\beta\gamma$$

Extended Backus-Naur Form (EBNF)

and

Repetition

$$A \to \alpha\{\beta\}\gamma$$

for

$$A \to \alpha\gamma$$
$$A \to \alpha B\gamma$$
$$B \to \beta$$
$$B \to \beta B$$

Extended Backus-Naur Form (EBNF)

and

Repetition

$$A \to \alpha\{\beta\}\gamma$$

for

$$A \to \alpha\gamma$$
$$A \to \alpha B\gamma$$
$$B \to \beta$$
$$B \to \beta B$$

Meaning

β can occur arbitrary often between α and γ (also not at all)