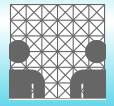


F2 — Automaten und formale Sprachen

Matthias Jantzen
(nach und mit Folienvorlagen von Berndt Farwer)

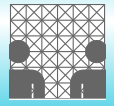
Fachbereich Informatik
AB „Theoretische Grundlagen der Informatik“ (TGI)
Universität Hamburg
jantzen@informatik.uni-hamburg.de

Akzeptoren vs. Generatoren

- Die Familie der regulären Sprachen haben wir als die von endlichen Automaten akzeptierten Sprachen kennengelernt.

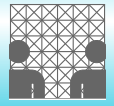
Akzeptoren vs. Generatoren

- Die Familie der regulären Sprachen haben wir als die von endlichen Automaten akzeptierten Sprachen kennengelernt.
- Automaten sind **Akzeptoren**.



Akzeptoren vs. Generatoren

- Die Familie der regulären Sprachen haben wir als die von endlichen Automaten akzeptierten Sprachen kennengelernt.
- Automaten sind **Akzeptoren**.
- Sie **prüfen**, ob ein Wort in der Sprache enthalten ist.



Akzeptoren vs. Generatoren

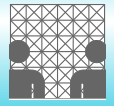
- Die Familie der regulären Sprachen haben wir als die von endlichen Automaten akzeptierten Sprachen kennengelernt.
- Automaten sind **Akzeptoren**.
- Sie **prüfen**, ob ein Wort in der Sprache enthalten ist.
- Sie **parsen** ihre Eingabe, definieren die Sprache also im Wesentlichen passiv.

Akzeptoren vs. Generatoren

- Die Familie der regulären Sprachen haben wir als die von endlichen Automaten akzeptierten Sprachen kennengelernt.
- Automaten sind **Akzeptoren**.
- Sie **prüfen**, ob ein Wort in der Sprache enthalten ist.
- Sie **parsen** ihre Eingabe, definieren die Sprache also im Wesentlichen passiv.
- Können Sprachen auch „aktiv“ definiert, also **erzeugt** bzw. **generiert** werden?

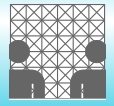
Akzeptoren vs. Generatoren

- Die Familie der regulären Sprachen haben wir als die von endlichen Automaten akzeptierten Sprachen kennengelernt.
- Automaten sind **Akzeptoren**.
- Sie **prüfen**, ob ein Wort in der Sprache enthalten ist.
- Sie **parsen** ihre Eingabe, definieren die Sprache also im Wesentlichen passiv.
- Können Sprachen auch „aktiv“ definiert, also **erzeugt** bzw. **generiert** werden?
- Dazu verwendet man Grammatiken.



Grammatik

- Bekannt aus der natürlichen Sprache: Grammatik

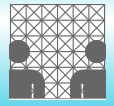


Grammatik

- Bekannt aus der natürlichen Sprache: Grammatik
- Regeln zur Bildung von Sätzen.

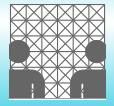
Grammatik

- Bekannt aus der natürlichen Sprache: Grammatik
- Regeln zur Bildung von Sätzen.
- Z.B. in der englischen Grammatik: Ein **Satz** besteht aus einer **Nominalphrase** gefolgt von einer **Verbalphrase**.



Grammatik

- Bekannt aus der natürlichen Sprache: Grammatik
- Regeln zur Bildung von Sätzen.
- Z.B. in der englischen Grammatik: Ein **Satz** besteht aus einer **Nominalphrase** gefolgt von einer **Verbalphrase**.
- **Wichtig:** die Reihenfolge der Satzbestandteile.



Englische Grammatik

$\langle \text{Satz} \rangle \longrightarrow \langle \text{Nominalphrase} \rangle \langle \text{Verbalphrase} \rangle$

$\langle \text{Nominalphrase} \rangle \longrightarrow \langle \text{Artikel} \rangle \langle \text{Nomen} \rangle$

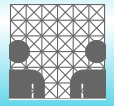
$\langle \text{Verbalphrase} \rangle \longrightarrow \langle \text{Verb} \rangle \langle \text{Objekt} \rangle$

$\langle \text{Objekt} \rangle \longrightarrow \langle \text{Nominalphrase} \rangle$

$\langle \text{Artikel} \rangle \longrightarrow \text{␣a} \mid \text{␣the}$

$\langle \text{Nomen} \rangle \longrightarrow \text{␣woman} \mid \text{␣man} \mid \text{␣book} \mid \text{␣golf}$

$\langle \text{Verb} \rangle \longrightarrow \text{␣reads} \mid \text{␣plays} \mid \text{␣buys}$



Englische Grammatik

$\langle \text{Satz} \rangle \longrightarrow \langle \text{Nominalphrase} \rangle \langle \text{Verbalphrase} \rangle$

$\langle \text{Nominalphrase} \rangle \longrightarrow \langle \text{Artikel} \rangle \langle \text{Nomen} \rangle$

$\langle \text{Verbalphrase} \rangle \longrightarrow \langle \text{Verb} \rangle \langle \text{Objekt} \rangle$

$\langle \text{Objekt} \rangle \longrightarrow \langle \text{Nominalphrase} \rangle$

$\langle \text{Artikel} \rangle \longrightarrow \text{␣a} \mid \text{␣the}$

$\langle \text{Nomen} \rangle \longrightarrow \text{␣woman} \mid \text{␣man} \mid \text{␣book} \mid \text{␣golf}$

$\langle \text{Verb} \rangle \longrightarrow \text{␣reads} \mid \text{␣plays} \mid \text{␣buys}$

- $\langle \textit{Name} \rangle$ ist eine **metalinguistische Variable**.

Backus/Naur-Notation

- Auch für Programmiersprachen wünscht man sich ein praktischeres Mittel, als einen Automaten:

Backus/Naur-Notation

- Auch für Programmiersprachen wünscht man sich ein praktischeres Mittel, als einen Automaten:

$\langle$type list$\rangle$ ::= $\langle$simple variable$\rangle$|$\langle$simple variable$\rangle$, $\langle$type list$\rangle$

$\langle$simple variable$\rangle$ ::= $\langle$variable identifier$\rangle$

$\langle$variable identifier$\rangle$ ::= $\langle$identifier$\rangle$

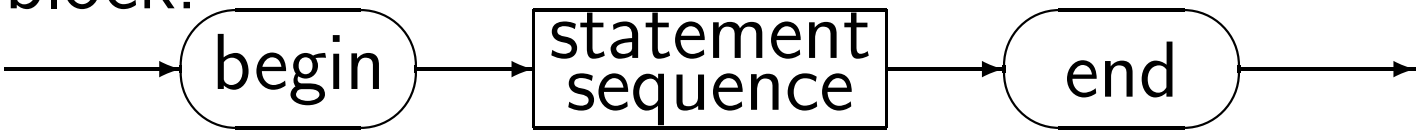
$\langle$identifier$\rangle$ ::= $\langle$letter$\rangle$|$\langle$identifier$\rangle\langle$letter$\rangle$|$\langle$identifier$\rangle\langle$digit$\rangle$

$\langle$digit$\rangle$ ::= $0|1|2|3|4|5|6|7|8|9$

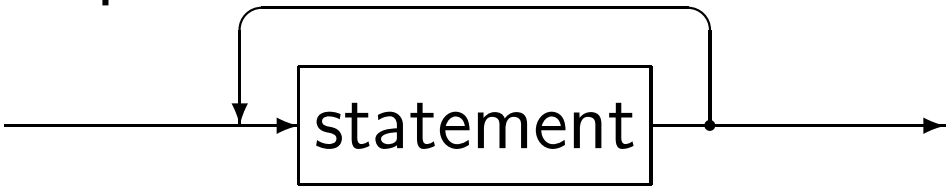
$\langle$letter$\rangle$ ::= $A|B|C|D|E|F|G|H|I|J|K|L|M|$
$N|O|P|Q|R|S|T|U|V|W|X|Y|Z$

Syntaxdiagramme

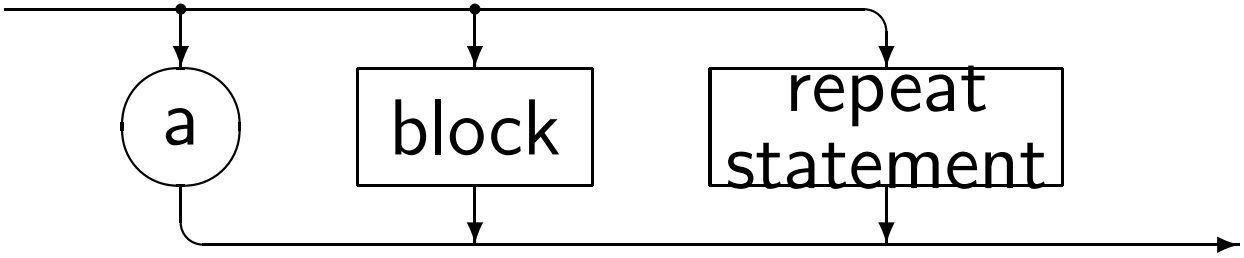
block:

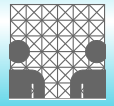


statement sequence:



statement:

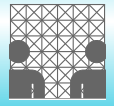




Formale Grammatik

Definition: Eine **Grammatik** wird beschrieben durch ein Quadrupel $G = (V_N, V_T, P, S)$ mit:

V_N ist ein endliches Alphabet von **Nonterminalen** (auch: metalinguistischen Variablen, Hilfszeichen oder syntaktischen Kategorien).



Formale Grammatik

Definition: Eine **Grammatik** wird beschrieben durch ein Quadrupel $G = (V_N, V_T, P, S)$ mit:

V_N ist ein endliches Alphabet von **Nonterminalen** (auch: metalinguistischen Variablen, Hilfszeichen oder syntaktischen Kategorien).

V_T ist ein mit V_N disjunktes endliches Alphabet von **Terminalen** (auch: Terminalsymbolen). Es muss also $V_T \cap V_N = \emptyset$ gelten.

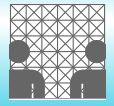
Formale Grammatik

Definition: Eine **Grammatik** wird beschrieben durch ein Quadrupel $G = (V_N, V_T, P, S)$ mit:

V_N ist ein endliches Alphabet von **Nonterminalen** (auch: metalinguistischen Variablen, Hilfszeichen oder syntaktischen Kategorien).

V_T ist ein mit V_N disjunktes endliches Alphabet von **Terminalen** (auch: Terminalsymbolen). Es muss also $V_T \cap V_N = \emptyset$ gelten.

Das Gesamtalphabet wird mit $V := V_T \uplus V_N$ bezeichnet.



Formale Grammatik

Definition: Eine **Grammatik** wird beschrieben durch ein Quadrupel $G = (V_N, V_T, P, S)$ mit:

V_N ist ein endliches Alphabet von **Nonterminalen** (auch: metalinguistischen Variablen, Hilfszeichen oder syntaktischen Kategorien).

V_T ist ein mit V_N disjunktes endliches Alphabet von **Terminalen** (auch: Terminalsymbolen). Es muss also $V_T \cap V_N = \emptyset$ gelten.

Das Gesamtalphabet wird mit $V := V_T \uplus V_N$ bezeichnet.

P ist eine endliche Menge von **Produktionen** oder **Regeln**. Es gilt $P \subseteq V^* \setminus V_T^* \times V^*$.

Formale Grammatik

Definition: Eine **Grammatik** wird beschrieben durch ein Quadrupel $G = (V_N, V_T, P, S)$ mit:

V_N ist ein endliches Alphabet von **Nonterminalen** (auch: metalinguistischen Variablen, Hilfszeichen oder syntaktischen Kategorien).

V_T ist ein mit V_N disjunktes endliches Alphabet von **Terminalen** (auch: Terminalsymbolen). Es muss also $V_T \cap V_N = \emptyset$ gelten.

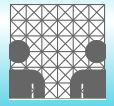
Das Gesamtalphabet wird mit $V := V_T \uplus V_N$ bezeichnet.

P ist eine endliche Menge von **Produktionen** oder **Regeln**. Es gilt $P \subseteq V^* \setminus V_T^* \times V^*$.

S ist das **Startsymbol** und es gilt $S \in V_N$.

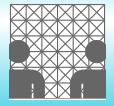
Konventionen

- Als **Nonterminale** werden meist *Großbuchstaben* verwendet $(A, B, C, \ldots)$.



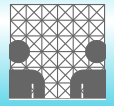
Konventionen

- Als **Nonterminale** werden meist *Großbuchstaben* verwendet $(A, B, C, \ldots)$.
- Als **Terminale** werden meist *Kleinbuchstaben* verwendet $(a, b, c, \ldots)$.



Konventionen

- Als **Nonterminale** werden meist *Großbuchstaben* verwendet $(A, B, C, \ldots)$.
- Als **Terminale** werden meist *Kleinbuchstaben* verwendet $(a, b, c, \ldots)$.
- $(v, w) \in P$ wird üblicherweise als $v \longrightarrow w$ geschrieben.

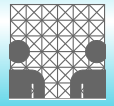


Konventionen

- Als **Nonterminale** werden meist *Großbuchstaben* verwendet $(A, B, C, \ldots)$.
- Als **Terminale** werden meist *Kleinbuchstaben* verwendet $(a, b, c, \ldots)$.
- $(v, w) \in P$ wird üblicherweise als $v \longrightarrow w$ geschrieben.
- Alternativen werden mit $|$ abgekürzt.

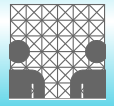
Konventionen

- Als **Nonterminale** werden meist *Großbuchstaben* verwendet $(A, B, C, \ldots)$.
- Als **Terminale** werden meist *Kleinbuchstaben* verwendet $(a, b, c, \ldots)$.
- $(v, w) \in P$ wird üblicherweise als $v \longrightarrow w$ geschrieben.
- Alternativen werden mit | abgekürzt.
- Die Mengen V_N und V_T können von den Regeln, d.h. der Menge P abgelesen werden. Es reicht deshalb meist, die **Regelmenge** und das **Startsymbol** anzugeben.



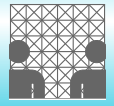
Beispiel: Grammatik

- Seien $V_N := \{A, B, C\}$ und $V_T := \{a, b\}$.



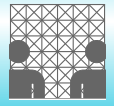
Beispiel: Grammatik

- Seien $V_N := \{A, B, C\}$ und $V_T := \{a, b\}$.
- Die Grammatik $G = (V_N, V_T, P, C)$ sei gegeben durch folgende Regelmenge P:



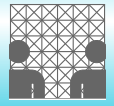
Beispiel: Grammatik

- Seien $V_N := \{A, B, C\}$ und $V_T := \{a, b\}$.
- Die Grammatik $G = (V_N, V_T, P, C)$ sei gegeben durch folgende Regelmenge P:
- $C \longrightarrow \not{\text{c}}aA \mid \not{\text{c}}bA \mid B$, $A \longrightarrow aA \mid bA$, $baB \longrightarrow bBa$, $bB \longrightarrow Bb$, $\not{\text{c}}B \longrightarrow \lambda$



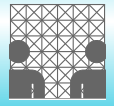
Beispiel: Grammatik

- Seien $V_N := \{A, B, C\}$ und $V_T := \{a, b\}$.
- Die Grammatik $G = (V_N, V_T, P, C)$ sei gegeben durch folgende Regelmenge P:
- $C \longrightarrow ¢aA \mid ¢bA \mid B$, $A \longrightarrow aA \mid bA$, $baB \longrightarrow bBa$, $bB \longrightarrow Bb$, $¢B \longrightarrow \lambda$
- Was kann man nun mit einer solchen Grammatik anfangen?



Beispiel: Grammatik

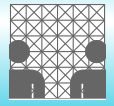
- Seien $V_N := \{A, B, C\}$ und $V_T := \{a, b\}$.
- Die Grammatik $G = (V_N, V_T, P, C)$ sei gegeben durch folgende Regelmenge P:
- $C \longrightarrow \not{\mathrm{c}}aA \mid \not{\mathrm{c}}bA \mid B$, $A \longrightarrow aA \mid bA$, $baB \longrightarrow bBa$, $bB \longrightarrow Bb$, $\not{\mathrm{c}}B \longrightarrow \lambda$
- Was kann man nun mit einer solchen Grammatik anfangen?
- Als Grundlage von Programmiersprachen reichen einfachere Grammatiken ...



Kontextfreie Grammatik

Definition: Eine **kontextfreie Grammatik** (CFG, *context-free grammar*) ist eine Grammatik $G := (V_N, V_T, P, S)$, für die einschränkend gilt:

$P \subseteq V_N \times V^*$ ist eine endliche Menge von **Produktionen** oder **Regeln**.

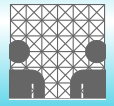


Kontextfreie Grammatik

Definition: Eine **kontextfreie Grammatik** (CFG, *context-free grammar*) ist eine Grammatik $G := (V_N, V_T, P, S)$, für die einschränkend gilt:

$P \subseteq V_N \times V^*$ ist eine endliche Menge von **Produktionen** oder **Regeln**.

- Eine kontextfreie Regel $(A, w) \in P$ wird üblicherweise als $A \longrightarrow w$ notiert.



Kontextfreie Grammatik

Definition: Eine **kontextfreie Grammatik** (CFG, *context-free grammar*) ist eine Grammatik $G := (V_N, V_T, P, S)$, für die einschränkend gilt:

$P \subseteq V_N \times V^*$ ist eine endliche Menge von **Produktionen** oder **Regeln**.

- Eine kontextfreie Regel $(A, w) \in P$ wird üblicherweise als $A \longrightarrow w$ notiert.
- Wir bezeichnen $A \longrightarrow \lambda$ als λ-**Produktion**.

Kontextfreie Grammatik

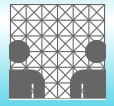
Definition: Eine **kontextfreie Grammatik** (CFG, *context-free grammar*) ist eine Grammatik $G := (V_N, V_T, P, S)$, für die einschränkend gilt:

$P \subseteq V_N \times V^*$ ist eine endliche Menge von **Produktionen** oder **Regeln**.

- Eine kontextfreie Regel $(A, w) \in P$ wird üblicherweise als $A \longrightarrow w$ notiert.
- Wir bezeichnen $A \longrightarrow \lambda$ als λ-**Produktion**.
- Gilt $P \subseteq V_N \times V^+$, so besitzt G keine λ-Produktionen und heißt λ-**frei**.

Beispiele: CFG

- $S \longrightarrow aSa \mid bSb \mid \lambda$ ist Kurzschreibweise der Regelmenge $P := \{(S, aSa), (S, bSb), (S, \lambda)\}$ einer CFG $G_1 := (\{S\}, \{a, b\}, P, S)$.



Beispiele: CFG

- $S \longrightarrow aSa \mid bSb \mid \lambda$ ist Kurzschreibweise der Regelmenge $P := \{(S, aSa), (S, bSb), (S, \lambda)\}$ einer CFG $G_1 := (\{S\}, \{a, b\}, P, S)$.
- **Programmiersprache:**
  E für expression, I für identifier, C für condition

  $S \longrightarrow I = E;$

  $I \longrightarrow \mathtt{u}$

  $E \longrightarrow (E) \mid -(E) \mid (E + E) \mid (E * E) \mid (E/E) \mid (E - E) \mid \mathtt{v}$

Beispiele: CFG

- $S \longrightarrow aSa \mid bSb \mid \lambda$ ist Kurzschreibweise der Regelmenge $P := \{(S, aSa), (S, bSb), (S, \lambda)\}$ einer CFG $G_1 := (\{S\}, \{a, b\}, P, S)$.
- **Programmiersprache:**
  E für expression, I für identifier, C für condition

  $S \longrightarrow I = E;$

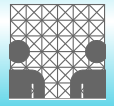
  $I \longrightarrow \texttt{u}$

  $E \longrightarrow (E) \mid -(E) \mid (E + E) \mid (E * E) \mid (E/E) \mid (E - E) \mid \texttt{v}$

  $S \longrightarrow S\texttt{while}(C)\{S\}; S \mid \lambda$

  $C \longrightarrow E > E$

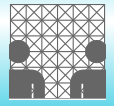
  $E \longrightarrow \texttt{a} \mid \texttt{b}$



Ableitungsrelation

Die **einschrittige Ableitung** eines Wortes v aus einem Wort u mittels der Produktionen der Grammatik G wird notiert als:

$$u \underset{G}{\Longrightarrow} v$$



Ableitungsrelation

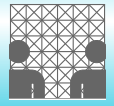
Die **einschrittige Ableitung** eines Wortes v aus einem Wort u mittels der Produktionen der Grammatik G wird notiert als:

$$u \underset{G}{\Longrightarrow} v$$

- Die Relation $\underset{G}{\Longrightarrow} \subseteq V^* \times V^*$ ist definiert durch:

$$u \underset{G}{\Longrightarrow} v \quad \text{gdw.} \quad \exists u_1, u_2 \in V^* : \exists w_l \in V^* V_N V^* :$$

$$\exists w_l \longrightarrow w_r \in P : \; u = u_1 w_l u_2 \wedge v = u_1 w_r u_2$$



Ableitungsrelation

Die **einschrittige Ableitung** eines Wortes v aus einem Wort u mittels der Produktionen der Grammatik G wird notiert als:

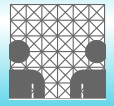
$$u \underset{G}{\Longrightarrow} v$$

- Die Relation $\underset{G}{\Longrightarrow} \subseteq V^* \times V^*$ ist definiert durch:

$$u \underset{G}{\Longrightarrow} v \quad \text{gdw.} \quad \exists u_1, u_2 \in V^* : \exists w_l \in V^* V_N V^* :$$

$$\exists w_l \longrightarrow w_r \in P : \; u = u_1 w_l u_2 \wedge v = u_1 w_r u_2$$

- Als **Ableitungsrelation** wird die reflexive, transitive Hülle der Relation $\underset{G}{\Longrightarrow}$ bezeichnet, die als $\overset{*}{\underset{G}{\Longrightarrow}} \subseteq V^* \times V^*$ geschrieben wird.

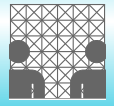


Ableitungsrelation bei CFG

- Für kontextfreie Grammatiken vereinfacht sich die Definition der *einschrittigen Ableitungsrelation* folgendermaßen:

$$u \underset{G}{\Longrightarrow} v \qquad \text{gdw.} \qquad \exists u_1, u_2 \in V^* :$$

$$\exists A \in V_N : \exists A \longrightarrow w \in P : \; u = u_1 A u_2 \wedge v = u_1 w u_2$$



Ableitungsrelation bei CFG

- Für kontextfreie Grammatiken vereinfacht sich die Definition der *einschrittigen Ableitungsrelation* folgendermaßen:

$$u \underset{G}{\Longrightarrow} v \qquad \text{gdw.} \qquad \exists u_1, u_2 \in V^* :$$

$$\exists A \in V_N : \exists A \longrightarrow w \in P : \; u = u_1 A u_2 \wedge v = u_1 w u_2$$

- Es gilt stets $w \overset{*}{\underset{G}{\Longrightarrow}} w$.

Ableitungsrelation bei CFG

- Für kontextfreie Grammatiken vereinfacht sich die Definition der *einschrittigen Ableitungsrelation* folgendermaßen:

$$u \underset{G}{\Longrightarrow} v \qquad \text{gdw.} \qquad \exists u_1, u_2 \in V^* :$$

$$\exists A \in V_N : \exists A \longrightarrow w \in P : \; u = u_1 A u_2 \wedge v = u_1 w u_2$$

- Es gilt stets $w \overset{*}{\underset{G}{\Longrightarrow}} w$.

- $u \in V^*$, mit $S \overset{*}{\underset{G}{\Longrightarrow}} u$ nennt man **Satzform**.

Ableitungsrelation bei CFG

- Für kontextfreie Grammatiken vereinfacht sich die Definition der *einschrittigen Ableitungsrelation* folgendermaßen:

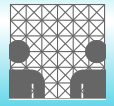
$$u \underset{G}{\Longrightarrow} v \qquad \text{gdw.} \qquad \exists u_1, u_2 \in V^* :$$

$$\exists A \in V_N : \exists A \longrightarrow w \in P : \; u = u_1 A u_2 \wedge v = u_1 w u_2$$

- Es gilt stets $w \overset{*}{\underset{G}{\Longrightarrow}} w$.

- $u \in V^*$, mit $S \overset{*}{\underset{G}{\Longrightarrow}} u$ nennt man **Satzform**.

- Letztendlich interessant sind nur die Satzformen über dem Terminalalphabet $\Rightarrow$ Definition der erzeugten Sprache.

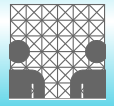


Ableitungen der Beispiel-CFG

- Als **Ableitung** der Länge n eines Wortes w in einer Grammatik bezeichnen wir eine *Sequenz* von

  Wörtern $w_1, w_2, \ldots w_{n-1}, w$ mit $S \underset{G}{\Longrightarrow} w_1$,

  $w_i \underset{G}{\Longrightarrow} w_{i+1}$ für $1 \leq i \leq n-1$ und $w_{n-1} \underset{G}{\Longrightarrow} w$.



Ableitungen der Beispiel-CFG

- Als **Ableitung** der Länge n eines Wortes w in einer Grammatik bezeichnen wir eine *Sequenz* von

  Wörtern $w_1, w_2, \ldots w_{n-1}, w$ mit $S \underset{G}{\Longrightarrow} w_1$,

  $w_i \underset{G}{\Longrightarrow} w_{i+1}$ für $1 \leq i \leq n-1$ und $w_{n-1} \underset{G}{\Longrightarrow} w$.

- Eine solche Abeitung schreiben wir häufig einfach als: $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_{n-1} \Rightarrow w$

Ableitungen der Beispiel-CFG

- Als **Ableitung** der Länge n eines Wortes w in einer Grammatik bezeichnen wir eine *Sequenz* von

  Wörtern $w_1, w_2, \ldots w_{n-1}, w$ mit $S \underset{G}{\Longrightarrow} w_1$,

  $w_i \underset{G}{\Longrightarrow} w_{i+1}$ für $1 \leq i \leq n-1$ und $w_{n-1} \underset{G}{\Longrightarrow} w$.

- Eine solche Abeitung schreiben wir häufig einfach als: $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_{n-1} \Rightarrow w$
- $S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$ ist Ableitung der Länge 3 in G_1

Ableitungen der Beispiel-CFG

- Als **Ableitung** der Länge n eines Wortes w in einer Grammatik bezeichnen wir eine *Sequenz* von

  Wörtern $w_1, w_2, \ldots w_{n-1}, w$ mit $S \underset{G}{\Longrightarrow} w_1$,

  $w_i \underset{G}{\Longrightarrow} w_{i+1}$ für $1 \leq i \leq n-1$ und $w_{n-1} \underset{G}{\Longrightarrow} w$.

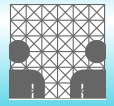
- Eine solche Abeitung schreiben wir häufig einfach als: $S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_{n-1} \Rightarrow w$
- $S \Rightarrow aSa \Rightarrow abSba \Rightarrow abba$ ist Ableitung der Länge 3 in G_1
- Eine Ableitung in der Programmiersprachengrammatik ...

Beispiel: Ableitung

$$
\begin{aligned}
S &\Rightarrow S\texttt{while}(C)\{S\};S \\
&\Rightarrow \texttt{while}(C)\{S\};S \\
&\Rightarrow \texttt{while}(E > E)\{S\};S \\
&\Rightarrow \texttt{while}(\texttt{a} > E)\{S\};S \\
&\Rightarrow \texttt{while}(\texttt{a} > \texttt{b})\{S\};S \\
&\Rightarrow \texttt{while}(\texttt{a} > \texttt{b})\{I = E;\};S \\
&\Rightarrow \texttt{while}(\texttt{a} > \texttt{b})\{\texttt{u} = E;\};S \\
&\Rightarrow \texttt{while}(\texttt{a} > \texttt{b})\{\texttt{u} = (E + E);\};S \\
&\Rightarrow \texttt{while}(\texttt{a} > \texttt{b})\{\texttt{u} = (\texttt{v} + E);\};S \\
&\Rightarrow \texttt{while}(\texttt{a} > \texttt{b})\{\texttt{u} = (\texttt{v} + \texttt{v});\};S \\
&\Rightarrow \texttt{while}(\texttt{a} > \texttt{b})\{\texttt{u} = (\texttt{v} + \texttt{v});\};
\end{aligned}
$$

Erzeugte Sprache

- Was ist die erzeugte Sprache einer Grammatik?



Erzeugte Sprache

- Was ist die erzeugte Sprache einer Grammatik?
- **Definition:** Sei $G = (V_N, V_T, P, S)$ eine Grammatik. Die von G **generierte** oder **erzeugte Sprache** ist:

$$L(G) := \{w \in V_T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}$$

Erzeugte Sprache

- Was ist die erzeugte Sprache einer Grammatik?
- **Definition:** Sei $G = (V_N, V_T, P, S)$ eine Grammatik. Die von G **generierte** oder **erzeugte Sprache** ist:

$$L(G) := \{w \in V_T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}$$

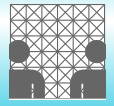
- Wir benötigen wieder eine Formalisierung des Äquivalenzbegriffes.

Erzeugte Sprache

- Was ist die erzeugte Sprache einer Grammatik?
- **Definition:** Sei $G = (V_N, V_T, P, S)$ eine Grammatik. Die von G **generierte** oder **erzeugte Sprache** ist:

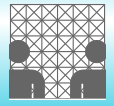
$$L(G) := \{w \in V_T^* \mid S \overset{*}{\underset{G}{\Longrightarrow}} w\}$$

- Wir benötigen wieder eine Formalisierung des Äquivalenzbegriffes.
- **Definition:** Zwei Grammatiken G_1 und G_2 heißen **äquivalent** gdw. sie die gleiche Sprache erzeugen, also wenn $L(G_1) = L(G_2)$ gilt.



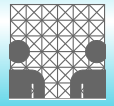
Kontextfreie Sprachen

- Die Familie aller kontextfreien Sprachen ist $\mathcal{C}f := \{M \mid \exists \,\mathsf{CFG}\ G :\ M = L(G)\}$.



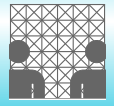
Kontextfreie Sprachen

- Die Familie aller kontextfreien Sprachen ist $\mathcal{C}f := \{M \mid \exists \,\text{CFG}\ G :\ M = L(G)\}$.
- **Beispiel:** Betrachten wir die Grammatik $G := (V_N, V_T, P, S)$ mit $P = \{S \longrightarrow aSb, S \longrightarrow \lambda\}$.



Kontextfreie Sprachen

- Die Familie aller kontextfreien Sprachen ist $\mathcal{C}f := \{M \mid \exists \text{ CFG } G : \ M = L(G)\}$.
- **Beispiel:** Betrachten wir die Grammatik $G := (V_N, V_T, P, S)$ mit $P = \{S \longrightarrow aSb, S \longrightarrow \lambda\}$.
  - Mit Startsymbol S genügt die Angabe der Regeln als: $S \longrightarrow aSb \mid \lambda$.



Kontextfreie Sprachen

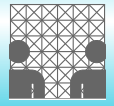
- Die Familie aller kontextfreien Sprachen ist $\mathcal{C}f := \{M \mid \exists \text{ CFG } G : \ M = L(G)\}$.
- **Beispiel:** Betrachten wir die Grammatik $G := (V_N, V_T, P, S)$ mit $P = \{S \longrightarrow aSb, S \longrightarrow \lambda\}$.
  - Mit Startsymbol S genügt die Angabe der Regeln als: $S \longrightarrow aSb \mid \lambda$.
  - Der senkrechte Strich wird als „oder“ gelesen. Wir haben es hier dennoch mit zwei Regeln zu tun!

Kontextfreie Sprachen

- Die Familie aller kontextfreien Sprachen ist $\mathcal{C}f := \{M \mid \exists \text{ CFG } G : \ M = L(G)\}$.
- **Beispiel:** Betrachten wir die Grammatik $G := (V_N, V_T, P, S)$ mit $P = \{S \longrightarrow aSb, S \longrightarrow \lambda\}$.
  - Mit Startsymbol S genügt die Angabe der Regeln als: $S \longrightarrow aSb \mid \lambda$.
  - Der senkrechte Strich wird als „oder“ gelesen. Wir haben es hier dennoch mit zwei Regeln zu tun!
  - G erzeugt die nicht reguläre Sprache $L(G) = DUP = \{a^n b^n \mid n \in I\!N\}$.

Eine Ableitung

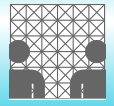
S



Eine Ableitung

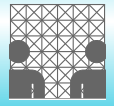
S

aSb



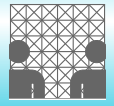
Eine Ableitung

S
aSb
aaSbb



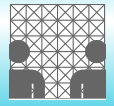
Eine Ableitung

S
aSb
aaSbb
aaaSbbb



Eine Ableitung

S
aSb
aaSbb
aaaSbbb
aaabbb

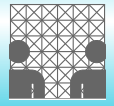


Eine Ableitung

S
aSb
aaSbb
aaaSbbb
aaabbb

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

zeigt, dass $aaabbb$ in $L(G_1)$ enthalten ist.



Eine Ableitung

S
aSb
aaSbb
aaaSbbb
aaabbb

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

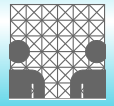
zeigt, dass $aaabbb$ in $L(G_1)$ enthalten ist.

Weitere Möglichkeiten der Darstellung einer Ableitung werden wir nächste Woche kennenlernen ...

Aufgabe

- Welche Sprache erzeugt die Grammatik mit folgenden Produktionen?

$$
\begin{aligned}
S &\longrightarrow aB \mid bA \\
A &\longrightarrow aS \mid bAA \mid a \\
B &\longrightarrow bS \mid aBB \mid b
\end{aligned}
$$

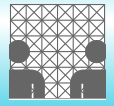


Aufgabe

- Welche Sprache erzeugt die Grammatik mit folgenden Produktionen?

$$
\begin{aligned}
S &\longrightarrow aB \mid bA \\
A &\longrightarrow aS \mid bAA \mid a \\
B &\longrightarrow bS \mid aBB \mid b
\end{aligned}
$$

- Jedes erzeugte Wort enthält gleichviele a's, wie b's.



Aufgabe

- Welche Sprache erzeugt die Grammatik mit folgenden Produktionen?

$$
\begin{aligned}
S &\longrightarrow aB \mid bA \\
A &\longrightarrow aS \mid bAA \mid a \\
B &\longrightarrow bS \mid aBB \mid b
\end{aligned}
$$

- Jedes erzeugte Wort enthält gleichviele a's, wie b's.
- Alle Wörter mit $|w|_a = |w|_b$ können erzeugt werden, ausser einem ...

Aufgabe

- Welche Sprache erzeugt die Grammatik mit folgenden Produktionen?

$$
\begin{aligned}
S &\longrightarrow aB \mid bA \\
A &\longrightarrow aS \mid bAA \mid a \\
B &\longrightarrow bS \mid aBB \mid b
\end{aligned}
$$

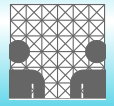
- Jedes erzeugte Wort enthält gleichviele a's, wie b's.
- Alle Wörter mit $|w|_a = |w|_b$ können erzeugt werden, ausser einem ...
- λ wird nicht erzeugt!

Aufgabe

- Welche Sprache erzeugt die Grammatik mit folgenden Produktionen?

$$\begin{aligned} S &\longrightarrow aB \mid bA \\ A &\longrightarrow aS \mid bAA \mid a \\ B &\longrightarrow bS \mid aBB \mid b \end{aligned}$$

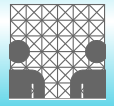
- Jedes erzeugte Wort enthält gleichviele a's, wie b's.
- Alle Wörter mit $|w|_a = |w|_b$ können erzeugt werden, ausser einem ...
- λ wird nicht erzeugt!
- Somit hat das kürzeste von G erzeugte Wort die Länge 2.



Beispiel

- Sei eine CFG G gegeben durch

$$S \longrightarrow AS \mid BA, \quad A \longrightarrow BAB \mid SB, \quad B \longrightarrow b$$

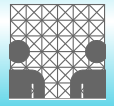


Beispiel

- Sei eine CFG G gegeben durch

$$S \longrightarrow AS \mid BA, \quad A \longrightarrow BAB \mid SB, \quad B \longrightarrow b$$

- Es gilt $L(G) = \emptyset$

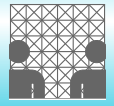


Beispiel

- Sei eine CFG G gegeben durch

$$S \longrightarrow AS \mid BA, \quad A \longrightarrow BAB \mid SB, \quad B \longrightarrow b$$

- Es gilt $L(G) = \emptyset$
- Offensichtlich sind hier alle Hilfszeichen überflüssig oder nutzlos.

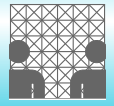


Beispiel

- Sei eine CFG G gegeben durch

$$S \longrightarrow AS \mid BA, \quad A \longrightarrow BAB \mid SB, \quad B \longrightarrow b$$

- Es gilt $L(G) = \emptyset$
- Offensichtlich sind hier alle Hilfszeichen überflüssig oder nutzlos.
- Wie kann dieser Sachverhalt formalisiert werden?



Beispiel

- Sei eine CFG G gegeben durch

$$S \longrightarrow AS \mid BA, \quad A \longrightarrow BAB \mid SB, \quad B \longrightarrow b$$

- Es gilt $L(G) = \emptyset$
- Offensichtlich sind hier alle Hilfszeichen überflüssig oder nutzlos.
- Wie kann dieser Sachverhalt formalisiert werden?
- Definition des Begriffes „reduzierte Grammatik“.

Reduzierte Grammatik

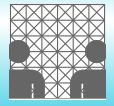
- **Definition:** Eine kontextfreie Grammatik $G := (V_N, V_T, P, S)$ heißt genau dann **reduziert**, wenn gilt:

Reduzierte Grammatik

- **Definition:** Eine kontextfreie Grammatik $G := (V_N, V_T, P, S)$ heißt genau dann **reduziert**, wenn gilt:

  1. Jedes Nonterminal ist **produktiv**, d.h.

$$\forall A \in V_N \text{ gilt } \exists w \in V_T^* : \; A \overset{*}{\underset{G}{\Longrightarrow}} w.$$



Reduzierte Grammatik

- **Definition:** Eine kontextfreie Grammatik $G := (V_N, V_T, P, S)$ heißt genau dann **reduziert**, wenn gilt:

  1. Jedes Nonterminal ist **produktiv**, d.h.

     $$\forall A \in V_N \text{ gilt } \exists w \in V_T^* : \; A \overset{*}{\underset{G}{\Longrightarrow}} w.$$

  2. Jedes Nonterminal ist **erreichbar**, d.h.

     $$\forall A \in V_N \text{ gilt } \exists u, v \in V^* : \; S \overset{*}{\underset{G}{\Longrightarrow}} uAv.$$

Reduzierte Grammatik

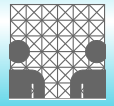
- **Definition:** Eine kontextfreie Grammatik $G := (V_N, V_T, P, S)$ heißt genau dann **reduziert**, wenn gilt:
  1. Jedes Nonterminal ist **produktiv**, d.h.

     $$\forall A \in V_N \text{ gilt } \exists w \in V_T^* : \; A \overset{*}{\underset{G}{\Longrightarrow}} w.$$

  2. Jedes Nonterminal ist **erreichbar**, d.h.

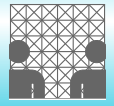
     $$\forall A \in V_N \text{ gilt } \exists u, v \in V^* : \; S \overset{*}{\underset{G}{\Longrightarrow}} uAv.$$

- Eine reduzierte Grammatik enthält möglicherweise trotzdem noch unnötig viele Symbole!



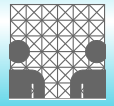
Ein Beispiel

- $S \longrightarrow A$, $A \longrightarrow B$, $B \longrightarrow aBa \mid bBb \mid \lambda$ ist reduziert.



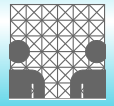
Ein Beispiel

- $S \longrightarrow A$, $A \longrightarrow B$, $B \longrightarrow aBa \mid bBb \mid \lambda$ ist reduziert.
- Die Grammatik hat **fünf Regeln** und **drei Nonterminale**.



Ein Beispiel

- $S \longrightarrow A$, $A \longrightarrow B$, $B \longrightarrow aBa \mid bBb \mid \lambda$ ist reduziert.
- Die Grammatik hat **fünf Regeln** und **drei Nonterminale**.
- Die äquivalente Grammatik mit nur **drei Regeln** $S \longrightarrow aSa \mid bSb \mid \lambda$ benötigt nur **ein Nonterminal**!



Ein Beispiel

- $S \longrightarrow A$, $A \longrightarrow B$, $B \longrightarrow aBa \mid bBb \mid \lambda$ ist reduziert.
- Die Grammatik hat **fünf Regeln** und **drei Nonterminale**.
- Die äquivalente Grammatik mit nur **drei Regeln** $S \longrightarrow aSa \mid bSb \mid \lambda$ benötigt nur **ein Nonterminal**!
- Zunächst trotzdem ein Ergebnis zu der Existenz reduzierter Grammatiken:

Ein Beispiel

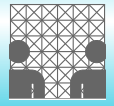
- $S \longrightarrow A$, $A \longrightarrow B$, $B \longrightarrow aBa \mid bBb \mid \lambda$ ist reduziert.
- Die Grammatik hat **fünf Regeln** und **drei Nonterminale**.
- Die äquivalente Grammatik mit nur **drei Regeln** $S \longrightarrow aSa \mid bSb \mid \lambda$ benötigt nur **ein Nonterminal**!
- Zunächst trotzdem ein Ergebnis zu der Existenz reduzierter Grammatiken:
- **Theorem:** Zu jeder CFG G kann effektiv eine äquivalente reduzierte CFG G' konstruiert werden.

Konstruktion reduzierter CFG

- Die Konstruktion besteht aus zwei separaten Teilen:

Konstruktion reduzierter CFG

- Die Konstruktion besteht aus zwei separaten Teilen:
- **Teil 1** entfernt Symbole (und Produktionen), so dass danach alle verwendeten Nonterminale auf reine Terminalwörter abgeleitet werden können, d.h. G *enthält danach nur noch produktive Nonterminalsymbole*.



Konstruktion reduzierter CFG

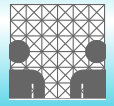
- Die Konstruktion besteht aus zwei separaten Teilen:
- **Teil 1** entfernt Symbole (und Produktionen), so dass danach alle verwendeten Nonterminale auf reine Terminalwörter abgeleitet werden können, d.h. G *enthält danach nur noch produktive Nonterminalsymbole*.
- **Teil 2** entfernt danach alle Symbole, die vom Startsymbol aus *nicht* erreichbar sind.

Konstruktion reduzierter CFG

- Die Konstruktion besteht aus zwei separaten Teilen:
- **Teil 1** entfernt Symbole (und Produktionen), so dass danach alle verwendeten Nonterminale auf reine Terminalwörter abgeleitet werden können, d.h. G *enthält danach nur noch produktive Nonterminalsymbole*.
- **Teil 2** entfernt danach alle Symbole, die vom Startsymbol aus *nicht* erreichbar sind.
- Die Reihenfolge der Anwendung dieser Schritte ist wichtig!

Konstruktion reduzierter CFG

- Die Konstruktion besteht aus zwei separaten Teilen:
- **Teil 1** entfernt Symbole (und Produktionen), so dass danach alle verwendeten Nonterminale auf reine Terminalwörter abgeleitet werden können, d.h. G *enthält danach nur noch produktive Nonterminalsymbole*.
- **Teil 2** entfernt danach alle Symbole, die vom Startsymbol aus *nicht* erreichbar sind.
- Die Reihenfolge der Anwendung dieser Schritte ist wichtig!
- . . . im Wesentlichen dasselbe Verfahren, wie bei der initialen Zusammenhangskomponente eines DFA.



Das Verfahren (1)

- **Teil 1 der Konstruktion:** Zu gegebener CFG $G := (V_N, V_T, P, S)$ definieren wir Mengen $M_i \subseteq V$ durch:

Das Verfahren (1)

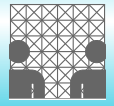
- **Teil 1 der Konstruktion:** Zu gegebener CFG $G := (V_N, V_T, P, S)$ definieren wir Mengen $M_i \subseteq V$ durch:
  - $M_0 := V_T$

Das Verfahren (1)

- **Teil 1 der Konstruktion:** Zu gegebener CFG $G := (V_N, V_T, P, S)$ definieren wir Mengen $M_i \subseteq V$ durch:
  - $M_0 := V_T$
  - $M_{i+1} := M_i \cup \{A \in V_N \mid \exists w \in M_i^* : \ A \longrightarrow w \in P\}$.

Das Verfahren (1)

- **Teil 1 der Konstruktion:** Zu gegebener CFG $G := (V_N, V_T, P, S)$ definieren wir Mengen $M_i \subseteq V$ durch:
  - $M_0 := V_T$
  - $M_{i+1} := M_i \cup \{A \in V_N \mid \exists w \in M_i^* : \ A \longrightarrow w \in P\}$.
  - Da für jedes $i \in I\!N$ $M_i \subseteq V$ endlich ist, gibt es einen Index k mit $M_k = M_{k+1}$.



Das Verfahren (1)

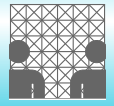
- **Teil 1 der Konstruktion:** Zu gegebener CFG $G := (V_N, V_T, P, S)$ definieren wir Mengen $M_i \subseteq V$ durch:
  - $M_0 := V_T$
  - $M_{i+1} := M_i \cup \{A \in V_N \mid \exists w \in M_i^* : \ A \longrightarrow w \in P\}$.
  - Da für jedes $i \in I\!N$ $M_i \subseteq V$ endlich ist, gibt es einen Index k mit $M_k = M_{k+1}$.
  - M_k enthält dann offensichtlich nur produktive Symbole.

Das Verfahren (1)

- **Teil 1 der Konstruktion:** Zu gegebener CFG $G := (V_N, V_T, P, S)$ definieren wir Mengen $M_i \subseteq V$ durch:
  - $M_0 := V_T$
  - $M_{i+1} := M_i \cup \{A \in V_N \mid \exists w \in M_i^* : \ A \longrightarrow w \in P\}$.
  - Da für jedes $i \in I\!N$ $M_i \subseteq V$ endlich ist, gibt es einen Index k mit $M_k = M_{k+1}$.
  - M_k enthält dann offensichtlich nur produktive Symbole.
  - Bilde neue CFG $G' = (V'_N, V'_T, P', S') := (M_k \cap V_N, V_T, P \cap (M_k \times M_k^*), S)$ mit $L(G') = L(G)$.

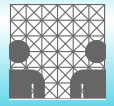
Das Verfahren (2)

- **Teil 2 der Konstruktion:** Es werden alle erreichbaren Symbole in G' bestimmt:



Das Verfahren (2)

- **Teil 2 der Konstruktion:** Es werden alle erreichbaren Symbole in G' bestimmt:
  - $M_0 := \{S'\}$



Das Verfahren (2)

- **Teil 2 der Konstruktion:** Es werden alle erreichbaren Symbole in G' bestimmt:
  - $M_0 := \{S'\}$
  - $M_{i+1} := M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : \ A \longrightarrow uBv \in P'\}$

Das Verfahren (2)

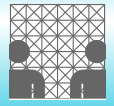
- **Teil 2 der Konstruktion:** Es werden alle erreichbaren Symbole in G' bestimmt:
  - $M_0 := \{S'\}$
  - $M_{i+1} := M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : \ A \longrightarrow uBv \in P'\}$
  - Wieder gilt $\exists k \in I\!N : \ M_k = M_{k+1}$

Das Verfahren (2)

- **Teil 2 der Konstruktion:** Es werden alle erreichbaren Symbole in G' bestimmt:
  - $M_0 := \{S'\}$
  - $M_{i+1} := M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : \ A \longrightarrow uBv \in P'\}$
  - Wieder gilt $\exists k \in I\!N : \ M_k = M_{k+1}$
  - M_k ist die Menge aller erreichbaren Nonterminale in G'.

Das Verfahren (2)

- **Teil 2 der Konstruktion:** Es werden alle erreichbaren Symbole in G' bestimmt:
  - $M_0 := \{S'\}$
  - $M_{i+1} := M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : \ A \longrightarrow uBv \in P'\}$
  - Wieder gilt $\exists k \in I\!N : \ M_k = M_{k+1}$
  - M_k ist die Menge aller erreichbaren Nonterminale in G'.
  - Definiere $G'' := (V''_N, V''_T, P'', S'')$, mit Produktionen, deren rechte und linke Seiten nur mit erreichbaren Nonterminalen gebildet werden.



Das Verfahren (2)

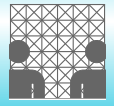
- **Teil 2 der Konstruktion:** Es werden alle erreichbaren Symbole in G' bestimmt:
  - $M_0 := \{S'\}$
  - $M_{i+1} := M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : \ A \longrightarrow uBv \in P'\}$
  - Wieder gilt $\exists k \in I\!N : \ M_k = M_{k+1}$
  - M_k ist die Menge aller erreichbaren Nonterminale in G'.
  - Definiere $G'' := (V''_N, V''_T, P'', S'')$, mit Produktionen, deren rechte und linke Seiten nur mit erreichbaren Nonterminalen gebildet werden.
  - $V''_N := M_k$,
    $P'' := P' \setminus \{(x, y) \mid x \notin M_k \vee y \notin (M_k \cup V'_T)^*$

Das Verfahren (2)

- **Teil 2 der Konstruktion:** Es werden alle erreichbaren Symbole in G' bestimmt:
  - $M_0 := \{S'\}$
  - $M_{i+1} := M_i \cup \{B \in V'_N \mid \exists A \in M_i \exists u, v \in V'^* : \ A \longrightarrow uBv \in P'\}$
  - Wieder gilt $\exists k \in I\!N : \ M_k = M_{k+1}$
  - M_k ist die Menge aller erreichbaren Nonterminale in G'.
  - Definiere $G'' := (V''_N, V''_T, P'', S'')$, mit Produktionen, deren rechte und linke Seiten nur mit erreichbaren Nonterminalen gebildet werden.
  - $V''_N := M_k$, $P'' := P' \setminus \{(x, y) \mid x \notin M_k \vee y \notin (M_k \cup V'_T)^*$
  - Die Menge V''_T der erreichbaren Terminalzeichen kann kleiner werden.

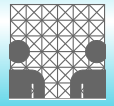
Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda,\ A \longrightarrow a$



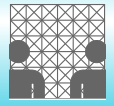
Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda,\ A \longrightarrow a$
- **1.$\rightarrow$ 2.**



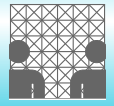
Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda$, $A \longrightarrow a$
- **1.$\rightarrow$ 2.**
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.



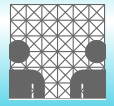
Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda$, $A \longrightarrow a$
- **1.$\rightarrow$ 2.**
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Das Nonterminal A ist nicht erreichbar, also wird $A \longrightarrow a$ gestrichen.



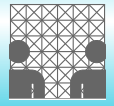
Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda$, $A \longrightarrow a$
- **1.$\rightarrow$ 2.**
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Das Nonterminal A ist nicht erreichbar, also wird $A \longrightarrow a$ gestrichen.
  - Resultat: $S \longrightarrow \lambda$



Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda$, $A \longrightarrow a$
- **1.$\rightarrow$ 2.**
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Das Nonterminal A ist nicht erreichbar, also wird $A \longrightarrow a$ gestrichen.
  - Resultat: $S \longrightarrow \lambda$
- **2.$\rightarrow$1.**



Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda$, $A \longrightarrow a$
- **1.$\rightarrow$ 2.**
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Das Nonterminal A ist nicht erreichbar, also wird $A \longrightarrow a$ gestrichen.
  - Resultat: $S \longrightarrow \lambda$
- **2.$\rightarrow$1.**
  - Alle Nonterminale sind erreichbar, somit wird keines gestrichen.

Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda$, $A \longrightarrow a$
- **1.→ 2.**
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Das Nonterminal A ist nicht erreichbar, also wird $A \longrightarrow a$ gestrichen.
  - Resultat: $S \longrightarrow \lambda$
- **2.→1.**
  - Alle Nonterminale sind erreichbar, somit wird keines gestrichen.
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.

Beispiel

- Gegeben: $S \longrightarrow AB \mid \lambda$, $A \longrightarrow a$
- **1.→ 2.**
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Das Nonterminal A ist nicht erreichbar, also wird $A \longrightarrow a$ gestrichen.
  - Resultat: $S \longrightarrow \lambda$
- **2.→1.**
  - Alle Nonterminale sind erreichbar, somit wird keines gestrichen.
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Resultat: $S \longrightarrow \lambda$ und $A \longrightarrow a$

Beispiel

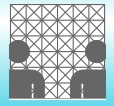
- Gegeben: $S \longrightarrow AB \mid \lambda$, $A \longrightarrow a$
- **1.→ 2.**
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Das Nonterminal A ist nicht erreichbar, also wird $A \longrightarrow a$ gestrichen.
  - Resultat: $S \longrightarrow \lambda$
- **2.→1.**
  - Alle Nonterminale sind erreichbar, somit wird keines gestrichen.
  - B ist unproduktiv, $S \longrightarrow AB$ wird gestrichen.
  - Resultat: $S \longrightarrow \lambda$ und $A \longrightarrow a$
- Beide Grammatiken erzeugen nur λ. Die zweite ist nicht reduziert!

Normalformen für CFGs

- Es ist manchmal unhandlich, viele mögliche Gestalten *rechter Seiten von Regeln* haben zu können.

Normalformen für CFGs

- Es ist manchmal unhandlich, viele mögliche Gestalten *rechter Seiten von Regeln* haben zu können.
- Auf den *linken Seiten* steht bei CFGs sowieso immer *genau ein* Nonterminal.



Normalformen für CFGs

- Es ist manchmal unhandlich, viele mögliche Gestalten *rechter Seiten von Regeln* haben zu können.
- Auf den *linken Seiten* steht bei CFGs sowieso immer *genau ein* Nonterminal.
- **Normalformen** helfen bei Konstruktionen und Beweisen.

Normalformen für CFGs

- Es ist manchmal unhandlich, viele mögliche Gestalten *rechter Seiten von Regeln* haben zu können.
- Auf den *linken Seiten* steht bei CFGs sowieso immer *genau ein* Nonterminal.
- **Normalformen** helfen bei Konstruktionen und Beweisen.
- **Definition:** Eine CFG $G := (V_N, V_T, P, S)$ ist in **Chomsky-Normalform** (CNF) gdw. alle Produktionen in P von der Form $A \longrightarrow BC$ oder $A \longrightarrow a$ für $A, B, C \in V_N$ und $a \in V_T$ sind.

Normalformen für CFGs

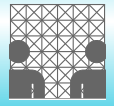
- Es ist manchmal unhandlich, viele mögliche Gestalten *rechter Seiten von Regeln* haben zu können.
- Auf den *linken Seiten* steht bei CFGs sowieso immer *genau ein* Nonterminal.
- **Normalformen** helfen bei Konstruktionen und Beweisen.
- **Definition:** Eine CFG $G := (V_N, V_T, P, S)$ ist in **Chomsky-Normalform** (CNF) gdw. alle Produktionen in P von der Form $A \longrightarrow BC$ oder $A \longrightarrow a$ für $A, B, C \in V_N$ und $a \in V_T$ sind.
- **Definition:** Eine CFG $G := (V_N, V_T, P, S)$ ist in **Greibach-Normalform** gdw. $P \subseteq V_N \times V_T \cdot V_N^*$.

Beispiele

- $S \longrightarrow SS \mid AB$, $A \longrightarrow AA \mid a$, $B \longrightarrow BB \mid b$ ist in Chomsky-Normalform, aber nicht in Greibach-Normalform.

Beispiele

- $S \longrightarrow SS \mid AB$, $A \longrightarrow AA \mid a$, $B \longrightarrow BB \mid b$ ist in Chomsky-Normalform, aber nicht in Greibach-Normalform.
- $S \longrightarrow aA$, $A \longrightarrow aA \mid aB$, $B \longrightarrow bB \mid b$ ist in Greibach-Normalform, jedoch nicht in Chomsky-Normalform.



Beispiele

- $S \longrightarrow SS \mid AB$, $A \longrightarrow AA \mid a$, $B \longrightarrow BB \mid b$ ist in Chomsky-Normalform, aber nicht in Greibach-Normalform.
- $S \longrightarrow aA$, $A \longrightarrow aA \mid aB$, $B \longrightarrow bB \mid b$ ist in Greibach-Normalform, jedoch nicht in Chomsky-Normalform.
- $S \longrightarrow aAbB$, $A \longrightarrow aA \mid \lambda$, $B \longrightarrow bB \mid \lambda$ ist weder in Greibach- noch in Chomsky-Normalform.

Beispiele

- $S \longrightarrow SS \mid AB$, $A \longrightarrow AA \mid a$, $B \longrightarrow BB \mid b$ ist in Chomsky-Normalform, aber nicht in Greibach-Normalform.
- $S \longrightarrow aA$, $A \longrightarrow aA \mid aB$, $B \longrightarrow bB \mid b$ ist in Greibach-Normalform, jedoch nicht in Chomsky-Normalform.
- $S \longrightarrow aAbB$, $A \longrightarrow aA \mid \lambda$, $B \longrightarrow bB \mid \lambda$ ist weder in Greibach- noch in Chomsky-Normalform.
- Wie sieht es mit $S \longrightarrow aA$, $aA \longrightarrow aaA \mid abB$, $bB \longrightarrow bbB \mid b$ aus?

Beispiele

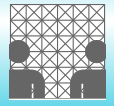
- $S \longrightarrow SS \mid AB$, $A \longrightarrow AA \mid a$, $B \longrightarrow BB \mid b$ ist in Chomsky-Normalform, aber nicht in Greibach-Normalform.
- $S \longrightarrow aA$, $A \longrightarrow aA \mid aB$, $B \longrightarrow bB \mid b$ ist in Greibach-Normalform, jedoch nicht in Chomsky-Normalform.
- $S \longrightarrow aAbB$, $A \longrightarrow aA \mid \lambda$, $B \longrightarrow bB \mid \lambda$ ist weder in Greibach- noch in Chomsky-Normalform.
- Wie sieht es mit $S \longrightarrow aA$, $aA \longrightarrow aaA \mid abB$, $bB \longrightarrow bbB \mid b$ aus?

  Diese Grammatik ist nicht einmal kontextfrei!

Beispiele

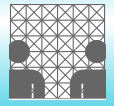
- $S \longrightarrow SS \mid AB$, $A \longrightarrow AA \mid a$, $B \longrightarrow BB \mid b$ ist in Chomsky-Normalform, aber nicht in Greibach-Normalform.
- $S \longrightarrow aA$, $A \longrightarrow aA \mid aB$, $B \longrightarrow bB \mid b$ ist in Greibach-Normalform, jedoch nicht in Chomsky-Normalform.
- $S \longrightarrow aAbB$, $A \longrightarrow aA \mid \lambda$, $B \longrightarrow bB \mid \lambda$ ist weder in Greibach- noch in Chomsky-Normalform.
- Wie sieht es mit $S \longrightarrow aA$, $aA \longrightarrow aaA \mid abB$, $bB \longrightarrow bbB \mid b$ aus?

  Diese Grammatik ist nicht einmal kontextfrei!
- $S \longrightarrow a \mid b$ ist sowohl in CNF, als auch in GNF!



Ausblick

- Normalformen sind nur sinnvoll, wenn sie für eine genügend große Menge von Grammatiken existieren.

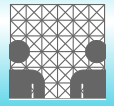


Ausblick

- Normalformen sind nur sinnvoll, wenn sie für eine genügend große Menge von Grammatiken existieren.
- Konstruktion einer CNF für beliebige CFG

Ausblick

- Normalformen sind nur sinnvoll, wenn sie für eine genügend große Menge von Grammatiken existieren.
- Konstruktion einer CNF für beliebige CFG
- Eigenschaften der kontextfreien Sprachen



Ausblick

- Normalformen sind nur sinnvoll, wenn sie für eine genügend große Menge von Grammatiken existieren.
- Konstruktion einer CNF für beliebige CFG
- Eigenschaften der kontextfreien Sprachen
- Grenzen der kontextfreien Sprachen