

FORMALE SYSTEME

12. Vorlesung: Das Wortproblem für kontextfreie Sprachen

Hannes Straß

Folien: © Markus Krötzsch, <https://iccl.inf.tu-dresden.de/web/FS2020>, CC BY 3.0 DE

TU Dresden, 18. November 2021

Rückblick

Ableitungen in CFGs als Bäume

Grammatik:

$S \rightarrow A \mid M \mid V$ $A \rightarrow (S+S)$

$M \rightarrow (S*S)$ $V \rightarrow x \mid y \mid z$

Ableitung:

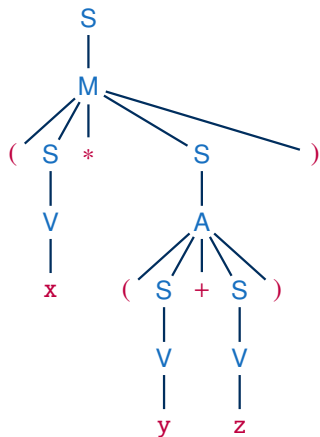
$S \Rightarrow M \Rightarrow (S*S) \Rightarrow (V*S)$

$\Rightarrow (x*S) \Rightarrow (x*A)$

$\Rightarrow (x*(S+S)) \Rightarrow (x*(V+S))$

$\Rightarrow (x*(y+S)) \Rightarrow (x*(y+V))$

$\Rightarrow (x*(y+z))$



Aufzeichnung startet . . .

Normalisierung kontextfreier Grammatiken

Motivation

Im Vergleich zu regulären Grammatiken erlauben CFGs wesentlich mehr Freiheiten bei der Formulierung von Produktionsregeln.

Für die Angabe von Algorithmen ist es hilfreich, die Grammatik erst in eine eingeschränktere Form zu überführen.

Beispiel: Wir haben in Vorlesung 2 gezeigt, wie man ϵ -Regeln in CFGs weitestgehend eliminieren kann, ohne dass sich dabei die Sprache ändert.

Es gibt mehrere sogenannte **Normalformen**, in die eine Grammatik durch derartige Vereinfachungen überführt werden kann.

Wiederholung: Erzeugung ϵ -freier CFGs

Der folgende Algorithmus kann ϵ -Regeln eliminieren.

Eingabe: CFG $G = \langle V, \Sigma, P, S \rangle$

Ausgabe: ϵ -freie CFG $G' = \langle V', \Sigma, P', S' \rangle$ mit $\mathbf{L}(G') = \mathbf{L}(G)$

- Initialisiere $P' := P$ und $V' := V$
- Berechne $V_\epsilon = \{A \in V \mid A \Rightarrow^* \epsilon\}$

(einfaches rekursives Verfahren, siehe Vorlesung 2)

- Entferne alle ϵ -Regeln aus P'
- Solange es in P' eine Regel $B \rightarrow xAy$ gibt, mit $x, y \in (\Sigma \cup V)^*$ und

$$A \in V_\epsilon \qquad |x| + |y| \geq 1 \qquad B \rightarrow xy \notin P'$$

wähle eine solche Regel und setze $P' := P' \cup \{B \rightarrow xy\}$

- Falls $S \in V_\epsilon$ dann definiere ein neues Startsymbol $S' \notin V$, setze $V' := V' \cup \{S'\}$ und $P' := P' \cup \{S' \rightarrow S, S' \rightarrow \epsilon\}$.
Andernfalls ($S \notin V_\epsilon$) verwenden wir einfach $S' := S$ als Startsymbol.

Eliminierung von Kettenregeln

Eine **Kettenregel** ist eine Regel der Form $A \rightarrow B$.

Eliminierung von Kettenregeln

Eine **Kettenregel** ist eine Regel der Form $A \rightarrow B$.

Satz: Für jede CFG G gibt es eine äquivalente CFG G' ohne Kettenregeln, d.h. eine CFG G' , für die $\mathbf{L}(G) = \mathbf{L}(G')$ gilt.

Eliminierung von Kettenregeln

Eine **Kettenregel** ist eine Regel der Form $A \rightarrow B$.

Satz: Für jede CFG G gibt es eine äquivalente CFG G' ohne Kettenregeln, d.h. eine CFG G' , für die $\mathbf{L}(G) = \mathbf{L}(G')$ gilt.

Beweis: Sei $G = \langle V, \Sigma, P, S \rangle$ ϵ -frei (o.B.d.A.).

G' ist die Grammatik $\langle V, \Sigma, P', S \rangle$, wobei sich P' wie folgt ergibt:

- Für jedes $A \in V$ bestimmen wir iterativ die Menge $E(A)$ aller $B \in V$, die man von A aus über Kettenregeln erreichen kann:
 - (1) $A \in E(A)$
 - (2) Falls $B \in E(A)$ und $B \rightarrow B' \in P$ mit $B' \in V$ dann $B' \in E(A)$(Schritt (2) wird wiederholt, bis keine Änderungen mehr auftreten.)
- Die Produktionsregeln von G' sind:

$$P' = \bigcup_{A \in V} \{A \rightarrow w \mid \text{es gibt } B \rightarrow w \in P \text{ mit } w \notin V \text{ und } B \in E(A)\} \quad \square$$

Beispiel: Eliminierung von Kettenregeln

Grammatik G mit Kettenregeln:

$$S \rightarrow A \mid M \mid V$$

$$A \rightarrow (S+S)$$

$$M \rightarrow (S*S)$$

$$V \rightarrow x \mid y \mid z$$

Beispiel: Eliminierung von Kettenregeln

Grammatik G mit Kettenregeln:

$$S \rightarrow A \mid M \mid V$$

$$A \rightarrow (S+S)$$

$$M \rightarrow (S*S)$$

$$V \rightarrow x \mid y \mid z$$

$$P' = \bigcup_{A \in V} \{A \rightarrow w \mid \text{es gibt } B \rightarrow w \in P \text{ mit } w \notin V \text{ und } B \in E(A)\}$$

Beispiel: Eliminierung von Kettenregeln

Grammatik G mit Kettenregeln:

$$S \rightarrow A \mid M \mid V$$

$$A \rightarrow (S+S)$$

$$M \rightarrow (S*S)$$

$$V \rightarrow x \mid y \mid z$$

$$P' = \bigcup_{A \in V} \{A \rightarrow w \mid \text{es gibt } B \rightarrow w \in P \text{ mit } w \notin V \text{ und } B \in E(A)\}$$

Mengen erreichbarer Variablen:

$$E(S) = \{S, A, M, V\}$$

$$E(A) = \{A\}$$

$$E(M) = \{M\}$$

$$E(V) = \{V\}$$

Beispiel: Eliminierung von Kettenregeln

Grammatik G mit Kettenregeln:

$$S \rightarrow A \mid M \mid V$$

$$A \rightarrow (S+S)$$

$$M \rightarrow (S*S)$$

$$V \rightarrow x \mid y \mid z$$

$$P' = \bigcup_{A \in V} \{A \rightarrow w \mid \text{es gibt } B \rightarrow w \in P \text{ mit } w \notin V \text{ und } B \in E(A)\}$$

Mengen erreichbarer Variablen:

$$E(S) = \{S, A, M, V\}$$

$$E(A) = \{A\}$$

$$E(M) = \{M\}$$

$$E(V) = \{V\}$$

Grammatik ohne Kettenregeln:

$$S \rightarrow (S+S) \mid (S*S) \mid x \mid y \mid z$$

$$A \rightarrow (S+S)$$

$$M \rightarrow (S*S)$$

$$V \rightarrow x \mid y \mid z$$

Quiz: Eliminierung von Kettenregeln

Quiz: Gegeben sei folgende CFG $G = \langle V, \Sigma, P, S \rangle$ mit $\Sigma = \{0, 1\}, \dots$

Die Chomsky-Normalform

Für die algorithmische Verarbeitung kontextfreier Grammatiken ist die folgende Form besonders praktisch:

Eine kontextfreie Grammatik $G = \langle V, \Sigma, P, S \rangle$ ist in **Chomsky-Normalform (CNF)**, wenn alle ihre Produktionsregeln eine der beiden folgenden Formen haben:

$$A \rightarrow BC \quad (\text{mit } B, C \in V) \quad \text{oder} \quad A \rightarrow c \quad (\text{mit } c \in \Sigma)$$

Die Chomsky-Normalform

Für die algorithmische Verarbeitung kontextfreier Grammatiken ist die folgende Form besonders praktisch:

Eine kontextfreie Grammatik $G = \langle V, \Sigma, P, S \rangle$ ist in **Chomsky-Normalform (CNF)**, wenn alle ihre Produktionsregeln eine der beiden folgenden Formen haben:

$$A \rightarrow BC \quad (\text{mit } B, C \in V) \quad \text{oder} \quad A \rightarrow c \quad (\text{mit } c \in \Sigma)$$

Beobachtung: Ist CFG G in Chomsky-Normalform, so gilt $\epsilon \notin \mathbf{L}(G)$.

- ↪ Grammatiken, deren Sprache das leere Wort enthalten, können nicht in CNF sein.
- ↪ Aber wie wir wissen, gibt es in ϵ -freien CFGs ohnehin höchstens eine Regel $S \rightarrow \epsilon$.
- ↪ Dieser Sonderfall kann in Algorithmen leicht berücksichtigt werden.
- ↪ Wir verzichten darauf, die CNF dafür noch zu erweitern.

Umwandlung in Chomsky-NF

Satz: Für jede Typ-2-Grammatik G mit $\epsilon \notin \mathbf{L}(G)$ gibt es eine äquivalente Typ-2-Grammatik $\text{CNF}(G)$ in Chomsky-Normalform, d.h. so dass $\mathbf{L}(G) = \mathbf{L}(\text{CNF}(G))$.

Umwandlung in Chomsky-NF

Satz: Für jede Typ-2-Grammatik G mit $\epsilon \notin \mathbf{L}(G)$ gibt es eine äquivalente Typ-2-Grammatik $\text{CNF}(G)$ in Chomsky-Normalform, d.h. so dass $\mathbf{L}(G) = \mathbf{L}(\text{CNF}(G))$.

Beweis: Wir haben bereits gezeigt:

- (1) Durch **Eliminierung von ϵ -Regeln** kann jede CFG in eine äquivalente ϵ -freie CFG umgewandelt werden.
- (2) Durch **Eliminierung von Kettenregeln** kann jede ϵ -freie CFG in eine äquivalente CFG umgewandelt werden, in der alle Regeln die Form $A \rightarrow w$ haben, mit $w \in \Sigma$ oder $|w| \geq 2$.

Umwandlung in Chomsky-NF

Satz: Für jede Typ-2-Grammatik G mit $\epsilon \notin \mathbf{L}(G)$ gibt es eine äquivalente Typ-2-Grammatik $\text{CNF}(G)$ in Chomsky-Normalform, d.h. so dass $\mathbf{L}(G) = \mathbf{L}(\text{CNF}(G))$.

Beweis: Wir haben bereits gezeigt:

- (1) Durch **Eliminierung von ϵ -Regeln** kann jede CFG in eine äquivalente ϵ -freie CFG umgewandelt werden.
- (2) Durch **Eliminierung von Kettenregeln** kann jede ϵ -freie CFG in eine äquivalente CFG umgewandelt werden, in der alle Regeln die Form $A \rightarrow w$ haben, mit $w \in \Sigma$ oder $|w| \geq 2$.

Zwei weitere Schritte führen zur Chomsky-NF:

- (3) **Extrahiere Regeln der Form $A \rightarrow c$** , so dass alle anderen Regeln $B \rightarrow w$ keine Terminale mehr in w enthalten.
- (4) **Zerlege Regeln der Form $A \rightarrow B_1 \cdots B_n$ für $n > 2$** , so dass solche Regeln nur noch mit $n = 2$ auftauchen.

Umwandlung in Chomsky-NF (2)

Satz: Für jede CFG G mit $\epsilon \notin \mathbf{L}(G)$ gibt es eine äquivalente CFG $\text{CNF}(G)$ in Chomsky-Normalform, d.h. so dass $\mathbf{L}(G) = \mathbf{L}(\text{CNF}(G))$.

Beweis: Schritt (3): Extrahiere Regeln der Form $A \rightarrow c$, so dass alle anderen Regeln $B \rightarrow w$ keine Terminale mehr in w enthalten.

Umwandlung in Chomsky-NF (2)

Satz: Für jede CFG G mit $\epsilon \notin \mathbf{L}(G)$ gibt es eine äquivalente CFG $\text{CNF}(G)$ in Chomsky-Normalform, d.h. so dass $\mathbf{L}(G) = \mathbf{L}(\text{CNF}(G))$.

Beweis: Schritt (3): Extrahiere Regeln der Form $A \rightarrow c$, so dass alle anderen Regeln $B \rightarrow w$ keine Terminale mehr in w enthalten.

- Führe für jedes Symbol $a \in \Sigma$ eine neue Variable V_a und eine Regel $V_a \rightarrow a$ ein.
- Für jede Produktionsregel $A \rightarrow w$ mit $|w| > 1$:
Ersetze jedes Vorkommen eines Symbols $a \in \Sigma$ in w durch V_a .

Anmerkungen:

Regeln $A \rightarrow w$ mit $|w| = 0$ existieren nicht (ϵ -Freiheit).

Regeln $A \rightarrow w$ mit $|w| = 1$ haben die Form $A \rightarrow c$ (keine Kettenregeln).

Umwandlung in Chomsky-NF (3)

Satz: Für jede Typ-2-Grammatik G mit $\epsilon \notin \mathbf{L}(G)$ gibt es eine äquivalente Typ-2-Grammatik $\text{CNF}(G)$ in Chomsky-Normalform, d.h. so dass $\mathbf{L}(G) = \mathbf{L}(\text{CNF}(G))$.

Beweis: Schritt (4): Zerlege Regeln der Form $A \rightarrow B_1 \cdots B_n$ für $n > 2$, so dass solche Regeln nur noch mit $n = 2$ auftauchen.

Umwandlung in Chomsky-NF (3)

Satz: Für jede Typ-2-Grammatik G mit $\epsilon \notin \mathbf{L}(G)$ gibt es eine äquivalente Typ-2-Grammatik $\text{CNF}(G)$ in Chomsky-Normalform, d.h. so dass $\mathbf{L}(G) = \mathbf{L}(\text{CNF}(G))$.

Beweis: Schritt (4): Zerlege Regeln der Form $A \rightarrow B_1 \cdots B_n$ für $n > 2$, so dass solche Regeln nur noch mit $n = 2$ auftauchen.

Für jede Produktionsregel $A \rightarrow B_1 \cdots B_n$ mit $n > 2$:

- Führe $n - 2$ neue Variablen C_1, \dots, C_{n-2} ein (intuitiv: C_i steht für $B_{i+1} \cdots B_n$).
- Ersetze die Regel durch neue Regeln:

$$\begin{aligned}A &\rightarrow B_1 C_1 \\C_1 &\rightarrow B_2 C_2 \\&\vdots \\C_{n-3} &\rightarrow B_{n-2} C_{n-2} \\C_{n-2} &\rightarrow B_{n-1} B_n\end{aligned}$$

Umwandlung in Chomsky-NF (4)

Satz: Für jede Typ-2-Grammatik G mit $\epsilon \notin \mathbf{L}(G)$ gibt es eine äquivalente Typ-2-Grammatik $\text{CNF}(G)$ in Chomsky-Normalform, d.h. so dass $\mathbf{L}(G) = \mathbf{L}(\text{CNF}(G))$.

Beweis: Die Korrektheit der Schritte lässt sich zeigen, indem man für eine beliebige Ableitung einer Grammatik G eine Ableitung der umgeformten Grammatik $\text{CNF}(G)$ angibt.

Zum Beispiel für Schritt 4:

Eine Anwendung der Regel $A \rightarrow B_1 \cdots B_n$ wird dargestellt durch die Ableitungen

$$A \Rightarrow B_1 C_1 \Rightarrow B_1 B_2 C_2 \Rightarrow \dots \Rightarrow B_1 \cdots B_{n-2} C_{n-2} \Rightarrow B_1 \cdots B_{n-2} B_{n-1} B_n$$

Korrektheit von Schritt 3 ist noch einfacher. □

Beispiel

ϵ -freie CFG ohne Kettenregeln:

$$S \rightarrow (S+S) \mid (S*S) \mid x \mid y \mid z$$

Beispiel

ϵ -freie CFG ohne Kettenregeln:

$$S \rightarrow (S+S) \mid (S*S) \mid x \mid y \mid z$$

Schritt (3): Extrahiere Regeln der Form $A \rightarrow c$, so dass alle anderen Regeln $B \rightarrow w$ keine Terminale mehr in w enthalten:

$$S \rightarrow V_{(} S V_{+} S V_{)} \mid V_{(} S V_{*} S V_{)} \mid x \mid y \mid z$$
$$V_{(} \rightarrow (\quad V_{+} \rightarrow + \quad V_{)} \rightarrow) \quad V_{*} \rightarrow * \quad \underbrace{V_x \rightarrow x \quad V_y \rightarrow y \quad V_z \rightarrow z}_{\text{ungenutzt (dürfen entfallen)}}$$

Beispiel

ϵ -freie CFG ohne Kettenregeln:

$$S \rightarrow (S+S) \mid (S*S) \mid x \mid y \mid z$$

Schritt (3): Extrahiere Regeln der Form $A \rightarrow c$, so dass alle anderen Regeln $B \rightarrow w$ keine Terminale mehr in w enthalten:

$$S \rightarrow V_{(} S V_{+} S V_{)} \mid V_{(} S V_{*} S V_{)} \mid x \mid y \mid z$$
$$V_{(} \rightarrow (\quad V_{+} \rightarrow + \quad V_{)} \rightarrow) \quad V_{*} \rightarrow * \quad \underbrace{V_x \rightarrow x \quad V_y \rightarrow y \quad V_z \rightarrow z}_{\text{ungenutzt (dürfen entfallen)}}$$

Schritt (4): Zerlege Regeln der Form $A \rightarrow B_1 \cdots B_n$ für $n > 2$, so dass solche Regeln nur noch mit $n = 2$ auftauchen:

$$S \rightarrow V_{(} C_1 \mid V_{(} D_1 \mid x \mid y \mid z$$
$$C_1 \rightarrow SC_2 \quad C_2 \rightarrow V_{+}C_3 \quad C_3 \rightarrow SV_{)}$$
$$D_1 \rightarrow SD_2 \quad D_2 \rightarrow V_{*}D_3 \quad D_3 \rightarrow SV_{)}$$
$$V_{(} \rightarrow (\quad V_{+} \rightarrow + \quad V_{)} \rightarrow) \quad V_{*} \rightarrow *$$

Ableitungsbäume für CNF-Grammatiken

Besonderheit der Chomsky-Normalform: Alle Ableitungsbäume sind im Inneren Binärbäume und haben daher eine sehr reguläre Struktur:

Ableitungsbäume für CNF-Grammatiken

Besonderheit der Chomsky-Normalform: Alle Ableitungsbäume sind im Inneren Binärbäume und haben daher eine sehr reguläre Struktur:

- Bei einem Wort w der Länge $|w|$ werden genau $|w|$ Regeln vom Typ $A \rightarrow c$ angewendet.
 - ↳ Es gibt in der Ebene darüber genau $|w|$ Variablen.

Ableitungsbäume für CNF-Grammatiken

Besonderheit der Chomsky-Normalform: Alle Ableitungsbäume sind im Inneren Binärbäume und haben daher eine sehr reguläre Struktur:

- Bei einem Wort w der Länge $|w|$ werden genau $|w|$ Regeln vom Typ $A \rightarrow c$ angewendet.
 - ↳ Es gibt in der Ebene darüber genau $|w|$ Variablen.
- Jede Anwendung einer Regel vom Typ $A \rightarrow BC$ erhöht die Anzahl der am Ende zu ersetzenden Variablen um 1.
 - ↳ Für ein Wort der Länge $|w|$ müssen genau $|w| - 1$ Regeln vom Typ $A \rightarrow BC$ angewendet werden.

Ableitungsbäume für CNF-Grammatiken

Besonderheit der Chomsky-Normalform: Alle Ableitungsbäume sind im Inneren Binärbäume und haben daher eine sehr reguläre Struktur:

- Bei einem Wort w der Länge $|w|$ werden genau $|w|$ Regeln vom Typ $A \rightarrow c$ angewendet.
 - ↪ Es gibt in der Ebene darüber genau $|w|$ Variablen.
- Jede Anwendung einer Regel vom Typ $A \rightarrow BC$ erhöht die Anzahl der am Ende zu ersetzenden Variablen um 1.
 - ↪ Für ein Wort der Länge $|w|$ müssen genau $|w| - 1$ Regeln vom Typ $A \rightarrow BC$ angewendet werden.

Wir folgern:

Satz: Ist G eine Grammatik in Chomsky-Normalform und $w \in \mathbf{L}(G)$, so hat jede Ableitung von w in G genau $2|w| - 1$ Ableitungsschritte.

Das Wortproblem für CFGs

Das Wortproblem für CFGs

Das **Wortproblem** für eine Sprache L über Alphabet Σ besteht darin, die folgende Funktion zu berechnen:

Eingabe: ein Wort $w \in \Sigma^*$

Ausgabe: „ja“ wenn $w \in L$ und „nein“ wenn $w \notin L$

Das Wortproblem für CFGs

Das **Wortproblem** für eine Sprache L über Alphabet Σ besteht darin, die folgende Funktion zu berechnen:

Eingabe: ein Wort $w \in \Sigma^*$

Ausgabe: „ja“ wenn $w \in L$ und „nein“ wenn $w \notin L$

Für Typ-2-Sprachen L können wir bereits ein (ineffizientes) Entscheidungsverfahren angeben:

- Es gibt eine CFG für L .
- Also gibt es eine CFG in Chomsky-NF für L .
- Also hat jedes Wort $w \in L$ eine Ableitung der Länge $2|w| - 1$.
- Wir können systematisch alle Ableitungen dieser Länge betrachten und prüfen, ob eine davon w erzeugt.

Das Wortproblem für CFGs

Das **Wortproblem** für eine Sprache L über Alphabet Σ besteht darin, die folgende Funktion zu berechnen:

Eingabe: ein Wort $w \in \Sigma^*$

Ausgabe: „ja“ wenn $w \in L$ und „nein“ wenn $w \notin L$

Für Typ-2-Sprachen L können wir bereits ein (ineffizientes) Entscheidungsverfahren angeben:

- Es gibt eine CFG für L .
- Also gibt es eine CFG in Chomsky-NF für L .
- Also hat jedes Wort $w \in L$ eine Ableitung der Länge $2|w| - 1$.
- Wir können systematisch alle Ableitungen dieser Länge betrachten und prüfen, ob eine davon w erzeugt.

↪ exponentieller Algorithmus

Geht es auch besser?

Quiz: Wortproblem für CFGs

Quiz: Wir betrachten die kontextfreie Grammatik $G = \langle V, \Sigma, P, S \rangle$ mit $\Sigma = \{0, 1\}$ und ...

Sakai, Kasami, Younger, Cocke und Schwartz

Verschiedene Menschen fanden eine bessere Lösung (besser als der naive, exponentielle Algorithmus):

- Itiroo Sakai, 1961: „Syntax in universal translation“
- Tadao Kasami, 1965: „An efficient recognition and syntax-analysis algorithm for context-free languages“
- Daniel H. Younger, 1967: „Recognition and parsing of context-free languages in time n^3 “
- John Cocke, Jacob T. Schwartz, 1970: „Programming languages and their compilers“

Das Ergebnis ist bekannt als CYK-Algorithmus (Cocke-Younger-Kasami).

CYK: Grundidee

Der CYK-Algorithmus arbeitet mit einer kontextfreien Grammatik G in Chomsky-NF.

Wie kann man prüfen, ob ein Wort $w = a_1 \cdot \dots \cdot a_n$ durch so eine Grammatik abgeleitet werden kann?

CYK: Grundidee

Der CYK-Algorithmus arbeitet mit einer kontextfreien Grammatik G in Chomsky-NF.

Wie kann man prüfen, ob ein Wort $w = a_1 \cdots a_n$ durch so eine Grammatik abgeleitet werden kann?

- Falls $|w| = 1$, dann ist $w \in \Sigma$ und es gilt:
 $w \in \mathbf{L}(G)$ genau dann, wenn es eine Regel $S \rightarrow w$ in G gibt.

CYK: Grundidee

Der CYK-Algorithmus arbeitet mit einer kontextfreien Grammatik G in Chomsky-NF.

Wie kann man prüfen, ob ein Wort $w = a_1 \cdots a_n$ durch so eine Grammatik abgeleitet werden kann?

- Falls $|w| = 1$, dann ist $w \in \Sigma$ und es gilt:
 $w \in \mathbf{L}(G)$ genau dann, wenn es eine Regel $S \rightarrow w$ in G gibt.
- Falls $|w| > 1$, dann ist:
 $w \in \mathbf{L}(G)$ genau dann, wenn es eine Regel $S \rightarrow AB$ und einen Index $i \in \{1, \dots, n-1\}$ gibt, so dass gilt:

$$A \Rightarrow^* a_1 \cdots a_i \quad \text{und} \quad B \Rightarrow^* a_{i+1} \cdots a_n$$

CYK: Grundidee

Der CYK-Algorithmus arbeitet mit einer kontextfreien Grammatik G in Chomsky-NF.

Wie kann man prüfen, ob ein Wort $w = a_1 \cdots a_n$ durch so eine Grammatik abgeleitet werden kann?

- Falls $|w| = 1$, dann ist $w \in \Sigma$ und es gilt:
 $w \in \mathbf{L}(G)$ genau dann, wenn es eine Regel $S \rightarrow w$ in G gibt.
- Falls $|w| > 1$, dann ist:
 $w \in \mathbf{L}(G)$ genau dann, wenn es eine Regel $S \rightarrow AB$ und einen Index $i \in \{1, \dots, n-1\}$ gibt, so dass gilt:

$$A \Rightarrow^* a_1 \cdots a_i \quad \text{und} \quad B \Rightarrow^* a_{i+1} \cdots a_n$$

Idee: Fall 2 reduziert das Problem $S \stackrel{?}{\Rightarrow}^* a_1 \cdots a_n$ auf zwei einfachere Probleme $A \stackrel{?}{\Rightarrow}^* a_1 \cdots a_i$ und $B \stackrel{?}{\Rightarrow}^* a_{i+1} \cdots a_n$, die man allerdings für alle Regeln $S \rightarrow AB$ und Indizes $i \in \{1, \dots, n-1\}$ lösen muss.

CYK: Praktische Umsetzung

Notation: Für $w = a_1 \cdots a_n$ schreiben wir $w_{i,j}$ für das Teilwort $a_i \cdots a_j$, also das Infix der Länge $j - i + 1$, welches an Position i beginnt.

Vorgehen: Wir berechnen für jedes Teilwort $w_{i,j}$ die Menge aller Variablen A , für die $A \Rightarrow^* w_{i,j}$ gilt. Diese Menge nennen wir $V[i,j]$.

- Wir beginnen mit den kürzesten Teilwörtern (Länge 1), also dem Fall $i = j$.
- Für längere Wörter betrachten wir jede mögliche Zweiteilung $w_{i,j} = w_{i,k}w_{k+1,j}$ und suchen Regeln der Form $A \rightarrow BC$, so dass $B \in V[i,k]$ und $C \in V[k+1,j]$.
(Dann gilt nämlich $A \in V[i,j]$.)

Ist am Ende das Startsymbol $S \in V[1, |w|]$, dann liegt w in der Sprache.

Notation

Für die Teilwörter $w_{i,j}$ gilt $i \leq j$.

→ Die Mengen $V[i,j]$ können als Dreiecksmatrix notiert werden.

Beispiel: Wir betrachten das Wort $w = a + b * c$ der Länge $|w| = 5$.

Darstellung der Mengen $V[i,j]$:

a	V[1, 1]	V[1, 2]	V[1, 3]	V[1, 4]	V[1, 5]
+		V[2, 2]	V[2, 3]	V[2, 4]	V[2, 5]
b			V[3, 3]	V[3, 4]	V[3, 5]
*				V[4, 4]	V[4, 5]
c					V[5, 5]
	a	+	b	*	c

Beispiel

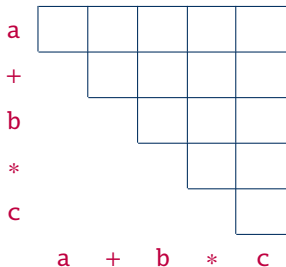
Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:



Beispiel

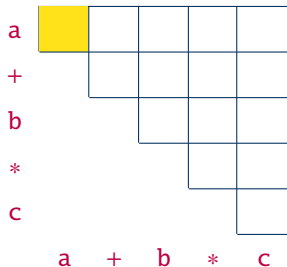
Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:



Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+					
b					
*					
c					
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+					
b					
*					
c					
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P			
b					
*					
c					
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i,j]$:

a	S				
+		P			
b			S		
*					
c					
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i,j]$:

a	S				
+		P			
b			S		
*				T	
c					
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P			
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P			
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P			
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P			
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

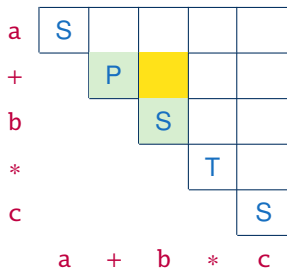
Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:



Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P	A		
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P	A		
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P	A		
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i,j]$:

a	S				
+		P	A		
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P	A		
b			S		
*				T	
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S				
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i,j]$:

a	S		S		
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S			
+		P	A			
b			S	S		
*				T	M	
c						S
	a	+	b	*	c	

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		S
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		S
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		S
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		S
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

Beispiel

Grammatik:

$S \rightarrow SA \mid SM \mid a \mid b \mid c$

$A \rightarrow PS \quad M \rightarrow TS \quad P \rightarrow + \quad T \rightarrow *$

Wort: $w = a + b * c$

Berechnung der Mengen $V[i, j]$:

a	S		S		S
+		P	A		A
b			S		S
*				T	M
c					S
	a	+	b	*	c

$S \in V[1, 5] \rightsquigarrow w$ kann erzeugt werden

Der CYK-Algorithmus

Eingabe: Wort $w = a_1 \cdots a_n$, CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$

Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“

for $i = 1, \dots, n$: // Teilwörter der Länge 1

$V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$

for $d = 1, \dots, n - 1$: // Differenz Endindex – Startindex

for $i = 1, \dots, n - d$: // Startindex

$j := i + d$

$V[i, j] := \emptyset$

for $k = i, \dots, j - 1$: // Trennindex

$V[i, j] := V[i, j] \cup \{A \in V \mid \text{es gibt eine Regel } A \rightarrow BC \in P \text{ mit}$
 $B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$

return $S \in V[1, n]$? „ja“ : „nein“

Bezug zur Tabelle

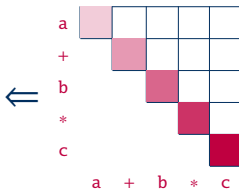
Eingabe: Wort $w = a_1 \cdots a_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$

Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“

for $i = 1, \dots, n$: // Teilwörter der Länge 1
 $V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$

for $d = 1, \dots, n - 1$: // Differenz Endind. – Startind.
for $i = 1, \dots, n - d$: // Startindex
 $j := i + d$
 $V[i, j] := \emptyset$
for $k = i, \dots, j - 1$: // Trennindex
 $V[i, j] := V[i, j] \cup$
 $\{A \in V \mid \text{es gibt } A \rightarrow BC \in P \text{ mit}$
 $B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$

return $S \in V[1, n]$? „ja“ : „nein“



Bezug zur Tabelle

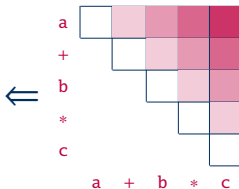
Eingabe: Wort $w = a_1 \dots a_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$

Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“

for $i = 1, \dots, n$: // Teilwörter der Länge 1
 $V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$

for $d = 1, \dots, n - 1$: // Differenz Endind. – Startind.
for $i = 1, \dots, n - d$: // Startindex
 $j := i + d$
 $V[i, j] := \emptyset$
for $k = i, \dots, j - 1$: // Trennindex
 $V[i, j] := V[i, j] \cup$
 $\{A \in V \mid \text{es gibt } A \rightarrow BC \in P \text{ mit}$
 $B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$

return $S \in V[1, n]$? „ja“ : „nein“



Bezug zur Tabelle

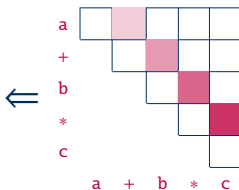
Eingabe: Wort $w = a_1 \dots a_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$

Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“

for $i = 1, \dots, n$: // Teilwörter der Länge 1
 $V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$

for $d = 1, \dots, n - 1$: // Differenz Endind. – Startind.
for $i = 1, \dots, n - d$: // Startindex
 $j := i + d$
 $V[i, j] := \emptyset$
for $k = i, \dots, j - 1$: // Trennindex
 $V[i, j] := V[i, j] \cup$
 $\{A \in V \mid \text{es gibt } A \rightarrow BC \in P \text{ mit}$
 $B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$

return $S \in V[1, n]$? „ja“ : „nein“



Bezug zur Tabelle

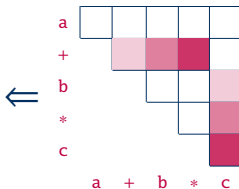
Eingabe: Wort $w = a_1 \dots a_n$,
CNF-Grammatik $G = \langle V, \Sigma, P, S \rangle$

Ausgabe: „ja“ wenn $w \in L(G)$; sonst „nein“

for $i = 1, \dots, n$: // Teilwörter der Länge 1
 $V[i, i] := \{A \in V \mid A \rightarrow a_i \in P\}$

for $d = 1, \dots, n - 1$: // Differenz Endind. – Startind.
for $i = 1, \dots, n - d$: // Startindex
 $j := i + d$
 $V[i, j] := \emptyset$
for $k = i, \dots, j - 1$: // Trennindex
 $V[i, j] := V[i, j] \cup$
 $\{A \in V \mid \text{es gibt } A \rightarrow BC \in P \text{ mit}$
 $B \in V[i, k] \text{ und } C \in V[k + 1, j]\}$

return $S \in V[1, n]$? „ja“ : „nein“



Komplexität

Wie komplex ist der CYK-Algorithmus?

- Es gibt drei geschachtelte Schleifen mit Lauflänge $O(|w|)$.
- Operationen für Zugriff auf Grammatik und Teilmengen von V :
 - polynomiell bezüglich der Größe der Grammatik;
 - konstant, wenn die Grammatik vorher bekannt und fest ist.

Satz: Das Wortproblem für kontextfreie Sprachen ist in $O(n^3)$ lösbar.

Anmerkung: Die Umwandlung in Chomsky-NF verlangt $\epsilon \notin \mathbf{L}(G)$, aber im Fall $\epsilon \in \mathbf{L}(G)$ kann man eine Grammatik G' mit $\epsilon \notin \mathbf{L}(G')$ erzeugen und $w = \epsilon$ gesondert testen.

Zusammenfassung und Ausblick

Jede Typ-2-Grammatik kann in eine äquivalente* Grammatik in **Chomsky-Normalform** umgewandelt werden.

* Besser: äquivalent bis auf das leere Wort, welches von der normalisierten Grammatik nicht erzeugt wird.

Mit Hilfe des **CYK-Algorithmus** ist das Wortproblem für kontextfreie Sprachen in polynomieller Zeit lösbar.

Offene Fragen:

- Haben kontextfreie Sprachen ein Berechnungsmodell?
- Wie sehen nicht-kontextfreie Sprachen aus und wie erkennt man sie?