



Theoretische Grundlagen der Informatik

Vorlesung am 12.01.2022

Torsten Ueckerdt | 12. Januar 2022

Letzte Vorlesung

Eine **Grammatik** ist $G = (\Sigma, V, S, R)$ mit

- Alphabet Σ aller Terminale,
- Menge V aller Variablen / Nichtterminale,
- Startvariable S aus V ,
- Menge R aller Regeln / Ableitungen, wobei eine Ableitung ein Tupel (ℓ, r) aus Wörtern in $(\Sigma \cup V)^*$ ist.

Wir schreiben auch $\ell \rightarrow r$ für die Regel (ℓ, r) und entsprechend $S \xrightarrow{*} w$ falls w aus S abgeleitet werden kann.

Die **erzeugte Sprache** ist $L(G) = \{w \in \Sigma^* \mid S \xrightarrow{*} w\}$.

Eine Grammatik ist **kontextfrei** oder **Chomsky Typ-2**, wenn alle Regeln die folgende Form haben:

$$A \rightarrow v \quad \text{mit } A \in V \text{ und } v \in (\Sigma \cup V)^*$$

Letzte Vorlesung

Beispiel

Grammatik $G = (\Sigma, V, S, R)$ mit

$$V = \{S, A, B\}$$

$$\Sigma = \{0, 1\}$$

$$R = \{S \rightarrow 0B \mid 1A, \\ A \rightarrow 0 \mid 0S \mid 1AA, \\ B \rightarrow 1 \mid 1S \mid 0BB\}$$

erzeugt die Sprache $L = L(G) \subseteq \{0, 1\}^+$ aller Wörter mit genauso vielen Einsen wie Nullen.

Zum Beispiel ist $110010 \in L(G)$, wegen der Ableitung

$$S \rightarrow 1A \rightarrow 11AA \rightarrow 11A0 \rightarrow 110S0 \rightarrow 1100B0 \rightarrow 110010.$$

Heute

- Syntaxbäume
- Eindeutigkeit
- Chomsky-Normalform
- CYK-Algorithmus

Syntaxbäume

Eine Grammatik ist **kontextfrei** oder **Chomsky Typ-2**, wenn alle Regeln die folgende Form haben:

$$A \rightarrow v \quad \text{mit } A \in V \text{ und } v \in (\Sigma \cup V)^*$$

Syntaxbäume visualisieren die Ableitung eines einzelnen Wortes in einer kontextfreien Grammatik.

- An der Wurzel eines Syntaxbaumes steht das Startsymbol.
- Jeder innere Knoten enthält eine Variable.
- Die Blätter sind Symbole aus Σ oder ε .
- Wenn ein innerer Knoten A als Nachfolger von links nach rechts $(\alpha_1, \dots, \alpha_r) \in V \cup \Sigma$ hat, so muss $A \rightarrow \alpha_1 \cdots \alpha_r$ eine Ableitungsregel der Grammatik sein.

Syntaxbäume – Beispiel

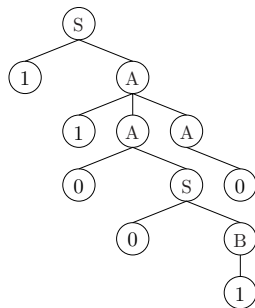
Betrachte die Regeln:

$$R = \{S \rightarrow 0B|1A, \quad A \rightarrow 0|0S|1AA, \quad B \rightarrow 1|1S|0BB\}$$

Betrachte die Ableitung:

$$S \rightarrow 1A \rightarrow 11AA \rightarrow 11A0 \rightarrow 110S0 \rightarrow 1100B0 \rightarrow 110010.$$

- Zu jeder Ableitung gehört genau ein Syntaxbaum.



Syntaxbäume – Beispiel

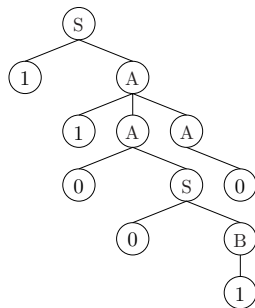
Betrachte die Regeln:

$$R = \{S \rightarrow 0B|1A, \quad A \rightarrow 0|0S|1AA, \quad B \rightarrow 1|1S|0BB\}$$

Betrachte die Ableitung:

$$S \rightarrow 1A \rightarrow 11AA \rightarrow 11A0 \rightarrow 110S0 \rightarrow 1100B0 \rightarrow 110010.$$

- Zu jeder Ableitung gehört genau ein Syntaxbaum.



Betrachte die Ableitung:

$$S \rightarrow 1A \rightarrow 11AA \rightarrow 110SA \rightarrow 1100BA \rightarrow 11001A \rightarrow 110010$$

- Zu einem Syntaxbaum können jedoch verschiedene Ableitungen des gleichen Wortes gehören.

Links/Rechtsableitung, Eindeutigkeit

Wegen der Kontextfreiheit ist bei **Chomsky-2 Grammatiken** die Reihenfolge, in der abgeleitet wird, für das Ergebnis unerheblich.

Definition.

Eine **Linksableitung** (**Rechtsableitung**) ist eine Ableitung, bei der in jedem Schritt die linkeste (rechtste) Variable abgeleitet wird.

Definition.

Eine kontextfreie Grammatik G heißt **eindeutig**, wenn es für jedes Wort $w \in L(G)$ genau einen Syntaxbaum gibt.

Eine kontextfreie Sprache L heißt **eindeutig**, wenn es eine eindeutige Grammatik G mit $L(G) = L$ gibt. Ansonsten heißt L **inhärent mehrdeutig**.

Beispiel: eindeutig

Die Sprache

$$L = \{0^n 1^n : n \geq 1\}$$

erzeugt durch die Grammatik

$$V = \{S\}$$

$$\Sigma = \{0, 1\}$$

$$R = \{S \rightarrow 01 \mid 0S1\}$$

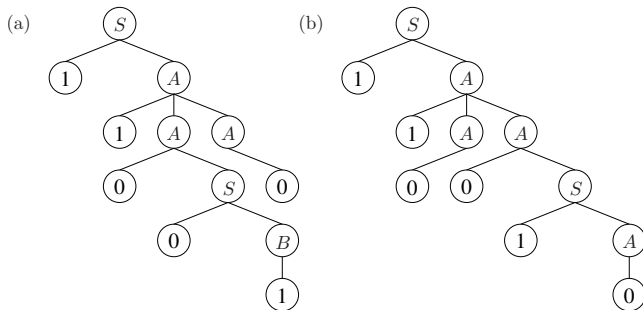
ist eindeutig.

Beispiel: nicht eindeutig

Die Sprache gegeben durch die Regeln

$$R = \{S \rightarrow 0B \mid 1A, A \rightarrow 0 \mid 0S \mid 1AA, B \rightarrow 1 \mid 1S \mid 0BB\}$$

ist nicht eindeutig.



Heute

- Syntaxbäume
- Eindeutigkeit
- Chomsky-Normalform
- CYK-Algorithmus

Chomsky-Normalform

Definition.

Eine kontextfreie Grammatik ist in **Chomsky-Normalform**, wenn alle Regeln von der Form:

$$A \rightarrow BC \quad \text{oder} \quad A \rightarrow a$$

sind, mit $A, B, C \in V$ und $a \in \Sigma$.

Erinnerung:

In **kontextfreien** Grammatiken sind alle Regeln von der Form:

$$A \rightarrow v$$

mit $A \in V$

und $v \in (\Sigma \cup V)^*$

Chomsky-Normalform

Definition.

Eine kontextfreie Grammatik ist in **Chomsky-Normalform**, wenn alle Regeln von der Form:

$$A \rightarrow BC \quad \text{oder} \quad A \rightarrow a$$

sind, mit $A, B, C \in V$ und $a \in \Sigma$.

- Grammatiken in Chomsky-Normalform können also nicht das Wort ε erzeugen.
- Für kontextfreie Sprachen, die ε enthalten, lässt sich eine Grammatik in Chomsky-Normalform leicht “erweitern” durch die Regeln

$$S' \rightarrow \varepsilon \quad \text{und} \quad S' \rightarrow S,$$

wobei S' ein neues Startsymbol zur Erzeugung von ε ist.

Erinnerung:

In **kontextfreien** Grammatiken sind alle Regeln von der Form:

$$A \rightarrow v$$

mit $A \in V$

und $v \in (\Sigma \cup V)^*$

Chomsky-Normalform

Satz.

Jede kontextfreie Grammatik kann in eine äquivalente Grammatik in **erweiterter Chomsky-Normalform** überführt werden.

Eine Grammatik in erweiterter Chomsky-Normalform ist eine Grammatik in Chomsky-Normalform mit den zusätzlichen Regeln $S' \rightarrow \varepsilon$ und $S' \rightarrow S$.

Beweis (konstruktiv):

- Wir geben eine “Schritt-für-Schritt”-Überführung der Regeln einer beliebigen kontextfreien Grammatik in Regeln in Normalform an.
- Großbuchstaben repräsentieren immer Variablen / Nichtterminale.
- Kleinbuchstaben repräsentieren immer Terminale.

Chomsky-Normalform

Wir veranschaulichen den Beweis an folgendem Beispiel:

Grammatik $G = (\Sigma, V, S, R)$ mit

$$\Sigma = \{a, b\}$$

$$V = \{A, B, C, D, E, S\}$$

und der folgenden Regelmenge R :

$$S \rightarrow A \mid aAa \mid bBb \mid \varepsilon$$

$$A \rightarrow C \mid a$$

$$B \rightarrow b$$

$$C \rightarrow AF \mid CDE \mid \varepsilon$$

$$D \rightarrow A \mid B \mid ab$$

$$E \rightarrow B$$

$$F \rightarrow D \mid E$$

Schritt 1

Vorgehen:

Ersetze dazu in allen rechten Seiten von Regeln Symbole $a \in \Sigma$ durch neue Variablen Y_a und füge die Regeln $Y_a \rightarrow a$ hinzu.

Ziel:

Alle Regeln enthalten auf der rechten Seite nur Symbole aus V oder nur ein Symbol aus Σ .

Schritt 1

Vorgehen:

Ersetze dazu in allen rechten Seiten von Regeln Symbole $a \in \Sigma$ durch neue Variablen Y_a und füge die Regeln $Y_a \rightarrow a$ hinzu.

$$S \rightarrow A \mid aAa \mid bBb \mid \varepsilon$$

$$A \rightarrow C \mid a$$

$$B \rightarrow b$$

$$C \rightarrow AF \mid CDE \mid \varepsilon$$

$$D \rightarrow A \mid B \mid ab$$

$$E \rightarrow B$$

$$F \rightarrow D \mid E$$

 \Rightarrow

$$S \rightarrow A \mid Y_aAY_a \mid Y_bBY_b \mid \varepsilon$$

$$A \rightarrow C \mid a$$

$$B \rightarrow b$$

$$C \rightarrow AF \mid CDE \mid \varepsilon$$

$$D \rightarrow A \mid B \mid Y_aY_b$$

$$E \rightarrow B$$

$$F \rightarrow D \mid E$$

$$Y_a \rightarrow a$$

$$Y_b \rightarrow b$$

Ziel:

Alle Regeln enthalten auf der rechten Seite nur Symbole aus V oder nur ein Symbol aus Σ .

Schritt 2

Ziel:

- Alle rechten Seiten haben Länge ≤ 2 .

Vorgehen:

- Sei $A \rightarrow B_1 \cdots B_m$ Regel mit $m > 2$.
- Führe $m - 2$ neue Variablen C_1, \dots, C_{m-2} ein, und ersetze die Regel

$$A \rightarrow B_1 \cdots B_m$$

durch neue Regeln

$$\begin{aligned} A &\rightarrow B_1 C_1 \\ C_i &\rightarrow B_{i+1} C_{i+1} \quad \text{für } 1 \leq i \leq m-3 \\ C_{m-2} &\rightarrow B_{m-1} B_m \end{aligned}$$

Schritt 2

$$S \rightarrow A \mid Y_a A Y_a \mid Y_b B Y_b \mid \varepsilon$$

$$A \rightarrow C \mid a$$

$$B \rightarrow b$$

$$C \rightarrow AF \mid CDE \mid \varepsilon$$

$$D \rightarrow A \mid B \mid Y_a Y_b$$

$$E \rightarrow B$$

$$F \rightarrow D \mid E$$

 \Rightarrow

$$S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \mid \varepsilon$$

$$A \rightarrow C \mid a$$

$$B \rightarrow b$$

$$C \rightarrow AF \mid C C_3 \mid \varepsilon$$

$$D \rightarrow A \mid B \mid Y_a Y_b$$

$$E \rightarrow B$$

$$F \rightarrow D \mid E$$

$$C_1 \rightarrow A Y_a$$

$$C_2 \rightarrow B Y_b$$

$$C_3 \rightarrow DE$$

$$Y_a \rightarrow a$$

$$Y_b \rightarrow b$$

$$Y_a \rightarrow a$$

$$Y_b \rightarrow b$$

Ziel:

Alle rechten Seiten
haben Länge ≤ 2 .

Schritt 3

Vorgehen, Phase 1:

Finde die Menge V' aller Variablen A für die $A \xrightarrow{*} \varepsilon$ existiert.

- Es werden erst alle A mit $A \rightarrow \varepsilon$ in V' aufgenommen.
- Dann wird geprüft, ob neue Regeln $B \rightarrow \varepsilon$ entstehen, wenn man A in allen Regeln auf der rechten Seite A durch ε ersetzt.
- Ist dies der Fall, so werden die entsprechenden Variablen B in V' aufgenommen und genauso behandelt.
- Das Verfahren hört auf, wenn V' sich nicht mehr ändert.

Ziel:

Es kommen keine Regeln $A \rightarrow \varepsilon$ vor.

Schritt 3

Vorgehen, Phase 1:

Finde die Menge V' aller Variablen A für die $A \xrightarrow{*} \varepsilon$ existiert.

- Es werden erst alle A mit $A \rightarrow \varepsilon$ in V' aufgenommen.
- Dann wird geprüft, ob neue Regeln $B \rightarrow \varepsilon$ entstehen, wenn man A in allen Regeln auf der rechten Seite A durch ε ersetzt.
- Ist dies der Fall, so werden die entsprechenden Variablen B in V' aufgenommen und genauso behandelt.
- Das Verfahren hört auf, wenn V' sich nicht mehr ändert.

Bemerkung:

- In Phase 1 werden noch keine Regeln geändert.
- Die Ersetzung ist also nur “testweise”.
- Am Ende enthält V' alle Variablen A mit $A \xrightarrow{*} \varepsilon$.

Ziel:

Es kommen keine Regeln $A \rightarrow \varepsilon$ vor.

Schritt 3

Vorgehen, Phase 1:

Finde die Menge V' aller Variablen A für die $A \xrightarrow{*} \varepsilon$ existiert.

Vorgehen, Phase 2:

Ersetzung der Regeln $A \rightarrow \varepsilon$.

- Gegeben V' aus Phase 1.
 - Streiche alle Regeln $A \rightarrow \varepsilon$.
 - Für $A \rightarrow BC$ füge die zusätzliche Regel
 - $A \rightarrow B$ falls $C \in V'$,
 - $A \rightarrow C$ falls $B \in V'$
- ein.
- (Die Regel $A \rightarrow BC$ wird nicht gestrichen).

Ziel:

Es kommen keine Regeln $A \rightarrow \varepsilon$ vor.

Schritt 3

Vorgehen, Phase 1:

- Initialisierung
 $V' = \{A \mid A \rightarrow \varepsilon\} = \{S, C\}$
- Erster Durchlauf liefert: A
 $\Rightarrow V' = \{A, C, S\}$
- Zweiter Durchlauf liefert: D
 $\Rightarrow V' = \{A, C, D, S\}$
- Dritter Durchlauf liefert: F
 $\Rightarrow V' = \{A, C, D, F, S\}$
- Vierter Durchlauf liefert nichts neues.

$$S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \mid \varepsilon$$

$$A \rightarrow C \mid a$$

$$B \rightarrow b$$

$$C \rightarrow AF \mid CC_3 \mid \varepsilon$$

$$D \rightarrow A \mid B \mid Y_a Y_b$$

$$E \rightarrow B$$

$$F \rightarrow D \mid E$$

$$C_1 \rightarrow AY_a$$

$$C_2 \rightarrow BY_b$$

$$C_3 \rightarrow DE$$

$$Y_a \rightarrow a$$

$$Y_b \rightarrow b$$

Ziel:

Es kommen keine Regeln $A \rightarrow \varepsilon$ vor.

Schritt 3

Vorgehen, Phase 2:

- Streiche ε -Produktionen
- Simuliere Ableitungen $A \xrightarrow{*} \varepsilon$ auf den verbleibenden Regeln
 $V' = \{A, C, D, F, S\}$

$$S \rightarrow A \mid Y_a C_1 \mid Y_b C_2$$

$$A \rightarrow C \mid a$$

$$B \rightarrow b$$

$$C \rightarrow AF \mid F \mid A \mid CC_3 \mid C_3$$

$$D \rightarrow A \mid B \mid Y_a Y_b$$

$$E \rightarrow B$$

$$F \rightarrow D \mid E$$

$$C_1 \rightarrow AY_a \mid Y_a$$

$$C_2 \rightarrow BY_b$$

$$C_3 \rightarrow DE \mid E$$

$$Y_a \rightarrow a$$

$$Y_b \rightarrow b$$

Ziel:

Es kommen keine Regeln $A \rightarrow \varepsilon$ vor.

Schritt 4

Beispiel:

$$A \rightarrow B \mid C$$
$$B \rightarrow C$$
$$C \rightarrow c$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Schritt 4

Vorgehen:

- **Phase 1:** Finde alle Kreise $A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_n \rightarrow A_1$.
Ersetze alle A_i durch A_1 .
- **Phase 2:** Betrachte die Regeln der Form $A \rightarrow B$
in **umgekehrt topologischer Sortierung**.
 - Für jede Regel $A \rightarrow B$ und jede Regel $B \rightarrow \beta$
füge Regel $A \rightarrow \beta$ hinzu.
 - Lösche Regel $A \rightarrow B$.

Ziel:

Die Grammatik enthält
keine (Ketten-)Regeln
der Form $A \rightarrow B$.

Schritt 4

Vorgehen:

- **Phase 1:** Finde alle Kreise $A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_n \rightarrow A_1$.
Ersetze alle A_i durch A_1 .
- **Phase 2:** Betrachte die Regeln der Form $A \rightarrow B$
in **umgekehrt topologischer Sortierung**.
 - Für jede Regel $A \rightarrow B$ und jede Regel $B \rightarrow \beta$
füge Regel $A \rightarrow \beta$ hinzu.
 - Lösche Regel $A \rightarrow B$.

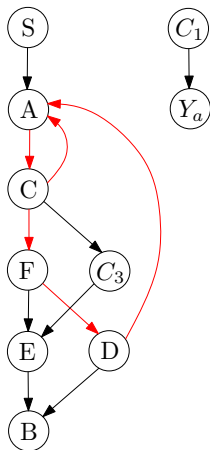
Topologische Sortierung der Regelmenge:

- V_1, \dots, V_k Menge von Variablen aus Kettenregeln
- V_1, \dots, V_k topologisch sortiert, wenn gilt:
- $V_i \xrightarrow{*} V_j \Rightarrow i < j$
- Voraussetzung: Es gibt keine zyklischen Abhängigkeiten

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Abhängigkeitsgraph



$$S \rightarrow A \mid Y_a C_1 \mid Y_b C_2$$

$$A \rightarrow C \mid a$$

$$B \rightarrow b$$

$$C \rightarrow AF \mid F \mid A \mid CC_3 \mid C_3$$

$$D \rightarrow A \mid B \mid Y_a Y_b$$

$$E \rightarrow B$$

$$F \rightarrow D \mid E$$

$$C_1 \rightarrow AY_a \mid Y_a$$

$$C_2 \rightarrow BY_b$$

$$C_3 \rightarrow DE \mid E$$

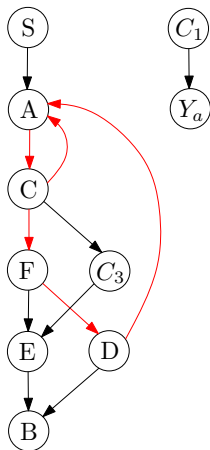
$$Y_a \rightarrow a$$

$$Y_b \rightarrow b$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Schritt 4 – Phase 1



- Zyklen $A \rightarrow C$ und $A \rightarrow C \rightarrow F \rightarrow D \rightarrow A$
- $\Rightarrow A, C, F, D$ äquivalent
- Entferne an Zyklen beteiligte Regeln
- Ersetze Vorkommen von C, F, D in allen Regeln durch A
- Lösche Regeln der Form $A \rightarrow A$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Schritt 4 – Phase 1

$S \rightarrow A \mid Y_a C_1 \mid Y_b C_2$ $A \rightarrow C \mid a$ $B \rightarrow b$ $C \rightarrow AF \mid F \mid A \mid CC_3 \mid C_3$ $D \rightarrow A \mid B \mid Y_a Y_b$ $E \rightarrow B$ $F \rightarrow D \mid E$ $C_1 \rightarrow AY_a \mid Y_a$ $C_2 \rightarrow BY_b$ $C_3 \rightarrow DE \mid E$ $Y_a \rightarrow a$ $Y_b \rightarrow b$	\Rightarrow	$S \rightarrow A \mid Y_a C_1 \mid Y_b C_2$ $A \rightarrow a$ $B \rightarrow b$ $A \rightarrow AA \mid AC_3 \mid C_3$ $A \rightarrow B \mid Y_a Y_b$ $E \rightarrow B$ $A \rightarrow E$ $C_1 \rightarrow AY_a \mid Y_a$ $C_2 \rightarrow BY_b$ $C_3 \rightarrow AE \mid E$ $Y_a \rightarrow a$ $Y_b \rightarrow b$
---	---------------	--

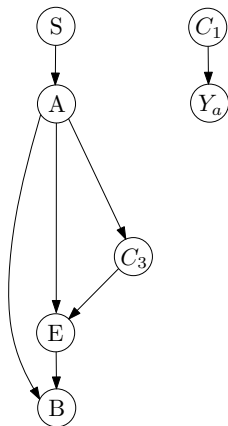
Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Zyklus:

$A \rightarrow C \rightarrow F \rightarrow D \rightarrow A$

Schritt 4 – Phase 2



$$\begin{aligned}
 S &\rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 A &\rightarrow a \mid AA \mid AC_3 \mid C_3 \mid B \mid Y_a Y_b \mid E \\
 B &\rightarrow b \\
 E &\rightarrow B \\
 C_1 &\rightarrow AY_a \mid Y_a \\
 C_2 &\rightarrow BY_b \\
 C_3 &\rightarrow AE \mid E \\
 Y_a &\rightarrow a \\
 Y_b &\rightarrow b
 \end{aligned}$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Topologische Sortierung: $S, A, C_3, E, B, C_1, Y_a$

Schritt 4 – Phase 2

- Ersetze $A \rightarrow B$ durch $A \rightarrow \beta$, falls Regel $B \rightarrow \beta$ existiert.

$$\begin{array}{l}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid \\
 \quad B \mid Y_a Y_b \mid E \\
 B \rightarrow b \\
 E \rightarrow B \\
 C_1 \rightarrow AY_a \mid Y_a \\
 C_2 \rightarrow BY_b \\
 C_3 \rightarrow AE \mid E \\
 Y_a \rightarrow a \\
 Y_b \rightarrow b
 \end{array}
 \quad \Rightarrow \quad
 \begin{array}{l}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid \\
 \quad B \mid Y_a Y_b \mid E \\
 B \rightarrow b \\
 E \rightarrow B \\
 C_1 \rightarrow AY_a \mid a \\
 C_2 \rightarrow BY_b \\
 C_3 \rightarrow AE \mid E \\
 Y_a \rightarrow a \\
 Y_b \rightarrow b
 \end{array}$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Topologische Sortierung:
 $S, A, C_3, E, B, C_1, Y_a$

Schritt 4 – Phase 2

- Ersetze $A \rightarrow B$ durch $A \rightarrow \beta$, falls Regel $B \rightarrow \beta$ existiert.

$$\begin{array}{l}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid \\
 \quad B \mid Y_a Y_b \mid E \\
 \\
 B \rightarrow b \\
 \\
 E \rightarrow B \\
 \\
 C_1 \rightarrow AY_a \mid Y_a \\
 \\
 C_2 \rightarrow BY_b \\
 \\
 C_3 \rightarrow AE \mid E \\
 \\
 Y_a \rightarrow a \\
 \\
 Y_b \rightarrow b
 \end{array}
 \quad \Rightarrow \quad
 \begin{array}{l}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid \\
 \quad B \mid Y_a Y_b \mid E \\
 \\
 B \rightarrow b \\
 \\
 E \rightarrow B \\
 \\
 C_1 \rightarrow AY_a \mid a \\
 \\
 C_2 \rightarrow BY_b \\
 \\
 C_3 \rightarrow AE \mid E \\
 \\
 Y_a \rightarrow a \\
 \\
 Y_b \rightarrow b
 \end{array}$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Topologische Sortierung:
 $S, A, C_3, E, B, C_1, Y_a$

Schritt 4 – Phase 2

- Ersetze $A \rightarrow B$ durch $A \rightarrow \beta$, falls Regel $B \rightarrow \beta$ existiert.

$$\begin{array}{lcl}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 & & S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid & & A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid \\
 \quad B \mid Y_a Y_b \mid E & & \quad b \mid Y_a Y_b \mid E \\
 \\
 B \rightarrow b & & B \rightarrow b \\
 \\
 E \rightarrow B & & E \rightarrow b \\
 \\
 C_1 \rightarrow AY_a \mid Y_a & \Rightarrow & C_1 \rightarrow AY_a \mid a \\
 \\
 C_2 \rightarrow BY_b & & C_2 \rightarrow BY_b \\
 \\
 C_3 \rightarrow AE \mid E & & C_3 \rightarrow AE \mid E \\
 \\
 Y_a \rightarrow a & & Y_a \rightarrow a \\
 \\
 Y_b \rightarrow b & & Y_b \rightarrow b
 \end{array}$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Topologische Sortierung:
 $S, A, C_3, E, B, C_1, Y_a$

Schritt 4 – Phase 2

- Ersetze $A \rightarrow B$ durch $A \rightarrow \beta$, falls Regel $B \rightarrow \beta$ existiert.

$$\begin{array}{lcl}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 & & S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid & & A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid \\
 \quad B \mid Y_a Y_b \mid E & & \quad b \mid Y_a Y_b \\
 \\
 B \rightarrow b & & B \rightarrow b \\
 \\
 E \rightarrow B & & E \rightarrow b \\
 \\
 C_1 \rightarrow AY_a \mid Y_a & \Rightarrow & C_1 \rightarrow AY_a \mid a \\
 \\
 C_2 \rightarrow BY_b & & C_2 \rightarrow BY_b \\
 \\
 C_3 \rightarrow AE \mid E & & C_3 \rightarrow AE \mid b \\
 \\
 Y_a \rightarrow a & & Y_a \rightarrow a \\
 \\
 Y_b \rightarrow b & & Y_b \rightarrow b
 \end{array}$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Topologische Sortierung:
 $S, A, C_3, E, B, C_1, Y_a$

Schritt 4 – Phase 2

- Ersetze $A \rightarrow B$ durch $A \rightarrow \beta$, falls Regel $B \rightarrow \beta$ existiert.

$$\begin{array}{l}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid \\
 \quad B \mid Y_a Y_b \mid E \\
 B \rightarrow b \\
 E \rightarrow B \\
 C_1 \rightarrow AY_a \mid Y_a \\
 C_2 \rightarrow BY_b \\
 C_3 \rightarrow AE \mid E \\
 Y_a \rightarrow a \\
 Y_b \rightarrow b
 \end{array}
 \quad \Rightarrow \quad
 \begin{array}{l}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 \\
 A \rightarrow a \mid AA \mid AC_3 \mid AE \mid \\
 \quad b \mid Y_a Y_b \\
 B \rightarrow b \\
 E \rightarrow b \\
 C_1 \rightarrow AY_a \mid a \\
 C_2 \rightarrow BY_b \\
 C_3 \rightarrow AE \mid b \\
 Y_a \rightarrow a \\
 Y_b \rightarrow b
 \end{array}$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Topologische Sortierung:

$S, A, C_3, E, B, C_1, Y_a$

Schritt 4 – Phase 2

- Ersetze $A \rightarrow B$ durch $A \rightarrow \beta$, falls Regel $B \rightarrow \beta$ existiert.

$$\begin{array}{l}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid \\
 \quad B \mid Y_a Y_b \mid E \\
 B \rightarrow b \\
 E \rightarrow B \\
 C_1 \rightarrow AY_a \mid Y_a \\
 C_2 \rightarrow BY_b \\
 C_3 \rightarrow AE \mid E \\
 Y_a \rightarrow a \\
 Y_b \rightarrow b
 \end{array}
 \quad \Rightarrow \quad
 \begin{array}{l}
 S \rightarrow Y_a C_1 \mid Y_b C_2 \mid a \mid AA \\
 \quad AC_3 \mid AE \mid b \mid Y_a Y_b \\
 A \rightarrow a \mid AA \mid AC_3 \mid AE \mid \\
 \quad b \mid Y_a Y_b \\
 B \rightarrow b \\
 E \rightarrow b \\
 C_1 \rightarrow AY_a \mid a \\
 C_2 \rightarrow BY_b \\
 C_3 \rightarrow AE \mid b \\
 Y_a \rightarrow a \\
 Y_b \rightarrow b
 \end{array}$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Topologische Sortierung:

$S, A, C_3, E, B, C_1, Y_a$

Schritt 4 – Phase 2

- Ersetze $A \rightarrow B$ durch $A \rightarrow \beta$, falls Regel $B \rightarrow \beta$ existiert.

$$\begin{array}{lcl}
 S \rightarrow A \mid Y_a C_1 \mid Y_b C_2 & & S \rightarrow Y_a C_1 \mid Y_b C_2 \mid a \mid AA \\
 & & AC_3 \mid AE \mid b \mid Y_a Y_b \\
 A \rightarrow a \mid AA \mid AC_3 \mid C_3 \mid & & A \rightarrow a \mid AA \mid AC_3 \mid AE \mid \\
 & & B \mid Y_a Y_b \mid E \\
 B \rightarrow b & & B \rightarrow b \\
 E \rightarrow B & & E \rightarrow b \\
 C_1 \rightarrow AY_a \mid Y_a & \Rightarrow & C_1 \rightarrow AY_a \mid a \\
 C_2 \rightarrow BY_b & & C_2 \rightarrow BY_b \\
 C_3 \rightarrow AE \mid E & & C_3 \rightarrow AE \mid b \\
 Y_a \rightarrow a & & Y_a \rightarrow a \\
 Y_b \rightarrow b & & Y_b \rightarrow b
 \end{array}$$

Ziel:

Die Grammatik enthält keine (Ketten-)Regeln der Form $A \rightarrow B$.

Topologische Sortierung:

$S, A, C_3, E, B, C_1, Y_a$

Sonderbehandlung $\varepsilon \in L(G)$

Sonderbehandlung von ε -Produktionen

- Die konstruierte Grammatik ist nun in Chomsky-Normalform, aber
- falls in der ursprünglichen Grammatik G eine Ableitungsfolge $S \xrightarrow{*} \varepsilon$ existierte, haben wir diese entfernt.
- Erweitere dann Grammatik um Regeln $S' \rightarrow S$ und $S' \rightarrow \varepsilon$ für neues Startsymbol S' .

Heute

- Syntaxbäume
- Eindeutigkeit
- Chomsky-Normalform
- CYK-Algorithmus

Der CYK-Algorithmus

Satz.

Es gibt einen Algorithmus (den Cocke-Younger-Kasami Algorithmus), der für eine kontextfreie Grammatik G in Chomsky-Normalform und ein Wort $w \in \Sigma^*$ in Zeit $O(|R| \cdot n^3)$ entscheidet, ob $w \in L(G)$, wobei $n = |w|$ und $|R|$ die Anzahl der Regeln von G ist.

Beweis – Beschreibung des CYK-Algorithmus

- Sei $w = w_1 \cdots w_n$.
- Sei $V_{ij} \subseteq V$ so dass $A \in V_{ij}$ genau dann, wenn $A \xrightarrow{*} w_i \cdots w_j$.
- Für alle $1 \leq i \leq j \leq n$ berechne die Menge $V_{ij} \subseteq V$.
- Dann ist $w \in L(G)$ genau dann, wenn $S \in V_{1n}$ ist.
- Die Tabelle der V_{ij} wird nach wachsendem $\ell := j - i$ aufgebaut, beginnend mit $\ell = 0$.
- Für $j - i = \ell > 0$ wird die Berechnung von V_{ij} systematisch auf zuvor berechnete V_{ik}, V_{k+1j} mit $i \leq k < j$ zurückgeführt.

Bemerkung

- Wir benutzen dynamische Programmierung.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y_1 C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_U S$
 $C_2 \rightarrow SY_1$
 $Y_* \rightarrow *$
 $Y_U \rightarrow U$
 $Y_1 \rightarrow ($
 $Y_2 \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	V_{13}	V_{24}	V_{35}	V_{46}	V_{57}	V_{68}		
V_{ii+1}	V_{12}	V_{23}	V_{34}	V_{45}	V_{56}	V_{67}	V_{78}	
V_{ii}	V_{11}	V_{22}	V_{33}	V_{44}	V_{55}	V_{66}	V_{77}	V_{88}
	1	*	U	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y(C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_U S$
 $C_2 \rightarrow SY_)$
 $Y_* \rightarrow *$
 $Y_U \rightarrow U$
 $Y_(\rightarrow ($
 $Y_)\rightarrow)$

V_{ii+7}	V_1							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	V_{13}	V_{24}	V_{35}	V_{46}	V_{57}	V_{68}		
V_{ii+1}	V_{12}	V_{23}	V_{34}	V_{45}	V_{56}	V_{67}	V_{78}	
V_{ii}	V_{11}	V_{22}	V_{33}	V_{44}	V_{55}	V_{66}	V_{77}	V_{88}
	1	*	U	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Vorgehen

- Sei $V_{ij} \subseteq V$ so dass $A \in V_{ij}$ genau dann, wenn $A \xrightarrow{*} w_i \cdots w_j$.
- Die Tabelle der V_{ij} wird nach wachsendem $\ell := j - i$ aufgebaut, beginnend mit $\ell = 0$.
- Für $j - i = \ell > 0$ wird die Berechnung von V_{ij} systematisch auf zuvor berechnete V_{ik}, V_{k+1j} mit $i \leq k < j$ zurückgeführt.

Fall $\ell = 0$:

- Konstruiere die Mengen V_{ij} , d.h. alle $A \in V$ mit $A \xrightarrow{*} w_j$.
- Da G in Chomsky-Normalform ist, gilt $A \xrightarrow{*} w_j$ nur, wenn $(A \rightarrow w_j) \in R$.
- Die Berechnung von V_{ij} ist für alle $i \in \{1, \dots, n\}$ in $O(|R|)$ möglich.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y_1 C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_U S$
 $C_2 \rightarrow SY_1$
 $Y_* \rightarrow *$
 $Y_U \rightarrow U$
 $Y_1 \rightarrow ($
 $Y_2 \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	V_{13}	V_{24}	V_{35}	V_{46}	V_{57}	V_{68}		
V_{ii+1}	V_{12}	V_{23}	V_{34}	V_{45}	V_{56}	V_{67}	V_{78}	
V_{ii}	V_{11}	V_{22}	V_{33}	V_{44}	V_{55}	V_{66}	V_{77}	V_{88}
	1	*	U	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y(C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_\cup S$
 $C_2 \rightarrow SY_)$
 $Y_* \rightarrow *$
 $Y_\cup \rightarrow \cup$
 $Y_(\rightarrow ($
 $Y_) \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	V_{13}	V_{24}	V_{35}	V_{46}	V_{57}	V_{68}		
V_{ii+1}	V_{12}	V_{23}	V_{34}	V_{45}	V_{56}	V_{67}	V_{78}	
V_{ii}	S	Y_*	Y_\cup	$Y_($	S	S	$Y_)$	Y_*
	↓	↓	↓	↓	↓	↓	↓	↓
	1	*	\cup	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Vorgehen

- Sei $V_{ij} \subseteq V$ so dass $A \in V_{ij}$ genau dann, wenn $A \xrightarrow{*} w_i \cdots w_j$.
- Die Tabelle der V_{ij} wird nach wachsendem $\ell := j - i$ aufgebaut, beginnend mit $\ell = 0$.
- Für $j - i = \ell > 0$ wird die Berechnung von V_{ij} systematisch auf zuvor berechnete V_{ik}, V_{k+1j} mit $i \leq k < j$ zurückgeführt.

Fall $\ell > 0$:

- Jede Ableitung von $w_i \cdots w_j$ muss mit einer Regel der Form

$$A \rightarrow BC$$

beginnen, wobei ein $k \in \{i, \dots, j - 1\}$ existiert mit

- $B \xrightarrow{*} w_i \cdots w_k$ und
- $C \xrightarrow{*} w_{k+1} \cdots w_j$.

CYK-Algorithmus – Vorgehen

Verfahren

- Speichere alle Mengen V_{rs} als Arrays der Länge $|V|$, in denen für jedes $A \in V$ markiert ist, ob $A \in V_{rs}$.
- **Berechnung von V_{ij} :**
Überprüfe für jede Regel $(A \rightarrow BC) \in R$ und jedes k , ob

$$B \xrightarrow{*} w_i \cdots w_k$$

$$C \xrightarrow{*} w_{k+1} \cdots w_j$$

durch Ansehen der Stelle

- B im Array zu V_{ik} und
- C im Array zu $V_{k+1 j}$.

CYK-Algorithmus – Vorgehen

Verfahren

- Speichere alle Mengen V_{rs} als Arrays der Länge $|V|$, in denen für jedes $A \in V$ markiert ist, ob $A \in V_{rs}$.
- **Berechnung von V_{ij} :**
Überprüfe für jede Regel $(A \rightarrow BC) \in R$ und jedes k , ob

$$B \xrightarrow{*} w_i \cdots w_k$$
$$C \xrightarrow{*} w_{k+1} \cdots w_j$$

durch Ansehen der Stelle

- B im Array zu V_{ik} und
- C im Array zu $V_{k+1 j}$.

Bemerkung

- Dies benötigt Aufwand in $O(n \cdot |R|)$

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y_1 C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_U S$
 $C_2 \rightarrow SY_1$
 $Y_* \rightarrow *$
 $Y_U \rightarrow U$
 $Y_1 \rightarrow ($
 $Y_2 \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	V_{13}	V_{24}	V_{35}	V_{46}	V_{57}	V_{68}		
V_{ii+1}	V_{12}	V_{23}	V_{34}	V_{45}	V_{56}	V_{67}	V_{78}	
V_{ii}	S	Y_*	Y_U	Y_1	S	S	Y_2	Y_*
	↓	↓	↓	↓	↓	↓	↓	↓
	1	*	U	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y(C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_\cup S$
 $C_2 \rightarrow SY_)$
 $Y_* \rightarrow *$
 $Y_\cup \rightarrow \cup$
 $Y(\rightarrow ($
 $Y) \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	V_{13}	V_{24}	V_{35}	V_{46}	V_{57}	V_{68}		
V_{ii+1}	S	\emptyset	\emptyset	\emptyset	S	C_2	\emptyset	
V_{ii}	S	Y_*	Y_\cup	$Y($	S	S	$Y)$	Y_*
	1	*	\cup	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y(C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_\cup S$
 $C_2 \rightarrow SY_)$
 $Y_* \rightarrow *$
 $Y_\cup \rightarrow \cup$
 $Y(\rightarrow ($
 $Y) \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	V_{14}	V_{25}	V_{36}	V_{47}	V_{58}			
V_{ii+2}	\emptyset	\emptyset	\emptyset	\emptyset	C_2	\emptyset		
V_{ii+1}	S	\emptyset	\emptyset	\emptyset	S	C_2	\emptyset	
V_{ii}	S	Y_*	Y_\cup	$Y($	S	S	$Y)$	Y_*
	1	*	\cup	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y(C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_\cup S$
 $C_2 \rightarrow SY_)$
 $Y_* \rightarrow *$
 $Y_\cup \rightarrow \cup$
 $Y(\rightarrow ($
 $Y) \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	V_{15}	V_{26}	V_{37}	V_{48}				
V_{ii+3}	\emptyset	\emptyset	\emptyset	S	\emptyset			
V_{ii+2}	\emptyset	\emptyset	\emptyset	\emptyset	C_2	\emptyset		
V_{ii+1}	S	\emptyset	\emptyset	\emptyset	S	C_2	\emptyset	
V_{ii}	S	Y_*	Y_\cup	$Y($	S	S	$Y)$	Y_*
	1	*	\cup	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y(C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_{\cup} S$
 $C_2 \rightarrow SY_{)}$
 $Y_* \rightarrow *$
 $Y_{\cup} \rightarrow \cup$
 $Y_{(} \rightarrow ($
 $Y_{)} \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	V_{16}	V_{27}	V_{38}					
V_{ii+4}	\emptyset	\emptyset	C_1	S				
V_{ii+3}	\emptyset	\emptyset	\emptyset	S	\emptyset			
V_{ii+2}	\emptyset	\emptyset	\emptyset	\emptyset	C_2	\emptyset		
V_{ii+1}	S	\emptyset	\emptyset	\emptyset	S	C_2	\emptyset	
V_{ii}	S	Y_*	Y_{\cup}	$Y_{(}$	S	S	$Y_{)}$	Y_*
	1	*	\cup	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y(C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_{\cup} S$
 $C_2 \rightarrow SY_{\cup}$
 $Y_* \rightarrow *$
 $Y_{\cup} \rightarrow \cup$
 $Y_{\left(} \rightarrow (\right.$
 $Y_{\right)} \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	V_{17}	V_{28}						
V_{ii+5}	\emptyset	\emptyset	\emptyset					
V_{ii+4}	\emptyset	\emptyset	C_1	S				
V_{ii+3}	\emptyset	\emptyset	\emptyset	S	\emptyset			
V_{ii+2}	\emptyset	\emptyset	\emptyset	\emptyset	C_2	\emptyset		
V_{ii+1}	S	\emptyset	\emptyset	\emptyset	S	C_2	\emptyset	
V_{ii}	S	Y_*	Y_{\cup}	$Y_{\left(}$	S	S	$Y_{\right)}$	Y_*
	1	*	\cup	(0	1)	*
i	1	2	3	4	5	6	7	8

Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y(C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_{\cup} S$
 $C_2 \rightarrow SY_{\cup}$
 $Y_* \rightarrow *$
 $Y_{\cup} \rightarrow \cup$
 $Y_{(} \rightarrow ($
 $Y_{)} \rightarrow)$

V_{ii+7}	V_{18}							
V_{ii+6}	S	\emptyset						
V_{ii+5}	\emptyset	\emptyset	\emptyset					
V_{ii+4}	\emptyset	\emptyset	C_1	S				
V_{ii+3}	\emptyset	\emptyset	\emptyset	S	\emptyset			
V_{ii+2}	\emptyset	\emptyset	\emptyset	\emptyset	C_2	\emptyset		
V_{ii+1}	S	\emptyset	\emptyset	\emptyset	S	C_2	\emptyset	
V_{ii}	S	Y_*	Y_{\cup}	$Y_{(}$	S	S	$Y_{)}$	Y_*
	1	*	\cup	(0	1)	*
i	1	2	3	4	5	6	7	8

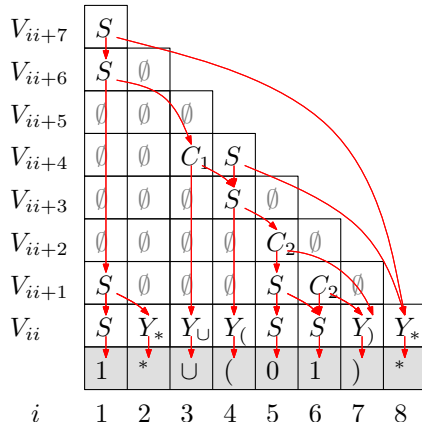
Wir testen, ob das Wort

$$w = 1^* \cup (01)^*$$

von der Grammatik erzeugt werden kann.

CYK-Algorithmus – Beispiel

$S \rightarrow SY_* \mid SC_1 \mid$
 $SS \mid Y_1 C_2 \mid$
 $e \mid \emptyset \mid 0 \mid 1$
 $C_1 \rightarrow Y_\cup S$
 $C_2 \rightarrow SY_)$
 $Y_* \rightarrow *$
 $Y_\cup \rightarrow \cup$
 $Y_1 \rightarrow ($
 $Y_2 \rightarrow)$



Wir testen, ob das Wort
 $w = 1^* \cup (01)^*$
 von der Grammatik
 erzeugt werden kann.

Ergebnisse zum Wortproblem

- **Typ-0 Grammatik.**
Das Wortproblem ist nicht entscheidbar.
- **Typ-1 Grammatik.**
Das Wortproblem ist \mathcal{NP} -vollständig.
- **Typ-2 Grammatik.**
Das Wortproblem ist in polynomieller Zeit lösbar.
- **Typ-3 Grammatik.**
Das Wortproblem ist in linearer Zeit lösbar.