Automaten und Formale Sprachen
10. Vorlesung

Prof. Dr. Dietrich Kuske

FG Automaten und Logik, TU Ilmenau

Wintersemester 2022/23

Chomsky-Normalform

Definition

Eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ist in Chomsky-Normalform, falls

- alle Produktionen von G die Form

  $$A \to BC \text{ oder } A \to a \; (A, B, C \in V,\ a \in \Sigma)$$

  haben
- oder alle Produktionen von G die Form

  $$A \to BC \text{ oder } A \to a \text{ oder } S \to \varepsilon \; (A, B, C \in V,\ a \in \Sigma)$$

  haben und S nie auf der rechten Seite einer Produktion vorkommt.

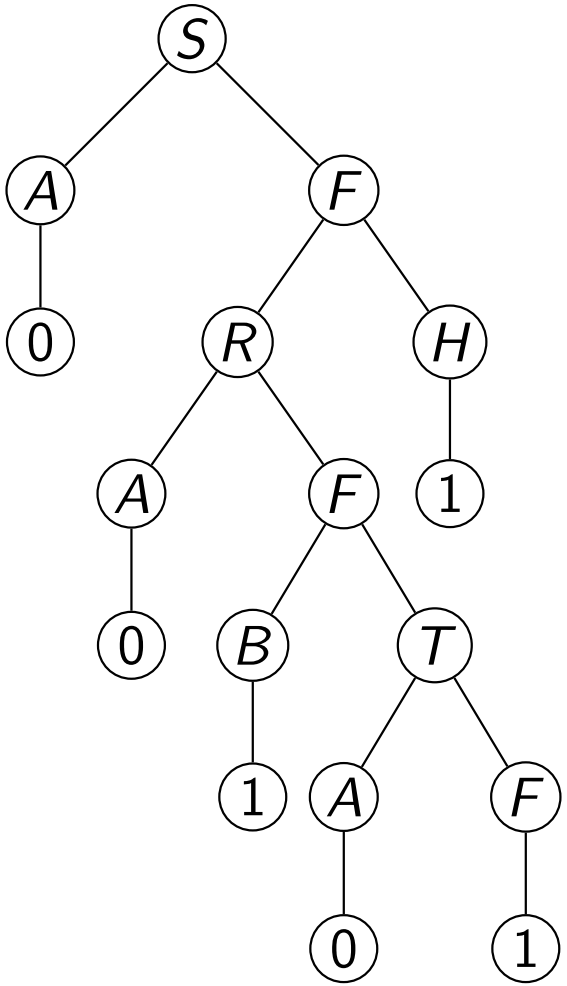
Betrachte die Grammatik mit $\Sigma = \{0, 1\}$ und den folgenden Produktionen:

$$\begin{array}{lcllcl} S & \to & \varepsilon \mid AF & R & \to & AF \\ F & \to & BR \mid RH \mid 1 & H & \to & BR \mid 1 \\ A & \to & 0 & B & \to & 1 \end{array}$$

Sie ist in Chomsky-Normalform und hat u.a. die Linksableitungen

$$\begin{array}{rcl} S & \Rightarrow & \varepsilon \text{ und} \\ \underline{S} & \Rightarrow & \underline{A}F \Rightarrow 0\underline{F} \Rightarrow 0\underline{R}H \Rightarrow 0\underline{A}FH \Rightarrow 00\underline{F}H \Rightarrow 00\underline{B}RH \\ & \Rightarrow & 001\underline{R}H \Rightarrow 001\underline{A}FH \Rightarrow 0010\underline{F}H \Rightarrow 00101\underline{H} \Rightarrow 001011 \end{array}$$

Die zweite Linksableitung hat den folgenden Ableitungsbaum:



Beobachtung

Sei G in Chomsky-Normalform und T ein Ableitungsbaum eines Wortes $w \in \Sigma^+$ der Länge n. Dann gilt

- jeder innere Knoten des Ableitungsbaums hat genau 2 mit Nichtterminalen beschriftete Kinder oder genau ein mit einem Terminal beschriftetes Kind und
- es gibt n Blätter.

Also hat T genau $3n - 1$ viele Knoten.

(Folie 9.9: keine ähnliche Aussage kann für alle kontext-freien Grammatiken gemacht werden.)

Satz

Zu jeder kontextfreien Grammatik G gibt es eine Grammatik G' in Chomsky-Normalform mit $L(G) = L(G')$.

Beweis:
Schritt 1: ε-Behandlung

Lemma auf Folie 9.23: o.E. ist G auch kontext-sensitiv, d.h.

- G hat keine Produktionen der Form $A \to \varepsilon$ mit $A \neq S$
- und nur dann die Regel $S \to \varepsilon$, wenn S auf keiner rechten Seite vorkommt.

Schritt 2: Vereinzeln der Terminale auf rechten Seiten

- Führe für jedes $a \in \Sigma$ ein neues Nichtterminal $A_a \notin V$ ein.
- Ersetze jedes Vorkommen von a in einer rechten Seite durch A_a.
- Führe neue Produktion $A_a \to a$ für $a \in \Sigma$ ein.

$\rightsquigarrow$ Die Sprache der Grammatik bleibt unverändert und alle Produktionen haben die Form

- $A \to a$ mit $a \in \Sigma$ oder
- $A \to A_1 \cdots A_n$ mit $n \geq 1$ und $A, A_1, \ldots, A_n \in V$ oder
- $S \to \varepsilon$ und S kommt auf keiner rechten Seite vor.

Schritt 3: Elimination von Produktionen $A \to A_1 \cdots A_n$ mit $n \geq 3$.

Sei $A \to A_1 \cdots A_n$ eine Produktion mit $n \geq 3$.

Wir führen neue Nichtterminale $B_2, \ldots, B_{n-1}$ ein und ersetzen die Produktion $A \to A_1 \cdots A_n$ durch die folgenden Produktionen:

$$A \to A_1 B_2, \quad B_i \to A_i B_{i+1} \ (2 \leq i \leq n-2), \quad B_{n-1} \to A_{n-1} A_n$$

$\leadsto$ Die Sprache der Grammatik bleibt unverändert und alle Produktionen haben die Form

- $A \to a$ mit $a \in \Sigma$ oder
- $A \to BC$ oder $A \to B$ mit $A, B, C \in V$ oder
- $S \to \varepsilon$ und S kommt auf keiner rechten Seite vor.

Schritt 4: Elimination von Kettenregeln (d.h. Regeln der Form $A \to B$ mit $A, B \in V$).

Kommt eine Kettenregel $A \to B$ mit $A, B \in V$ vor, so ersetze sie durch die Menge aller Regeln $A \to \alpha$ mit $B \to \alpha$ und $\alpha \notin V$. Wiederhole dies, bis keine Kettenregel mehr vorhanden ist.

$\rightsquigarrow$ Die Sprache der Grammatik bleibt unverändert und alle Produktionen haben die Form

- $A \to a$ mit $a \in \Sigma$ oder
- $A \to BC$ mit $A, B, C \in V$ oder
- $S \to \varepsilon$ und S kommt auf keiner rechten Seite vor.

Beispiel: Sei

$$G = \big(\{S, A\}, \{a, b, c\}, P, S\big)$$

mit folgender Produktionenmenge P:

$$S \to aAb \mid ab \mid A \text{ und } A \to S \mid aaSc$$

Wir wandeln G in Chomsky-Normalform um.

Schritt 1: Wir machen G auch kontext-sensitiv (was nach dem Lemma auf Folie 9.23 immer möglich ist, hier aber nichts ändert, da G schon kontext-sensitiv ist).

Ergebnis: $S \to aAb \mid ab \mid A$
$A \to S \mid aaSc$

Schritt 2: Vereinzeln der Terminale

Ergebnis:
$$
\begin{aligned}
S &\to A_a A A_b \mid A_a A_b \mid A \\
A &\to S \mid A_a A_a S A_c \\
A_a &\to a \\
A_b &\to b \\
A_c &\to c
\end{aligned}
$$

Schritt 3: Elimination von Produktionen $A \to A_1 \cdots A_n$ mit $n \geq 3$.

Ergebnis:

$$
\begin{array}{lcl}
S & \to & A_a B \mid A_a A_b \mid A \\
B & \to & A A_b \\
A & \to & S \mid A_a A_a S A_c \\
A_a & \to & a \\
A_b & \to & b \\
A_c & \to & c
\end{array}
\qquad
\begin{array}{lcl}
S & \to & A_a B \mid A_a A_b \mid A \\
B & \to & A A_b \\
A & \to & S \mid A_a C \\
C & \to & A_a D \\
D & \to & S A_c \\
A_a & \to & a \\
A_b & \to & b \\
A_c & \to & c
\end{array}
$$

Schritt 4: Elimination von Kettenregeln.

Unsere Grammatik hat die Kettenregeln $S \to A$ und $A \to S$, die jetzt nacheinander ersetzt werden müssen.

Ergebnis:

$$
\begin{array}{lcl}
S & \to & A_aB \mid A_aA_b \mid A_aC \\
B & \to & AA_b \\
A & \to & S \mid A_aC \\
C & \to & A_aD \\
D & \to & SA_c \\
A_a & \to & a \\
A_b & \to & b \\
A_c & \to & c
\end{array}
\qquad
\begin{array}{lcl}
S & \to & A_aB \mid A_aA_b \mid A_aC \\
B & \to & AA_b \\
A & \to & A_aB \mid A_aA_b \mid A_aC \mid A_aC \\
C & \to & A_aD \\
D & \to & SA_c \\
A_a & \to & a \\
A_b & \to & b \\
A_c & \to & c
\end{array}
$$

Diese Grammatik ist in Chomsky-Normalform und äquivalent zur Ausgangsgrammatik.

Der Cocke-Younger-Kasami- oder CYK-Algorithmus

Sei G kontextfreie Grammatik.
Gesucht ist ein Algorithmus, mit dessen Hilfe wir entscheiden können, ob ein gegebenes Wort zu $L(G)$ gehört.

1. Versuch: Nach Folie 9.30 gibt es einen solchen Algorithmus.

Erinnerung: Nach Folie 9.23 können wir annehmen, daß G auch kontextsensitiv ist. Dann werden diejenigen Wörter berechnet, die sich in $\leq |w|$ vielen Schritten ableiten lassen, und getestet, ob w darunter ist (siehe Beweis Folie 2.24).

Dieses Verfahren hat also exponentielle Laufzeit.

heutiges Ziel: polynomieller Algorithmus

Voraussetzung: Die Grammatik ist in Chomsky-Normalform, alle Produktionen haben also die Form $A \to a$ oder $A \to BC$ (die Regel $S \to \varepsilon$ kann vernachlässigt werden, da sie ja nur bei der Ableitung des leeren Wortes eine Rolle spielt).

Idee: Gegeben sei ein Wort $w \in \Sigma^+$. Wir wollen feststellen, aus welchen Nichtterminalen es abgeleitet werden kann.

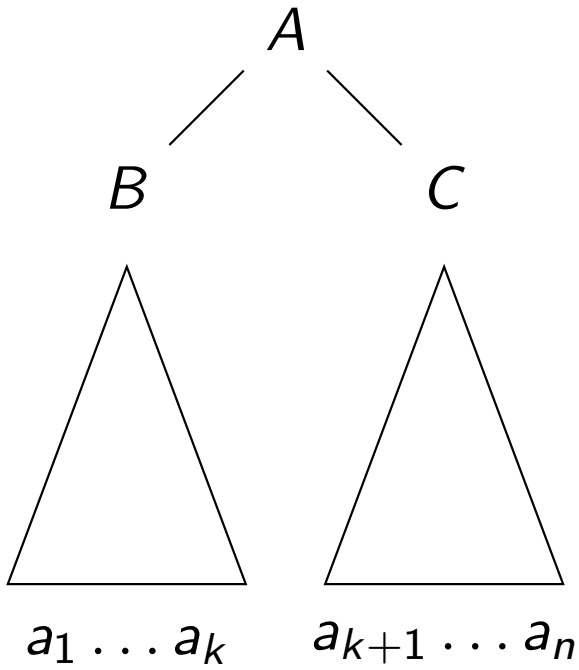
- **Möglichkeit 1:** $w = a \in \Sigma$, d.h., w besteht aus einem einzigen Alphabetsymbol.

  Dann kann w nur aus denjenigen Nichtterminalen A abgeleitet werden, für die es eine Produktion $A \to a$ gibt.

- **Möglichkeit 2:** $w = a_1 \cdots a_n$ mit $n \geq 2$.

  In diesem Fall gilt: Zunächst muß eine Produktion $A \to BC$ angewandt werden, dann muß ein Teil $a_1 \cdots a_k$ des Wortes aus B und der andere Teil $a_{k+1} \cdots a_n$ aus C abgeleitet werden (für ein $1 \leq k < n$).

Im 2. Fall sieht der Ableitungsbaum folgendermaßen aus:



Es ist jedoch nicht klar, wo das Wort w geteilt werden muß, d.h., wie groß die Position k ist!

Daher: Probiere alle möglichen k's durch. Das heißt:

Gegeben ein Wort $w = a_1 \cdots a_n$.

Für alle k mit $1 \leq k < n$ mache folgendes:

- Bestimme die Menge $V_{1,k}$ der Nichtterminale, aus denen sich $a_1 \cdots a_k$ ableiten läßt.
- Bestimme die Menge $V_{k+1,n-k}$ der Nichtterminale, aus denen sich $a_{k+1} \cdots a_n$ ableiten läßt.
- Stelle fest, ob es Nichtterminale A, B, C gibt mit $(A \to BC) \in P$, $B \in V_{1,k}$ und $C \in V_{k+1,n-k}$.
  In diesem Fall läßt sich w aus A ableiten.

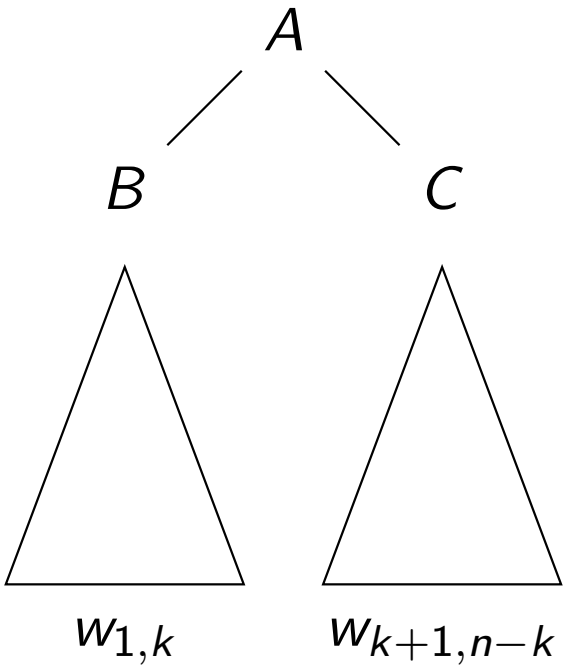
Um Mehraufwand zu vermeiden, verwenden wir die Methode der dynamischen Programmierung, das heißt:

- berechne zuerst alle Nichtterminale, aus denen sich Faktoren der Länge 1 ableiten lassen,
- berechne dann alle Nichtterminale, aus denen sich Faktoren der Länge 2 ableiten lassen,
- $\vdots$
- zuletzt berechne alle Nichtterminale, aus denen sich w ableiten läßt. Das Wort w liegt genau dann in der von der Grammatik erzeugten Sprache, wenn S sich unter diesen Nichtterminalen befindet.

Notation: Wir bezeichnen mit $w_{i,j}$ den Faktor von w, der an der Stelle i beginnt und die Länge j hat:

$$w = a_1 \cdots a_n \quad \rightsquigarrow \quad w_{i,j} = a_i \cdots a_{i+j-1}$$

Damit sieht das obige Bild folgendermaßen aus:



Wir bezeichnen mit $V_{i,j}$ die Menge aller Nichtterminale, aus denen sich $w_{i,j}$ ableiten läßt:

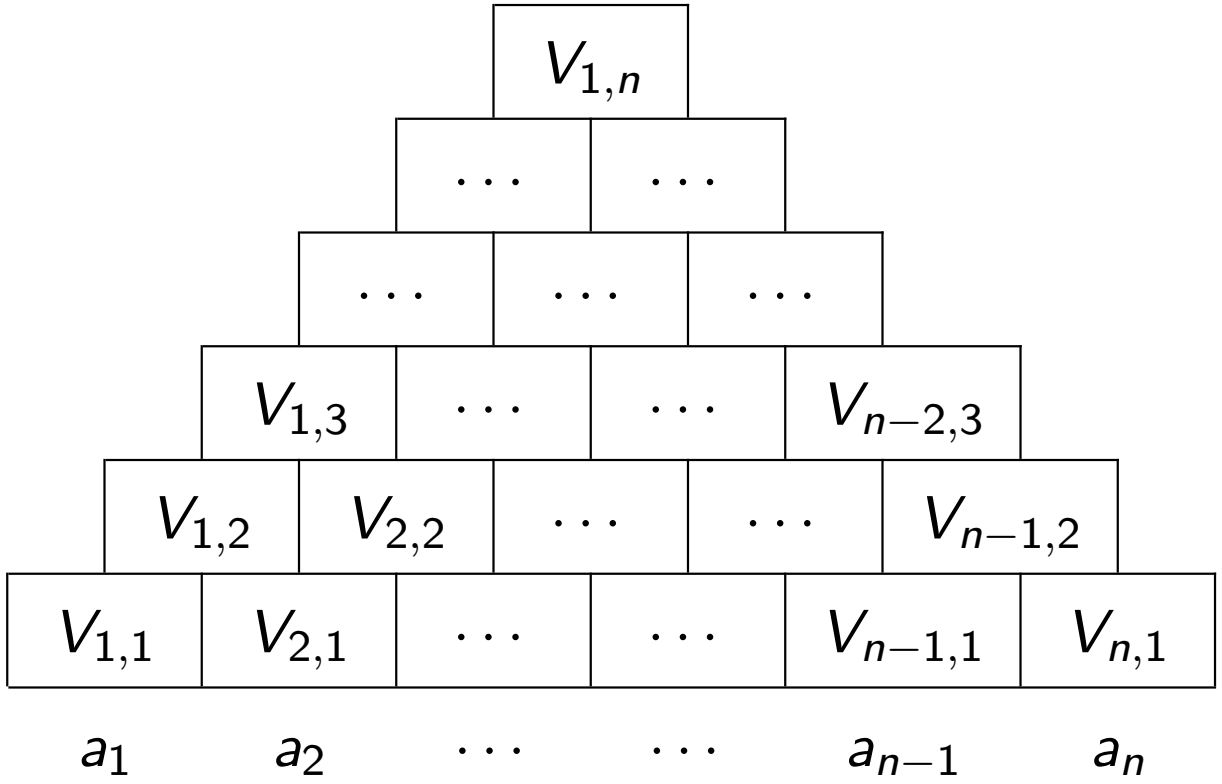
$$V_{i,j} = \{A \in V \mid A \Rightarrow_G^* w_{i,j}\}$$

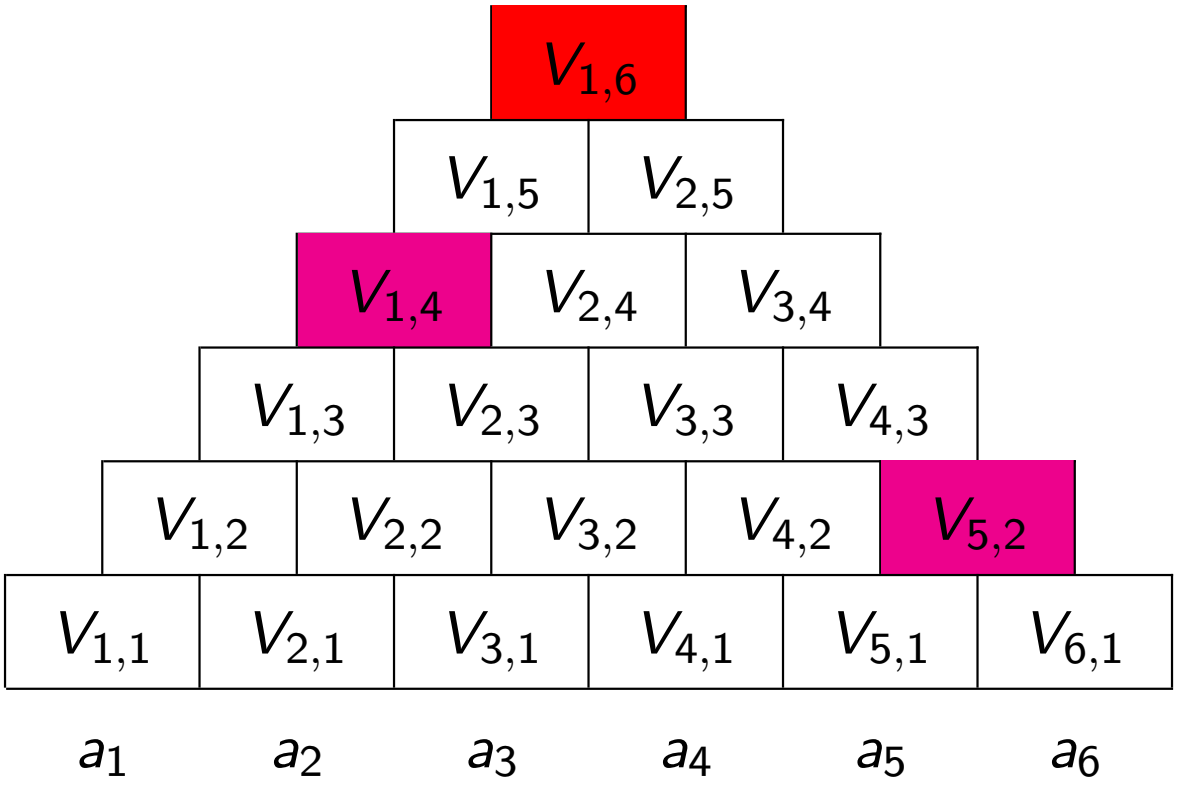
$V_{i,j}$ lässt sich aus Mengen $V_{\ell,k}$ mit $k < j$ folgendermaßen bestimmen:

$$V_{i,j} = \left\{A \;\middle|\; \exists (A \to BC) \in P \; \exists k < j : B \in V_{i,k} \text{ und } C \in V_{i+k,j-k}\right\}$$

Praktische Ausführung des CYK-Algorithmus:

Wir tragen die Mengen $V_{i,j}$ von Nichtterminalen zeilenweise in folgende Tabelle ein:



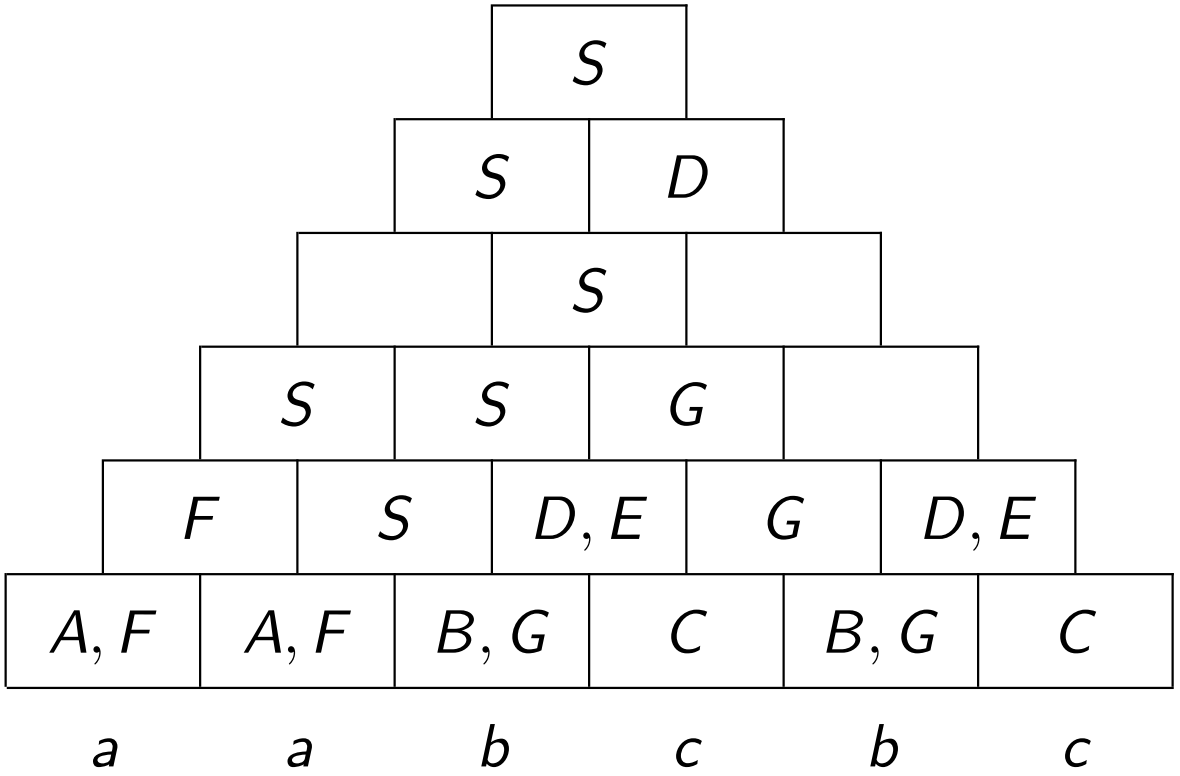


Wenn es $(A \to BC) \in P$ gibt mit $B \in V_{1,4}$ und $C \in V_{5,2}$,
so gilt $A \Rightarrow BC \Rightarrow^* a_1 \dots a_4\, C \Rightarrow^* a_1 \dots a_6$.
Füge also A zu $V_{1,6}$ hinzu.

Beispiel: Betrachte die Grammatik G mit folgenden Produktionen:

$$\begin{aligned} S &\to AD \mid FG \\ D &\to SE \mid BC \\ E &\to BC \\ F &\to AF \mid a \\ G &\to BG \mid CG \mid b \\ A &\to a \\ B &\to b \\ C &\to c \end{aligned}$$

Frage: Gilt $aabcbc \in L(G)$?


S
S D
S
S S G
F S D, E G D, E
A, F A, F B, G C B, G C
a a b c b c

Beispiel: Betrachte die Grammatik G mit den folgenden Produktionen:

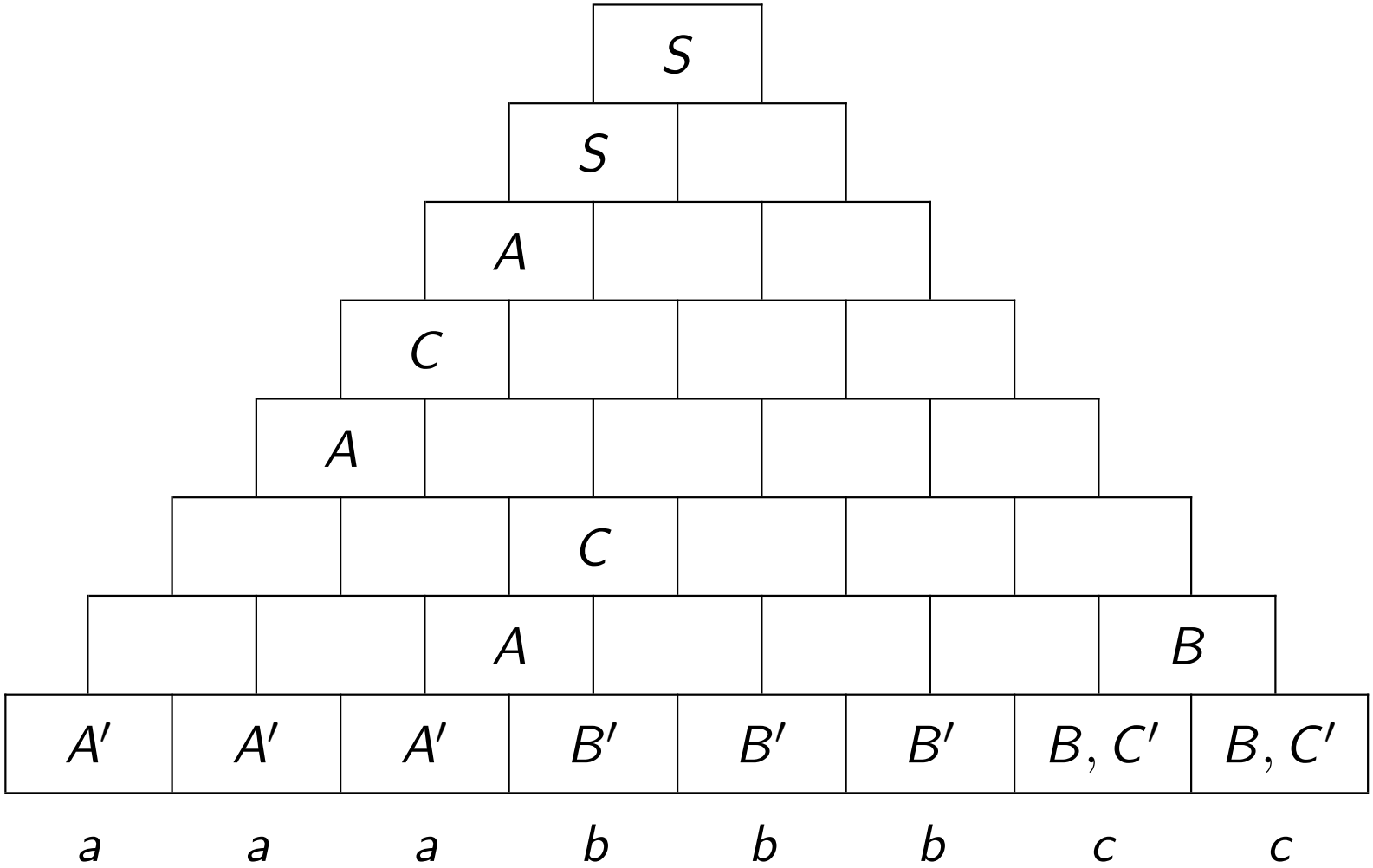
$$\begin{aligned} S &\to AB \\ A &\to ab \mid aAb \\ B &\to c \mid cB \end{aligned}$$

Frage: Gilt $aaabbbcc \in L(G)$?

Vorbereitung: Umwandlung der Grammatik in Chomsky-Normalform:

$$\begin{aligned} S &\to AB \\ A &\to A'B' \mid A'C \\ C &\to AB' \\ B &\to c \mid C'B \end{aligned}$$

$$\begin{aligned} A' &\to a \\ B' &\to b \\ C' &\to c \end{aligned}$$


S
S
A
C
A
C
A
B
A′ A′ A′ B′ B′ B′ B, C′ B, C′
a a a b b b c c

Laufzeit des CYK-Algorithmus

Sei $n = |w|$ die Länge des Wortes, das untersucht wird. Die Größe der Grammatik wird als konstant angesehen. Dann gilt:

- $O(n^2)$ Tabellenfelder müssen ausgefüllt werden.
- Für das Ausfüllen jedes Tabellenfeldes müssen höchstens n Paare anderer Felder betrachtet werden.
  (Für $V_{1,n}$ müssen beispielsweise die Felder $V_{1,n-1}$, $V_{n,1}$ und $V_{1,n-2}$, $V_{n-1,2}$ und ... und $V_{1,1}$, $V_{2,n-1}$ betrachtet werden. Insgesamt $\leq n-1$ Paare von Feldern.)

Daher ergibt sich insgesamt als Zeitkomplexität: $O(n^3)$.

Damit ist der CYK-Algorithmus sehr viel effizienter als der Algorithmus von Folie 9.30. Aber für gewisse kontextfreie Grammatiken geht es schneller mit ...

Zusammenfassung 10. Vorlesung

in dieser Vorlesung neu

- Chomsky-Normalform
- CYK-Algorithmus

kommende Vorlesung

- Verallgemeinerung der NFAs, um genau die kontext-freien Sprachen zu akzeptieren