

# Automaten und Formale Sprachen

## 10. Vorlesung

Prof. Dr. Dietrich Kuske

FG Automaten und Logik, TU Ilmenau

Wintersemester 2023/24

# Chomsky-Normalform

## Definition

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

- alle Produktionen von  $G$  die Form

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

haben

- oder alle Produktionen von  $G$  die Form

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

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

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

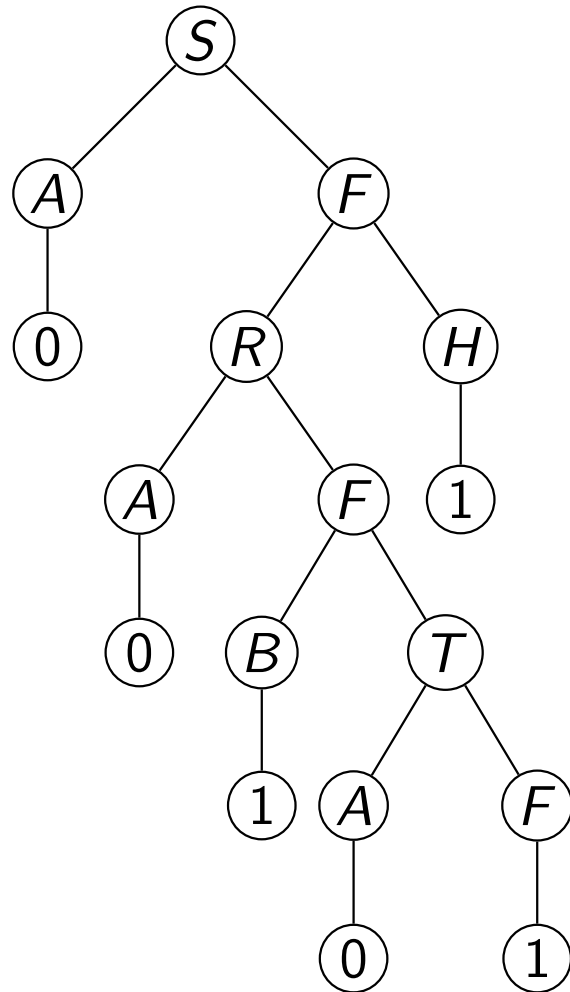
$$\begin{array}{ll} S \rightarrow \varepsilon \mid AF & R \rightarrow AF \\ F \rightarrow BR \mid RH \mid 1 & H \rightarrow BR \mid 1 \\ A \rightarrow 0 & B \rightarrow 1 \end{array}$$

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

$$S \Rightarrow \varepsilon \text{ und}$$

$$\begin{aligned} \underline{S} &\Rightarrow \underline{AF} \Rightarrow 0\underline{F} \Rightarrow 0\underline{RH} \Rightarrow 0\underline{AFH} \Rightarrow 00\underline{FH} \Rightarrow 00\underline{BRH} \\ &\Rightarrow 001\underline{RH} \Rightarrow 001\underline{AFH} \Rightarrow 0010\underline{FH} \Rightarrow 00101\underline{H} \Rightarrow 001011 \end{aligned}$$

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: $\varepsilon$ -Behandlung

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

- $G$  hat keine Produktionen der Form  $A \rightarrow \varepsilon$  mit  $A \neq S$
- und nur dann die Regel  $S \rightarrow \varepsilon$ , 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  $A_a \rightarrow a$  für  $a \in \Sigma$  ein.

~> Die Sprache der Grammatik bleibt unverändert und alle Produktionen haben die Form

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

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

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

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

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

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

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

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

Kommt eine Kettenregel  $A \rightarrow B$  mit  $A, B \in V$  vor, so ersetze sie durch die Menge aller Regeln  $A \rightarrow \alpha$  mit  $B \rightarrow \alpha$  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

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

Damit ist die Grammatik in Chomsky-Normalform. □



**Beispiel:** Sei

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

mit folgender Produktionsmenge  $P$ :

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

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).

$$\begin{array}{l} \text{Ergebnis: } S \rightarrow aAb \mid ab \mid A \\ A \rightarrow S \mid aaSc \end{array}$$

## Schritt 2: Vereinzeln der Terminale

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

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

Ergebnis:

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

**Schritt 4:** Elimination von Kettenregeln.

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

Ergebnis:

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

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  $A \rightarrow a$  oder  $A \rightarrow BC$  (die Regel  $S \rightarrow \varepsilon$  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.

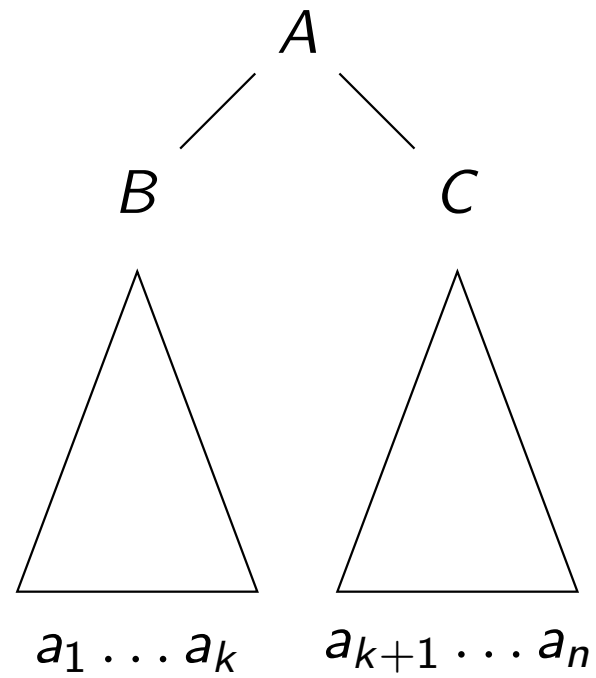
- **Möglichkeit 1:**  $w = a \in \Sigma$ , d.h.,  $w$  besteht aus einem einzigen Alphabetsymbol.

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

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

In diesem Fall gilt: Zunächst muß eine Produktion  $A \rightarrow BC$  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!

Probiere daher alle möglichen Werte für  $k$  durch. Das heißt:

Gegeben ein Wort  $w = a_1 \cdots a_n$  mit  $n \geq 2$ .

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  $(A \rightarrow BC) \in P$ ,  $B \in V_{1,k}$  und  $C \in V_{k+1,n-k}$ .  
In diesem Fall läßt sich  $w$  aus  $A$  ableiten.



um  $V_{1,k}$  zu bestimmen, gehen wir analog vor:

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

- Bestimme die Menge  $V_{1,j}$  der Nichtterminale, aus denen sich  $a_1 \cdots a_j$  ableiten läßt.
- Bestimme die Menge  $V_{j+1,k-j}$  der Nichtterminale, aus denen sich  $a_{j+1} \cdots a_k$  ableiten läßt.
- Dann ist  $V_{1,k}$  die Menge der Nichtterminale  $A$  mit  $A \rightarrow BC$ ,  $B \in V_{1,j}$  und  $C \in V_{j+1,k-j}$ .

Analog wird die Menge  $V_{k+1,n-k}$  bestimmt.

Die Menge  $V_{1,j}$  geht also in die Berechnung aller Mengen  $V_{1,k}$  mit  $j \leq k < n$  ein!

Um diese Mehrfachberechnung 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:

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

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

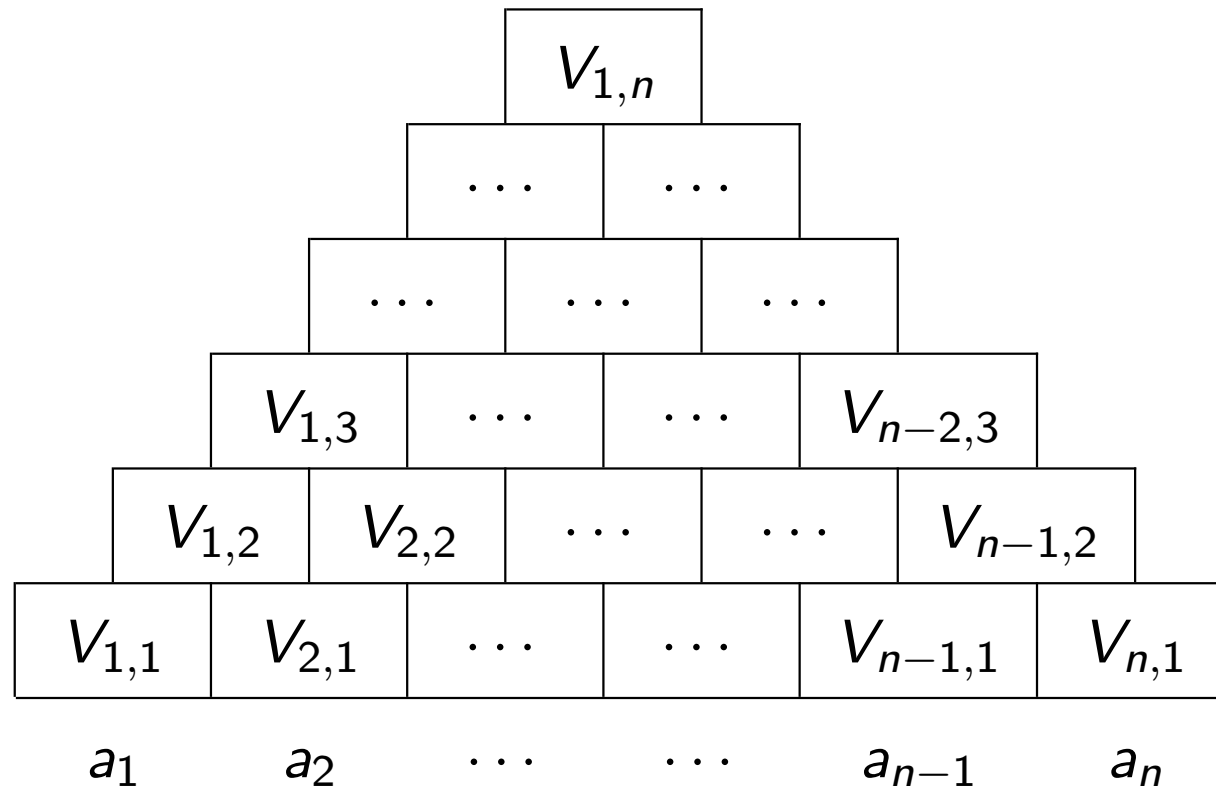
$$V_{i,j} = \{A \in V \mid A \Rightarrow_G^* w_{i,j}\}$$

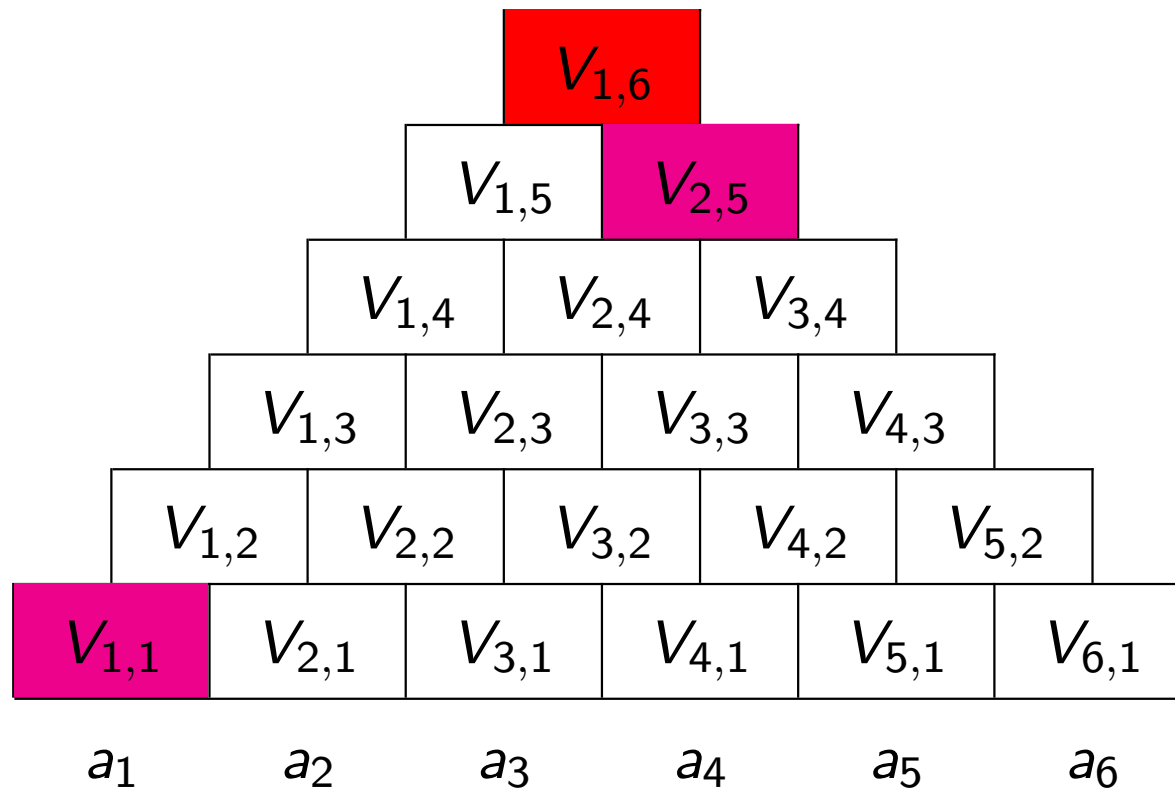
$V_{i,j}$  läßt sich aus den Mengen  $V_{\ell,k}$  mit  $k < j$  folgendermaßen bestimmen:

$$V_{i,j} = \{A \mid \exists (A \rightarrow BC) \in P \exists k < j : B \in V_{i,k} \text{ und } C \in V_{i+k,j-k}\}$$

## Praktische Ausführung des CYK-Algorithmus:

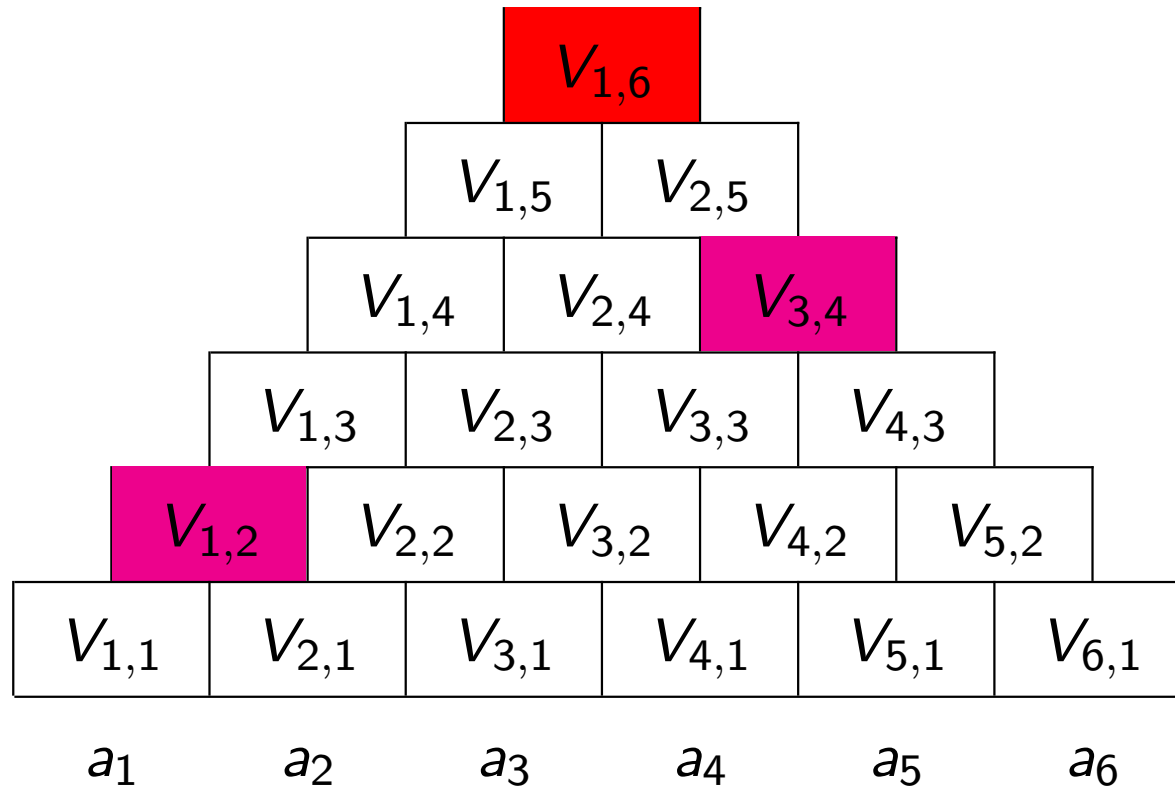
Wir tragen die Mengen  $V_{i,j}$  von Nichtterminalen zeilenweise in folgende Tabelle ein:





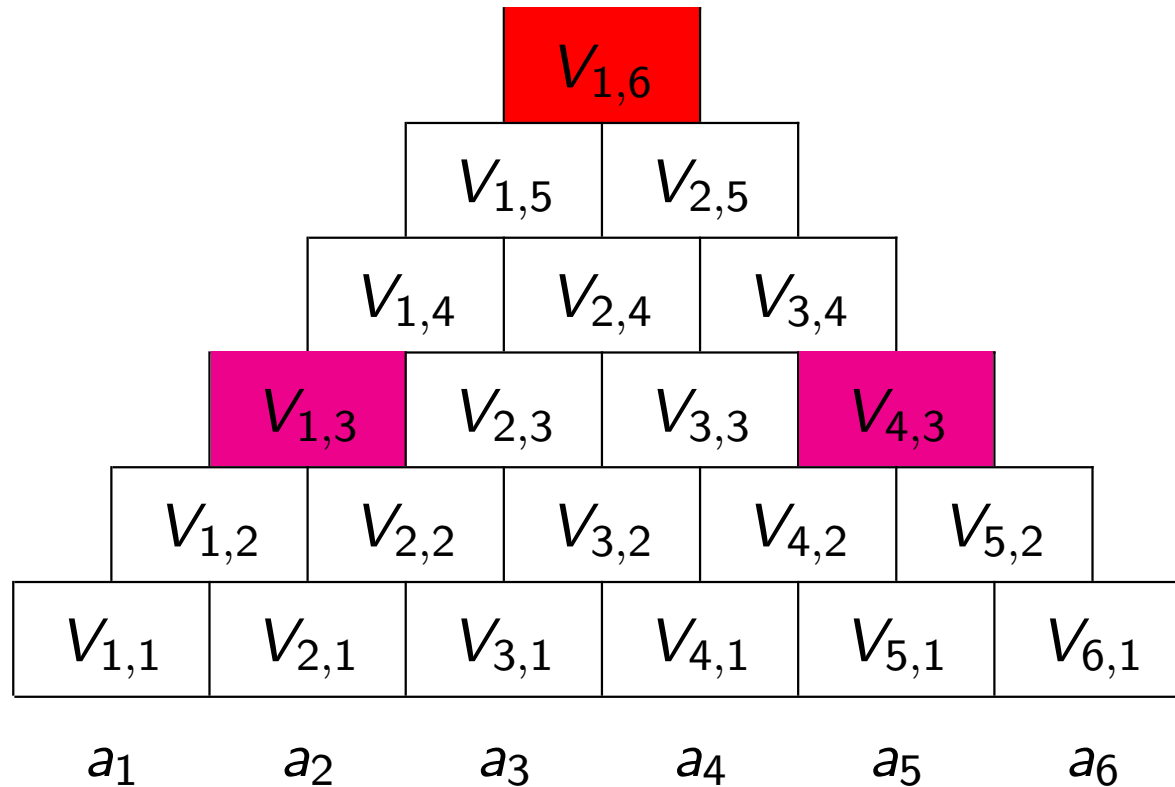
Wenn es  $(A \rightarrow BC) \in P$  gibt mit  $B \in V_{1,1}$  und  $C \in V_{2,5}$ ,  
so gilt  $A \Rightarrow BC \Rightarrow^* a_1 C \Rightarrow^* a_1 \dots a_6$ .

Füge also  $A$  zu  $V_{1,6}$  hinzu.

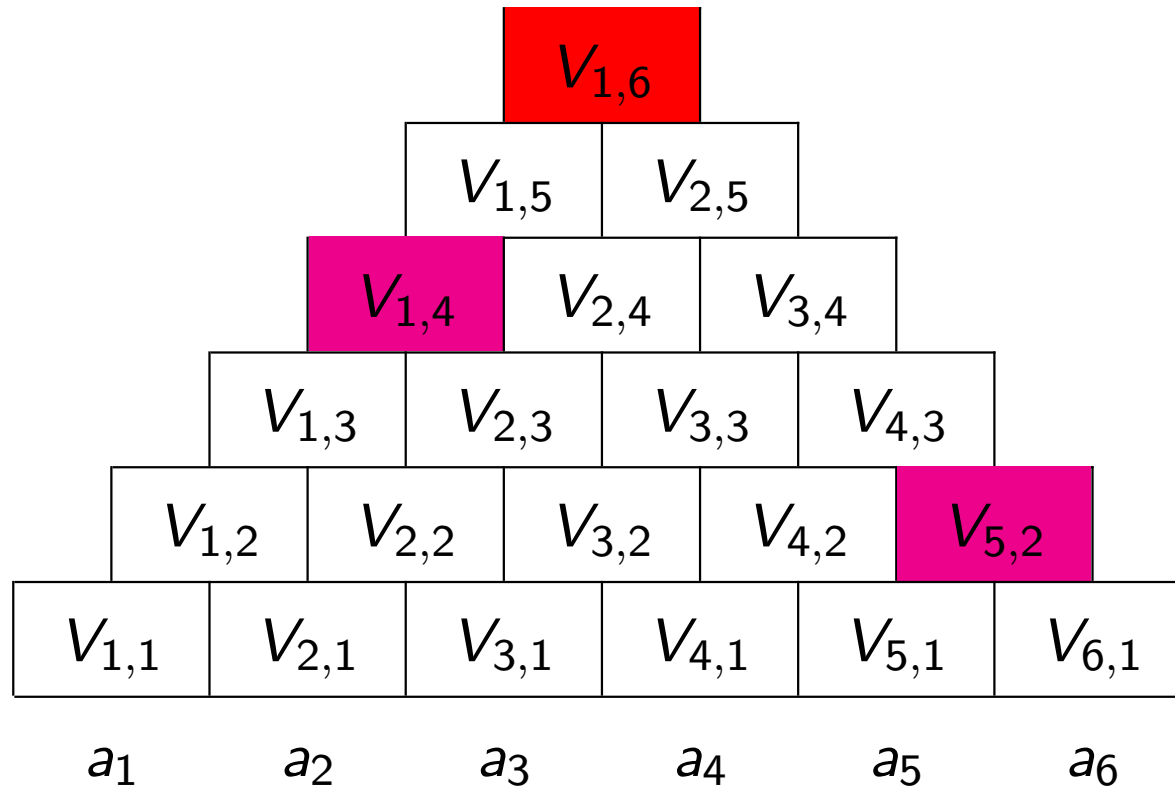


Wenn es  $(A \rightarrow BC) \in P$  gibt mit  $B \in V_{1,2}$  und  $C \in V_{3,4}$ ,  
 so gilt  $A \Rightarrow BC \Rightarrow^* a_1 a_2 C \Rightarrow^* a_1 \dots a_6$ .

Füge also  $A$  zu  $V_{1,6}$  hinzu.

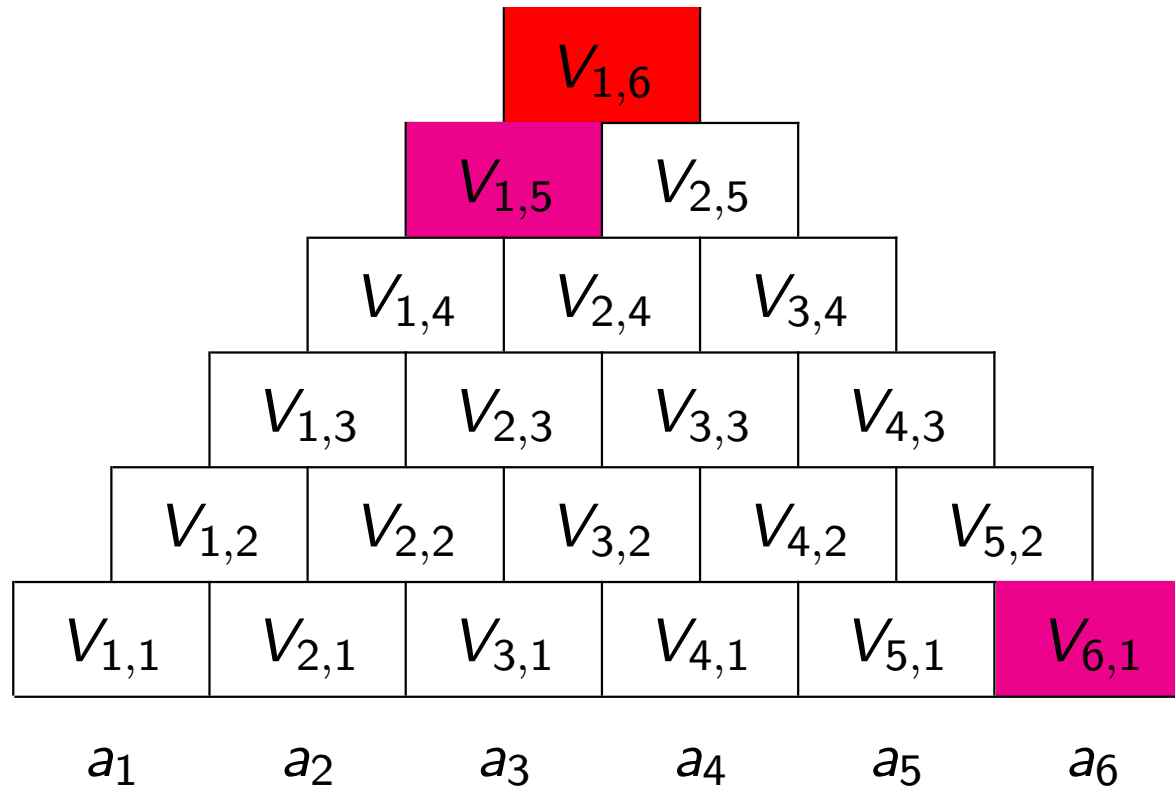


Wenn es  $(A \rightarrow BC) \in P$  gibt mit  $B \in V_{1,3}$  und  $C \in V_{4,3}$ ,  
 so gilt  $A \Rightarrow BC \Rightarrow^* a_1 \dots a_3 C \Rightarrow^* a_1 \dots a_6$ .  
 Füge also  $A$  zu  $V_{1,6}$  hinzu.



Wenn es  $(A \rightarrow BC) \in P$  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.





Wenn es  $(A \rightarrow BC) \in P$  gibt mit  $B \in V_{1,5}$  und  $C \in V_{6,1}$ ,  
 so gilt  $A \Rightarrow BC \Rightarrow^* a_1 \dots a_5 C \Rightarrow^* a_1 \dots a_6$ .  
 Füge also  $A$  zu  $V_{1,6}$  hinzu.

**Beispiel:** Betrachte die Grammatik  $G$  mit folgenden Produktionen:

$$S \rightarrow AD \mid FG$$

$$D \rightarrow SE \mid BC$$

$$E \rightarrow BC$$

$$F \rightarrow AF \mid a$$

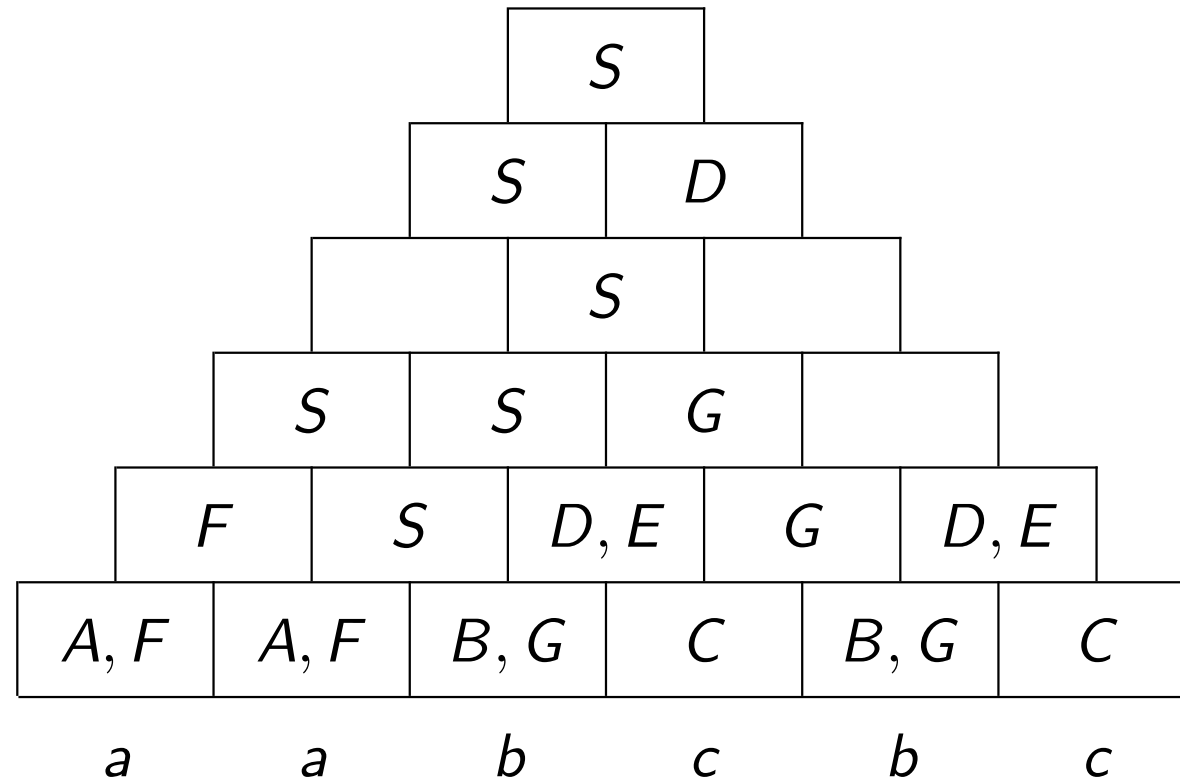
$$G \rightarrow BG \mid CG \mid b$$

$$A \rightarrow a$$

$$B \rightarrow b$$

$$C \rightarrow c$$

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



**Beispiel:** Betrachte die Grammatik  $G$  mit den folgenden Produktionen:

$$S \rightarrow AB$$

$$A \rightarrow ab \mid aAb$$

$$B \rightarrow c \mid cB$$

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

**Vorbereitung:** Umwandlung der Grammatik in Chomsky-Normalform:

$$S \rightarrow AB$$

$$A' \rightarrow a$$

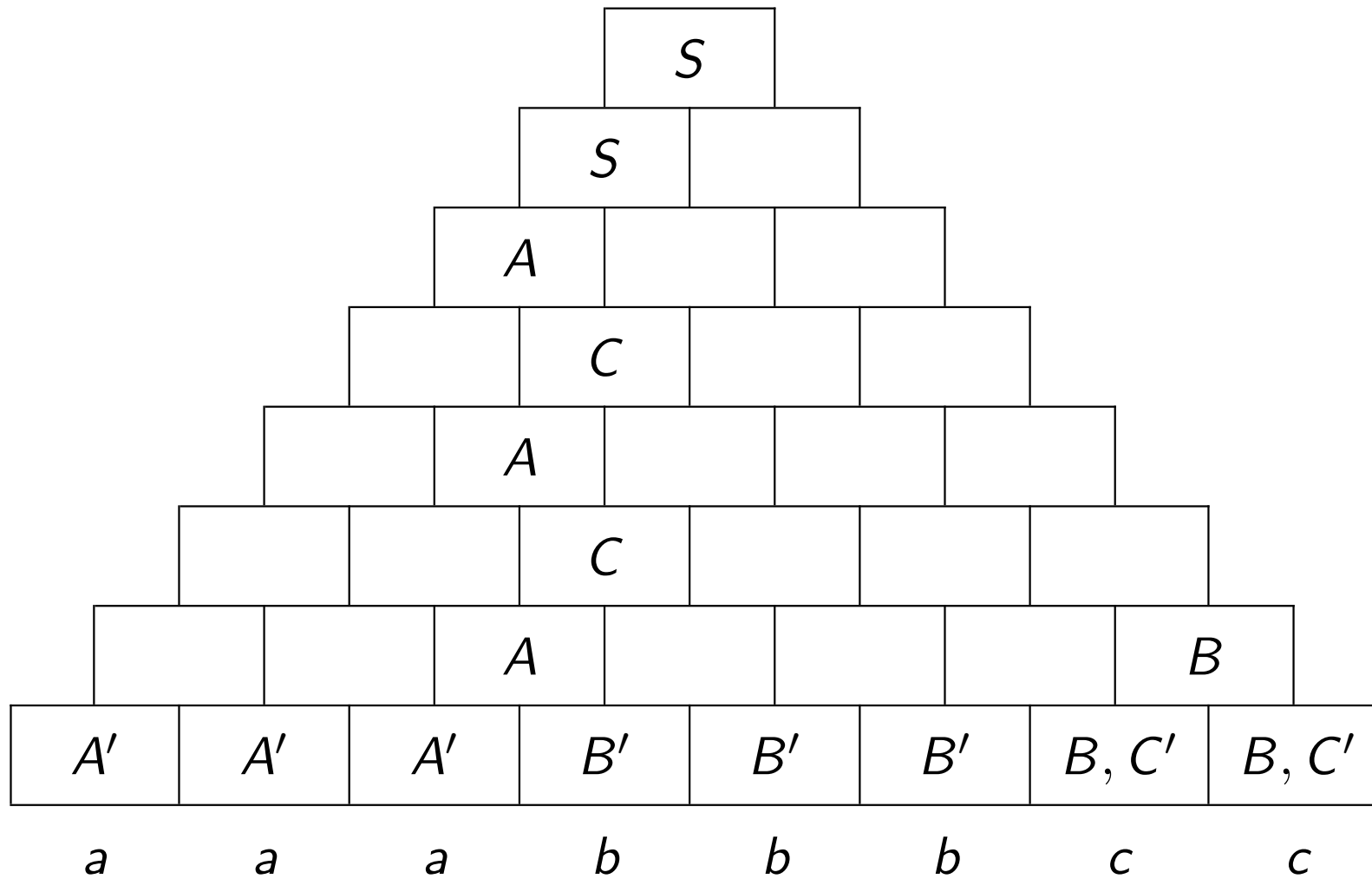
$$A \rightarrow A'B' \mid A'C$$

$$B' \rightarrow b$$

$$C \rightarrow AB'$$

$$C' \rightarrow c$$

$$B \rightarrow c \mid C'B$$



## 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