# Formale Sprachen und Automaten TINF18B4

Prof. Dr. Jörn Eisenbiegler

# Kapitel 4 – Typ 2

- Kellerautomaten
- Kontextfreie Grammatiken
- Zusammenhang von Kellerautomaten und kontextfreien Grammatiken
- Der Algorithmus von Cocke, Younger und Kasami
- Das Pumping-Lemma für kontextfreie Sprachen

# Der Algorithmus von Cocke, Younger und Kasami

- Die Chomsky-Normalform (CNF)
- Der Algorithmus von Cocke, Younger und Kasami

# Chomsky-Hierarchie

| Typ | Name | Erlaubte Produktionen | Akzeptierende Maschine | Beispiel |
|---|---|---|---|---|
| 3 | Regulär | N $\rightarrow wM$<br>$w \in T^*$ | Endlicher Automat | $a^n$ |
| 2 | Kontextfrei | N $\rightarrow w$<br>$w \in (N \cup T)^*$ | Kellerautomat | $a^n b^n$ |
| | | | | |
| | | | | |

Skript Worsch: Seite 80-83

# Ziel

Gegeben: eine Grammatik und ein Wort

Gesucht: der Ableitungsbaum

# Ansatz – Chomsky Normalform

Die Zahl der Regeln ist endlich

Die Länge des Wortes ist endlich

Jede Ableitung fügt entweder ein NonTerminal hinzu oder wandelt ein NonTerminal in ein Terminal

Für ein Wort der Länge n brauchen wir n-1 Schritte der Form
N $\rightarrow M_1 M_2$
und n Schritte der Form
N $\rightarrow$ t

# Beispiel

- $P = \{S \to AB|BC, A \to BA|a, B \to CC|b, C \to AB|a\}$
  $$w = ba$$

# Beispiel

Bottom-Top-Analyse

- $P = \{S \to AB|BC, A \to BA|a, B \to CC|b, C \to AB|a\}$
$$w = ba$$
- w entsteht durch einen Ableitungsschritt der Form
N $\to M_1 M_2$
und 2 Ableitungsschritte der Form
N $\to$ t
- b kann nur durch B $\to$ b entstehen
a kann durch A $\to$ a oder C $\to$ a entstehen
- ba kann aus BA oder BC entstehen
- BC kann durch S $\to$ BC entstehen

# Systematisch – Definition Teilwort

- Eingabe $w = t_0 \dots t_{n-1} \in T^*$
- Es sei $w[i,j)$ das Teilwort vom i-ten Terminalsymbol (eingeschlossen) bis zum j-ten Terminalsymbol (ausgeschlossen)
- Die Länge von $w[i,j)$ ist stets j-i
- Beispiele:
  - $w[2{,}4) = t_2 t_3$
  - $w[0,n) = w$
  - $w[i,i) = \varepsilon$

w = w[0,n), wobei n = |w|

# Systematisch – Mögliche Quellsymbole für Teilwort

- Die Menge aller NonTerminal Symbole aus denen ein Teilwort abgeleitet werden kann

- $D[i;j] = \{N | N \Rightarrow^* w([i,j)\ \}$

# Systematisch – Berechnung der möglichen Quellsymbole

- Wenn w[i, j) aus N ableitbar ist dann beginnt der Ableitungsbaum mit einer Produktion der Form N → $M_1 M_2$

- W[i, j) ist dann zusammengesetzt aus den Teilworten
  $w_1$ und $w_2$
  die von $M_1$ und $M_2$ erzeugt werden

- W[i, j) = w[i, k) w[k, j) = $w_1\ w_2$
  $M_1 \Rightarrow^* w[i, k)$und $M_2 \Rightarrow^* w[k, j)$

# Systematisch – Berechnung der möglichen Quellsymbole

N ist in $D[i;j]$ enthalten wenn es eine Produktion

- N $\rightarrow M_1 M_2$ gibt und
- $M_1 \in D[i;k]$ und $M_2 \in D[k;j]$

# Beispiel

- $P = \{S \to AB|BC, A \to BA|a, B \to CC|b, C \to AB|a\}$
$$w = ba$$
- D[0;1] = {B}
D[1;2] = {A, C}
- Es gibt eine Produktion $S \to BC$
- S ∈ D[0;2]
- Wegen $A \to BA$ ist auch A ∈ D[0;2]

# Algorithmus von Cocke, Younger und Kasami (kurz CYK)

- Der Algorithmus von Cocke, Younger und Kasami (kurz CYK) kann für jede Grammatik $G = (N, T, S, P)$ in Chomsky-Normalform und jedes Wort $w \in T^*$ auf deterministische Weise feststellen, ob $w \in L(G)$ ist oder nicht.

# Algorithmus von Cocke, Younger und Kasami (kurz CYK)

- CYK berechnet eine Pyramiden-Datenstruktur.

Oben drüber immer von
linkem unten drunter
bis rechtem unten drunter

$$D[0;5]$$
$$D[0;4] \quad D[1;5]$$
$$D[0;3] \quad D[1;4] \quad D[2;5]$$
$$D[0;2] \quad D[1;3] \quad D[2;4] \quad D[3;5]$$
$$D[0;1] \quad D[1;2] \quad D[2;3] \quad D[3;4] \quad D[4;5]$$
$$t_0 \quad t_1 \quad t_2 \quad t_3 \quad t_4$$
$$0 \quad 1 \quad 2 \quad 3 \quad 4 \quad 5$$

- $D[i;j] = \{N | N \Rightarrow^* w(i,j)\}$

- $t_0 \ldots t_{n-1}$ genau dann aus $L(G)$, wenn $S \in D[0;\ n]$ ist.

# Algorithmus von Cocke, Younger und Kasami (kurz CYK)

- Berechne alle D[i; i+1] aus den Regeln der Form N → t
- Für alle k von 2 bis n berechne alle D[i; i+k]
- Berechne D[i; i+k]
  Für alle Regeln der Form N → $M_1 M_2$
  Für alle r von 1 bis k-1
  Prüfe ob $M_1 \in D[i; r]$ und $M_2 \in D[r; i+k]$

# Algorithmus von Cocke, Younger und Kasami (kurz CYK)

$\boldsymbol{for}\ i \leftarrow 0\ \boldsymbol{to}\ n - 1\ \boldsymbol{do}$

$\quad D[i; i+1] \leftarrow \{N | N \rightarrow t_i \in P\}$

$\boldsymbol{od}$

$\boldsymbol{for}\ k \leftarrow 2\ \boldsymbol{to}\ n\ \boldsymbol{do}$

$\quad \boldsymbol{for}\ i \leftarrow 0\ \boldsymbol{to}\ n - k\ \boldsymbol{do}$

$\quad\quad D[i\,;\,i + k] \leftarrow \emptyset;$

$\quad\quad \boldsymbol{for}\ m \leftarrow i + 1\ \boldsymbol{to}\ i + k - 1\ \boldsymbol{do}$

$\quad\quad\quad \boldsymbol{if}\ N \rightarrow M_1 M_2 \in P \wedge M_1 \in D[i\,;m] \wedge M_2 \in D[m\,;i + k]\ \boldsymbol{then}$

$\quad\quad\quad\quad D[i\,;\,i + k] \leftarrow D[i\,;\,i + k] \cup \{N\}$

$\quad\quad\quad \boldsymbol{fi}$

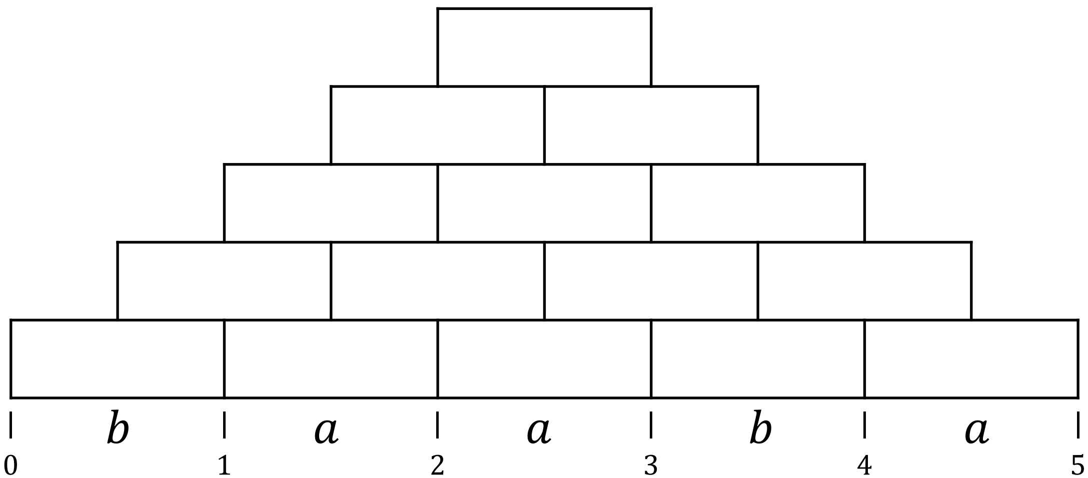
$\quad\quad \boldsymbol{od}$

$\quad \boldsymbol{od}$

$\boldsymbol{od}$

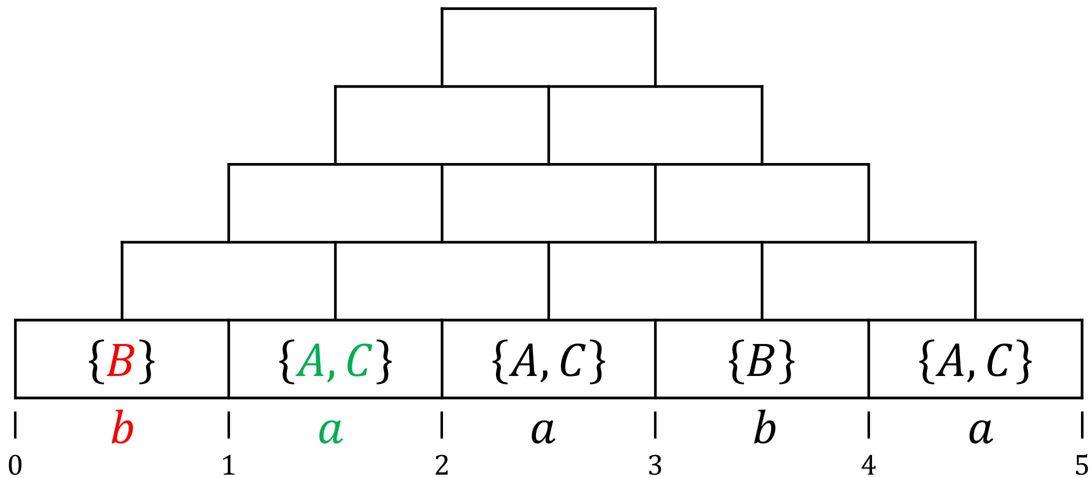
# Beispiel

- $P = \{S \rightarrow AB|BC, A \rightarrow BA|a, B \rightarrow CC|b, C \rightarrow AB|a\}$
  $w = baaba$



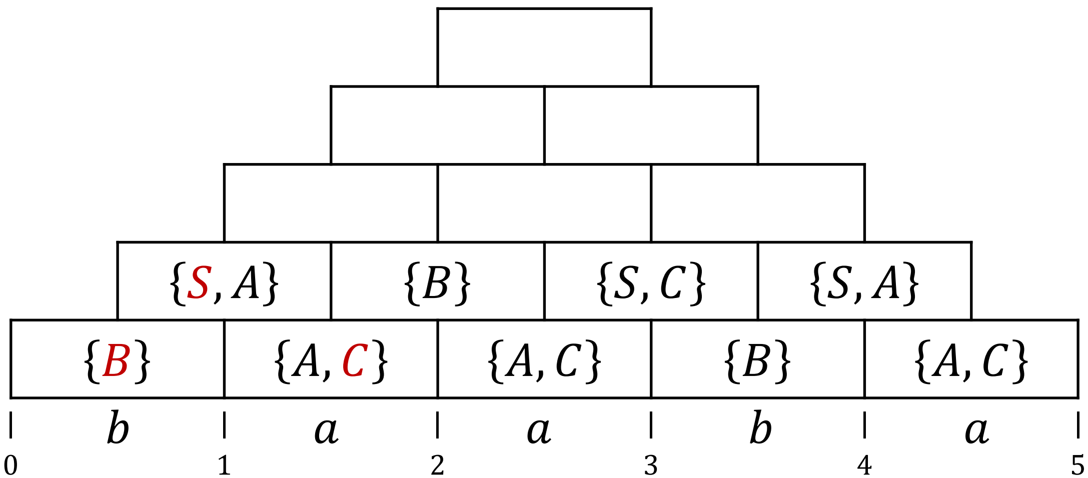
# Beispiel

- $P = \{S \to AB|BC, A \to BA|a, B \to CC|b, C \to AB|a\}$
  $w = baaba$



# Beispiel

- $P = \{S \rightarrow AB|BC, A \rightarrow BA|a, B \rightarrow CC|b, C \rightarrow AB|a\}$
  $w = baaba$



# Beispiel

- $P = \{S \rightarrow AB|BC, A \rightarrow BA|a, B \rightarrow CC|b, C \rightarrow AB|a\}$
  $w = baaba$

| | | | | |
|---|---|---|---|---|
| | | | | |
| | | | | |
| $\{\,\}$ | $\{B\}$ | $\{B\}$ | | |
| $\{S, A\}$ | $\{B\}$ | $\{S, C\}$ | $\{S, A\}$ | |
| $\{B\}$ | $\{A, C\}$ | $\{A, C\}$ | $\{B\}$ | $\{A, C\}$ |
| $b$ | $a$ | $a$ | $b$ | $a$ |

0 1 2 3 4 5

# Beispiel

- $P = \{S \to AB|BC, A \to BA|a, B \to CC|b, C \to AB|a\}$
  $w = baaba$

| | | | | |
|---|---|---|---|---|
| | | | | |
| $\{\,\}$ | $\{S,A,C\}$ | | | |
| $\{\,\}$ | $\{B\}$ | $\{B\}$ | | |
| $\{S,A\}$ | $\{B\}$ | $\{S,C\}$ | $\{S,A\}$ | |
| $\{B\}$ | $\{A,C\}$ | $\{A,C\}$ | $\{B\}$ | $\{A,C\}$ |
| $b$ | $a$ | $a$ | $b$ | $a$ |

0 1 2 3 4 5

$$S \in D[0,5] \Rightarrow w \in L(G)$$

# Beispiel

- $P = \{S \rightarrow AB|BC, A \rightarrow BA|a, B \rightarrow CC|b, C \rightarrow AB|a\}$
  $w = baaba$

| | | | | |
|---|---|---|---|---|
| $\{S,A,C\}$ | | | | |
| $\{\}$ | $\{S,A,C\}$ | | | |
| $\{\}$ | $\{B\}$ | $\{B\}$ | | |
| $\{S,A\}$ | $\{B\}$ | $\{S,C\}$ | $\{S,A\}$ | |
| $\{B\}$ | $\{A,C\}$ | $\{A,C\}$ | $\{B\}$ | $\{A,C\}$ |
| $b$ | $a$ | $a$ | $b$ | $a$ |
| 0–1 | 1–2 | 2–3 | 3–4 | 4–5 |

$$S \in D[0,5] \Rightarrow w \in L(G)$$

# Überblick

- Kontextfreie Sprachen lassen sich mit Hilfe von kontextfreien Grammatiken (oder Kellerautomaten) spezifizieren.
  - Dabei sind alle Produktionen von der Form $X \rightarrow w \quad mit\ X \in N\ und\ w \in (N \cup T)^*$.
- Kontextfreie Sprachen lassen sich mit Hilfe von Kellerautomaten erkennen.
  - Mit Kellerautomaten lassen sich dazu die Top-down- oder die Bottom-up-Syntaxanalyse durchführen.
  - Beide Verfahren sind nichtdeterministisch.
- Kontextfreie Sprachen lassen sich in CNF-Form wandeln.
  - In dieser Form lassen sich die Sprachen mit dem Algorithmus von Cocke, Younger und Kasami einer Laufzeit von $O(n^3)$ deterministisch erkennen.

# Lernziele

- Die Idee Problemstellungen so umzuformen das sie gutartig sind (Stichwort „Normalform“)
- Den Ansatz Probleme in immer kleiner Teilprobleme zu zerlegen (bis hinunter zur Trivialität)
- Ein Problem durch Hilfskonstrukte (hier: Teilwortableitungen) in Teilprobleme zu zerlegen

# Mögliche Klausuraufgaben

- Den CYK-Algorithmus erklären

- Mithilfe des CYK-Algorithmus prüfen ob ein Wort aus einer Grammatik in Chomsky-Normalform erzeugt werden kann