

# Kontextfreie Sprachen

- besitzen große Bedeutung bei der Definition von Programmiersprachen sowie der Syntaxüberprüfung von Programmen
- Chomsky-Normalform
- effiziente Lösung des Wortproblems (CYK-Algorithmus)
- Grenzen kontextfreier Sprachen (Pumping Lemma)

## Beispiel für eine kontextfreie Grammatik

$$G = (\{S, X, C\}, \{x, c, 0, +, -, ;, :=, \neq, \text{LOOP}, \text{WHILE}, \text{DO}, \text{END}\}, P, S)$$

mit folgenden Regeln in der Regelmenge  $P$ :

$$S \rightarrow S; S$$

$$S \rightarrow \text{LOOP } X \text{ DO } S \text{ END}$$

$$S \rightarrow \text{WHILE } X \neq 0 \text{ DO } S \text{ END}$$

$$S \rightarrow X := X + C$$

$$S \rightarrow X := X - C$$

$$X \rightarrow x$$

$$C \rightarrow c$$

# Chomsky Normalform

## Definition

Eine kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  ist in **Chomsky Normalform**, falls jede Regel in  $P$  in einer der Formen (i)-(iii) ist:

- (i)  $A \rightarrow BC$  mit  $A, B, C \in V$ ,
- (ii)  $A \rightarrow a$  mit  $A \in V$  und  $a \in \Sigma$ ,
- (iii)  $S \rightarrow \varepsilon$ , wobei  $S$  auf keiner rechten Seite einer Regel vorkommt.

## Satz

Zu jeder kontextfreien Grammatik  $G$  kann man eine kontextfreie Grammatik  $G'$  in Chomsky Normalform konstruieren mit  $L(G) = L(G')$ .

# Der Algorithmus von Cocke, Younger und Kasami

**Eingabe:** kontextfreie Grammatik  $G = (V, \Sigma, P, S)$  in Chomsky NF,  
Wort  $w \in \Sigma^*$

**Ausgabe:** *ja*, falls  $w \in L(G)$ ; *nein*, sonst.

## Grundidee des CYK-Algorithmus

- $w[s, t]$  sei das Teilwort von  $w$  von der Stelle  $s$  bis zur Stelle  $t$ , wobei  $1 \leq s \leq t \leq n = |w|$ .
- Bestimme für  $1 \leq j \leq n$  und  $1 \leq i \leq n - j + 1$  die Mengen

$$T[i, j] = \{A \in V \mid A \xRightarrow{*} w[i, i + j - 1]\}.$$

- Es gilt  $w \in L(G)$  genau dann, wenn  $S \in T[1, n]$ .

# CYK-Algorithmus – Fortsetzung

Induktive Bestimmung der Mengen  $T[i, i + j - 1]$ :

- Für  $j = 1$ :  $T[i, 1] = \{A \in V \mid A \rightarrow w[i, i] \in P\}$
- Für  $j > 1$ :

$$T[i, j] = \{A \in V \mid \text{es gibt } B, C \in V \text{ und } 1 \leq k < j \text{ mit} \\ A \rightarrow BC \in P, B \in T[i, i + k - 1], C \in T[i + k, i + j - 1]\}$$

- CYK-Algorithmus ist Beispiel für **Dynamische Programmierung**.

## Beispiel für CYK-Algorithmus

Die Sprache  $L = \{a^n b^n c^m \mid m, n \geq 1\}$  ist kontextfrei und wird von der Grammatik  $G = (V, \Sigma, P, S)$  mit den Regeln

$$S \rightarrow AB, \quad A \rightarrow aAb \mid ab, \quad B \rightarrow cB \mid c$$

erzeugt. Eine äquivalente Grammatik in Chomsky Normalform für  $L$  hat folgende Menge von Regeln:

$$\begin{array}{lll} S \rightarrow AB, & B \rightarrow EB \mid c, & C \rightarrow a, \\ A \rightarrow CD \mid CF, & F \rightarrow AD, & D \rightarrow b, \quad E \rightarrow c, \end{array}$$

Sei  $x = aaabbbcc$ . Dann erzeugt der Algorithmus folgende Tabelle.

# Beispiel für CYK-Algorithmus – Tabelle

$\xrightarrow{\quad\quad\quad} i$   
 $a \quad a \quad a \quad b \quad b \quad b \quad c \quad c \quad = x$

	$C$	$C$	$C$	$D$	$D$	$D$	$E, B$	$E, B$
			$A$				$B$	
			$F$					
		$A$						
		$F$						
	$A$							
	$S$							
	$S$							

$\downarrow j$

## Beispiel für CYK-Algorithmus – Erklärung

In der Tabelle stellt jede Zelle eine Menge  $T[i, j]$  von Nichtterminalen dar. Die Koordinatenachsen für  $i$  und  $j$  sind eingezeichnet.

Der Algorithmus besteht aus drei Teilen. Im ersten Teil wird die oberste Zeile ( $T[i, 1]$ ) berechnet.

Im zweiten Teil werden zeilenweise die weiteren Mengen  $T[i, j]$  berechnet, indem immer die zugehörige Spalte von oben und die nach rechts oben führende Diagonale betrachtet werden.

Im dritten Teil des Algorithmus schließlich wird sich die Menge  $T[1, n]$  angeschaut, hier  $T[1, 8]$ . Gilt  $S \in T[1, n]$ , dann und nur dann gibt der Algorithmus “ja” aus, das heißt, es gibt eine Ableitung  $S \xRightarrow{*} x$ , hier  $S \xRightarrow{*} aaabbbcc$ , somit erzeugt die gegebene Grammatik  $x = aaabbbcc$ .

## Fortsetzung CYK-Algorithmus

Noch eine Bemerkung: aus der aufgestellten Tabelle kann man auch die Ableitung für das Wort  $x$  ablesen, indem wir rückwärts die Regeln anwenden, die auf  $S$  in  $T[1, n]$  geführt haben. Hier sieht die Ableitung für  $aaabbbc$  dann folgendermaßen aus (die Nonterminale, die ersetzt werden, sind jeweils unterstrichen).

$$\begin{aligned} \underline{S} &\Longrightarrow \underline{A}B \Longrightarrow C\underline{F}B \Longrightarrow C\underline{A}DB \Longrightarrow CC\underline{F}DB \Longrightarrow CC\underline{A}DDB \\ &\Longrightarrow \underline{C}CCDDB \Longrightarrow a\underline{C}CCDDB \Longrightarrow aa\underline{C}DDB \Longrightarrow aaa\underline{D}DDB \\ &\Longrightarrow aaab\underline{D}DB \Longrightarrow aaabb\underline{D}B \Longrightarrow aaabbb\underline{B} \Longrightarrow aaabbbc \end{aligned}$$

# Pumping Lemma für kontextfreie Sprachen

Sei  $L$  eine kontextfreie Sprache. Dann gibt es eine Konstante  $k \in \mathbb{N}$ , so dass für alle Wörter  $z \in L$  mit  $|z| \geq k$  eine Zerlegung  $z = uvwxy$  existiert, so dass gilt:

- (i)  $|vwx| \leq k$  und  $|vx| \geq 1$ ,
- (ii) für alle  $i \in \mathbb{N}$  gilt  $uv^iwx^iy \in L$ .

# Pumping Lemma – Anwendung

Mit Hilfe des Pumping Lemmas kann man zeigen, dass folgende Sprachen nicht kontextfrei sind:

- $\{a^n b^n c^n \mid n \geq 1\}$
- $\{ww \mid w \in \{a, b\}^*\}$
- $\{a^m b^n \mid 0 \leq m, 1 \leq n \leq 2^m\}$

# Abschlusseigenschaften

## Satz

Die Menge der kontextfreien Sprachen ist unter den Operationen

- (i) Vereinigung,
- (ii) Produkt (Konkatenation) und
- (iii) Kleene-Stern

abgeschlossen. Sie ist unter den Operationen

- (iv) Durchschnitt und
- (v) Komplement

nicht abgeschlossen.

# Weitere Entscheidungsprobleme

## Satz

Das Leerheitsproblem und das Endlichkeitsproblem für kontextfreie Grammatiken sind entscheidbar.

## Satz

Das Schnittproblem und das Äquivalenzproblem für kontextfreie Grammatiken sind unentscheidbar.

# Kellerautomaten

- **endliche Automaten** besitzen nur **begrenzten** Speicherplatz (ihre Zustände).
- **Turingmaschinen** besitzen **unbegrenzten** Speicherplatz mit im Prinzip **wahlfreiem Zugriff**.
- **Kellerautomaten** besitzen **unbegrenzten** Speicherplatz in Form eines **Kellerspeichers (Stack)**.

# Definition Kellerautomat

**Definition** Ein (nichtdeterministischer) Kellerautomat (kurz PDA von *pushdown automaton*)  $M$  ist ein 6-Tupel  $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$ . Dabei ist

- $Z$  das Zustandsalphabet,
- $\Sigma$  das Eingabealphabet,
- $\Gamma$  das Kellularphabet,
- $z_0 \in Z$  der Anfangszustand,
- $\delta: Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow 2_{\neq}^{Z \times (\Gamma \setminus \{\#\})^*}$  die Zustandsüberföhrungsfunktion ( $2_{\neq}^X$  bedeutet dabei die Menge der endlichen Teilmengen von  $X$ ),
- $\# \in \Gamma$  das Kelleraufangssymbol.

# Arbeitsweise von Kellerautomaten

- Der PDA startet im Zustand  $z_0$  und mit dem Kellerinhalt  $\#$ .
- Bei  $(z', B_1B_2 \dots B_k) \in \delta(z, a, A)$  mit  $a \in \Sigma$  befindet sich der PDA im Zustand  $z$ , liest auf dem Eingabeband ein  $a$  und im Keller ein  $A$  (das oberste Symbol). Er entfernt dieses  $A$  aus dem Keller, geht in den Zustand  $z'$  über, bewegt den Lesekopf auf dem Eingabeband einen Schritt nach rechts und schreibt auf den Keller das Wort  $B_1B_2 \dots B_k$  derart, dass  $B_1$  das neue oberste Symbol des Kellers ist.
- Bei  $(z', B_1B_2 \dots B_k) \in \delta(z, \varepsilon, A)$  liest er im Gegensatz zur obigen Aktion auf dem Eingabeband nichts ein und bewegt den Lesekopf auf dem Eingabeband nicht.
- Der PDA akzeptiert die Eingabe, wenn er sie vollständig eingelesen hat und der Keller leer ist (auch  $\#$  steht nicht mehr im Keller).

# Konfiguration von Kellerautomaten

**Definition** Sei  $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$  ein Kellerautomat. Eine **Konfiguration** von  $M$  ist ein Tripel  $k \in Z \times \Sigma^* \times \Gamma^*$ .

Anschaulich beschreibt eine Konfiguration  $k = (z, v, \gamma)$  folgende augenblickliche Situation des PDA:

- er befindet sich im Zustand  $z$ ,
- $v$  ist die noch nicht verarbeitete Eingabe auf dem Eingabeband und
- im Keller steht das Wort  $\gamma$ , wobei das erste Symbol von  $\gamma$  das oberste Kellersymbol sein soll.

# PDA – Konfigurationsübergänge und akzeptierte Sprache

$k_1 \vdash k_2$  bedeutet, dass der Kellerautomat die Konfiguration  $k_1$  in genau einem Schritt in die Konfiguration  $k_2$  überführt.

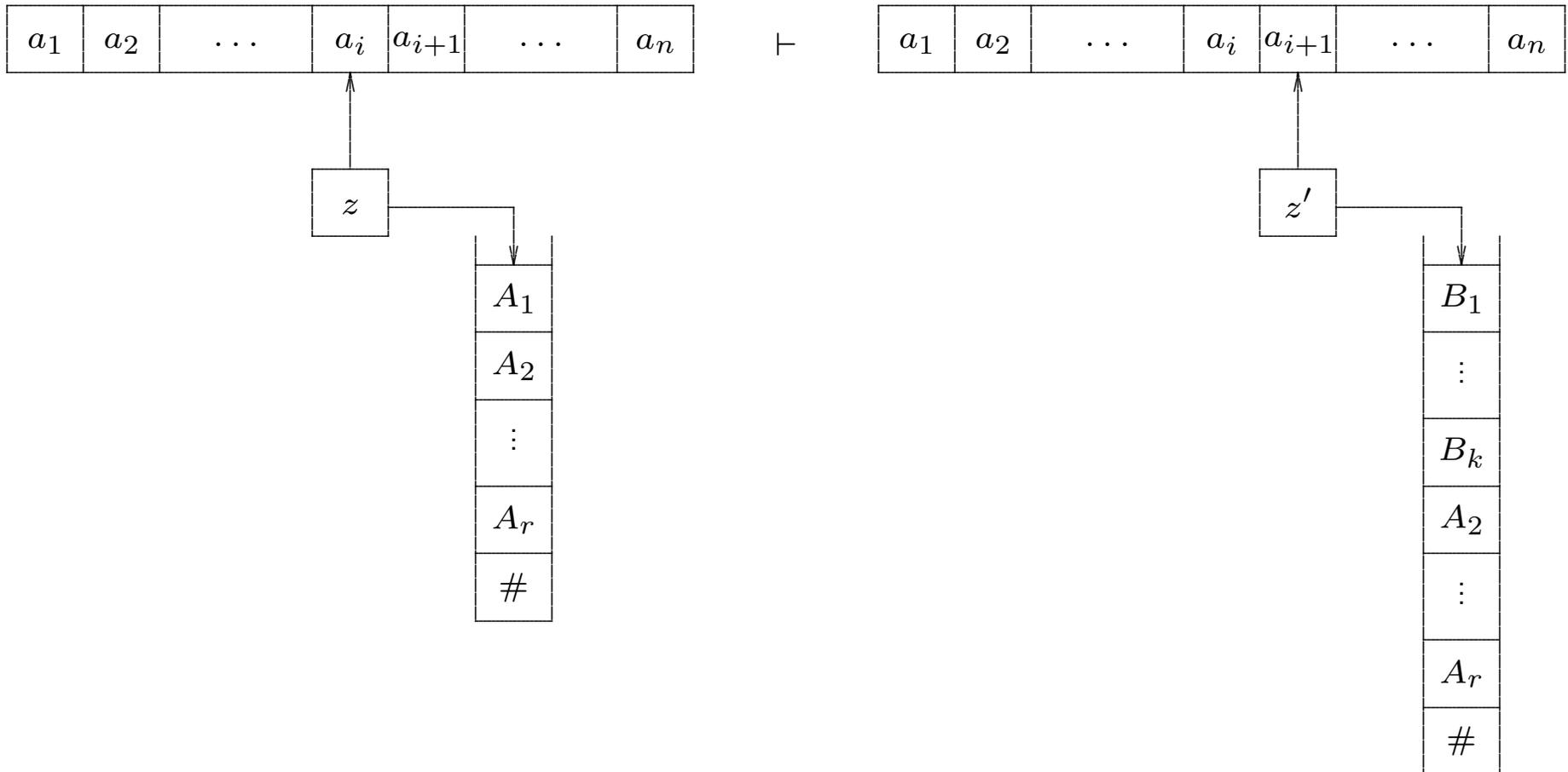
Unter  $k_1 \vdash^* k_2$  wollen wir wieder verstehen, dass der Kellerautomat  $k_1$  in endlich vielen Schritten (auch null) in  $k_2$  überführt.

## Definition

Für einen PDA  $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$  sei die von ihm **akzeptierte Sprache**  $T(M)$  definiert durch

$$T(M) = \{w \in \Sigma^* \mid (z_0, w, \#) \vdash^* (z, \varepsilon, \varepsilon) \text{ für ein } z \in Z\}.$$

# Abbildung Konfigurationsübergang



## Beispiel eines deterministischen PDA

Sei  $M = (\{z_0, z_1\}, \{a, b, c\}, \{A, B, \#\}, \delta, z_0, \#)$  ein Kellerautomat mit

$$\delta(z_0, a, X) = \{(z_0, AX)\} \quad \text{für } X \in \{A, B, \#\},$$

$$\delta(z_0, b, X) = \{(z_0, BX)\} \quad \text{für } X \in \{A, B, \#\},$$

$$\delta(z_0, c, X) = \{(z_1, X)\} \quad \text{für } X \in \{A, B, \#\},$$

$$\delta(z_1, a, A) = \{(z_1, \varepsilon)\},$$

$$\delta(z_1, b, B) = \{(z_1, \varepsilon)\},$$

$$\delta(z_1, \varepsilon, \#) = \{(z_1, \varepsilon)\}.$$

Dann gilt  $T(M) = \{w c w^R \mid w \in \{a, b\}^*\}$ , z.B.  $bacab \in T(M)$ , da

$$\begin{aligned} (z_0, bacab, \#) \vdash (z_0, acab, B\#) \vdash (z_0, cab, AB\#) \vdash (z_1, ab, AB\#) \\ \vdash (z_1, b, B\#) \vdash (z_1, \varepsilon, \#) \vdash (z_1, \varepsilon, \varepsilon). \end{aligned}$$

## Beispiel eines nichtdeterministischen PDA

Es sei  $M = (\{z_0, z_1\}, \{a, b\}, \{A, B, \#\}, \delta, z_0, \#)$  ein Kellerautomat, mit

$$\delta(z_0, a, X) = \{(z_0, AX)\} \quad \text{für } X \in \{B, \#\},$$

$$\delta(z_0, a, A) = \{(z_0, AA), (z_1, \varepsilon)\},$$

$$\delta(z_0, b, X) = \{(z_0, BX)\} \quad \text{für } X \in \{A, \#\},$$

$$\delta(z_0, b, B) = \{(z_0, BB), (z_1, \varepsilon)\},$$

$$\delta(z_0, \varepsilon, \#) = \{(z_1, \varepsilon)\},$$

$$\delta(z_1, a, A) = \{(z_1, \varepsilon)\},$$

$$\delta(z_1, b, B) = \{(z_1, \varepsilon)\},$$

$$\delta(z_1, \varepsilon, \#) = \{(z_1, \varepsilon)\}.$$

Dann gilt offensichtlich  $T(M) = \{ww^R \mid w \in \{a, b\}^*\}$ .

# Kellerautomaten und kontextfreie Sprachen

## Satz

Eine Sprache  $L$  ist kontextfrei genau dann, wenn ein (nichtdeterministischer) Kellerautomat  $M$  mit  $T(M) = L$  existiert.

Die von deterministischen Kellerautomaten akzeptierten Sprachen heißen **deterministisch kontextfrei**.

## Satz

Die Menge der deterministisch kontextfreien Sprachen ist eine **echte Teilmenge** der Menge der kontextfreien Sprachen.