

Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2020/21

Die Chomsky-Hierarchie

Man unterscheidet vier Typen von Grammatiken $G = (V, \Sigma, P, S)$

Definition

- ① G heißt vom Typ 3 oder regulär, falls für alle Regeln $u \rightarrow v$ gilt:

$$u \in V \text{ und } v \in \Sigma V \cup \Sigma \cup \{\varepsilon\}$$

(d.h. alle Regeln haben die Form $A \rightarrow aB$, $A \rightarrow a$ oder $A \rightarrow \varepsilon$)

- ② G heißt vom Typ 2 oder kontextfrei, falls für alle Regeln $u \rightarrow v$ gilt:

$$u \in V \quad (\text{d.h. alle Regeln haben die Form } A \rightarrow v)$$

- ③ G heißt vom Typ 1 oder kontextsensitiv, falls für alle Regeln $u \rightarrow v$ gilt:

$$|v| \geq |u| \quad (\text{mit Ausnahme der } \varepsilon\text{-Sonderregel, s. unten})$$

- ④ Jede Grammatik ist automatisch vom Typ 0

Die ε -Sonderregel

In einer kontextsensitiven Grammatik ist auch die Regel $S \rightarrow \varepsilon$ zulässig, falls das Startsymbol S nicht auf der rechten Seite einer Regel vorkommt

Bemerkung

- Es ist klar, dass jede reguläre Grammatik auch kontextfrei ist
- Zudem ist die Sprache $L = \{a^n b^n \mid n \geq 0\}$ nicht regulär
- Es ist aber leicht, eine kontextfreie Grammatik für L anzugeben:

$$G = (\{S\}, \{a, b\}, P, S) \text{ mit } P = \{S \rightarrow aSb, \varepsilon\}$$

- Also gilt $\text{REG} \not\subseteq \text{CFL}$
- Allerdings sind nicht alle kontextfreien Grammatiken kontextsensitiv
- Z.B. ist obige Grammatik G nicht kontextsensitiv, da sie die Regel $S \rightarrow \varepsilon$ enthält und S auf der rechten Seite der Regel $S \rightarrow aSb$ vorkommt
- Wir können G jedoch wie folgt in eine Grammatik G' umwandeln:
 - ersetze die Regel $S \rightarrow \varepsilon$ durch die Regel $S \rightarrow ab$ und
 - füge ein neues Startsymbol S' sowie die Regeln $S' \rightarrow S, \varepsilon$ hinzu
- Tatsächlich lässt sich jede kontextfreie Grammatik G in eine äquivalente kontextfreie Grammatik G' umwandeln, die auch kontextsensitiv ist

Definition

Eine Grammatik $G = (V, \Sigma, P, S)$ ist in **Chomsky-Normalform (CNF)**, falls $P \subseteq V \times (V^2 \cup \Sigma)$ ist, d.h. alle Regeln haben die Form $A \rightarrow BC$ oder $A \rightarrow a$

Satz

Zu jeder kontextfreien Grammatik G lässt sich eine CNF-Grammatik G' mit $L(G') = L(G) \setminus \{\varepsilon\}$ konstruieren

Korollar

CFL \subseteq CSL

Beweis

- Sei $L \in \text{CFL}$ und sei $G = (V, \Sigma, P, S)$ eine CNF-Grammatik mit $L(G) = L \setminus \{\varepsilon\}$
- Im Fall $\varepsilon \notin L$ folgt sofort $L = L(G) \in \text{CSL}$, da G kontextsensitiv ist
- Ist $\varepsilon \in L$, so erzeugt folgende kontextsensitive (und kontextfreie) Grammatik G' die Sprache $L = L(G) \cup \{\varepsilon\}$:

$$G' = (V \cup \{S_{neu}\}, \Sigma, P \cup \{S_{neu} \rightarrow S, \varepsilon\}, S_{neu})$$



- Der Beweis des Pumping-Lemmas für kontextfreie Sprachen basiert auf CNF-Grammatiken
- Zudem ermöglichen sie einen effizienten Algorithmus zur Lösung des Wortproblems für kontextfreie Sprachen

Das Pumping-Lemma für kontextfreie Sprachen

Zu jeder kontextfreien Sprache $L \in \text{CFL}$ gibt es eine Zahl l , so dass sich alle Wörter $z \in L$ mit $|z| \geq l$ in $z = uvwxy$ zerlegen lassen mit

- 1 $vx \neq \varepsilon$,
- 2 $|vwx| \leq l$ und
- 3 $uv^iwx^iy \in L$ für alle $i \geq 0$

Das Wortproblem für kontextfreie Grammatiken

Gegeben: Eine kontextfreie Grammatik G und ein Wort x

Gefragt: Ist $x \in L(G)$?

Beispiel

- Betrachte die Sprache $L = \{a^n b^n \mid n \geq 0\}$
- Dann lässt sich jedes Wort $z = a^n b^n = a^{n-1} a b b^{n-1}$ in L mit $|z| \geq l = 2$ pumpen
- Zerlegen wir nämlich z in

$$z = uvwxy \text{ mit } u = a^{n-1}, v = a, w = \varepsilon, x = b \text{ und } y = b^{n-1},$$

dann gilt

- ① $vx = ab \neq \varepsilon$
- ② $|vwx| = |ab| \leq 2$ und
- ③ $uv^i wx^i y = a^{n-1} a^i b^i b^{n-1} \in L$ für alle $i \geq 0$



Beispiel

- Die Sprache $L = \{a^n b^n c^n \mid n \geq 0\}$ ist nicht kontextfrei
- Für eine vorgegebene Zahl $l \geq 0$ hat nämlich das Wort $z = a^l b^l c^l \in L$ die Länge $|z| = 3l \geq l$
- Dieses Wort lässt sich aber nicht pumpen:

Für jede Zerlegung $z = uvwxy$ mit $vx \neq \varepsilon$ und $|vwx| \leq l$ gehört $z' = uv^0wx^0y$ nicht zu L :

- Wegen $vx \neq \varepsilon$ ist $|z'| < |z|$
- Wegen $|vwx| \leq l$ kommen in vx nicht alle drei Zeichen a, b, c vor
- Kommt aber in vx beispielsweise kein a vor, so ist $\#_a(z) = \#_a(z')$ und somit gilt

$$|z'| < |z| = 3 \#_a(z) = 3 \#_a(z')$$

- Also gehört z' nicht zu L



Abschlusseigenschaften von CFL

Satz

CFL ist abgeschlossen unter Vereinigung, Produkt und Sternhülle

Beweis

- Seien $G_1 = (V_1, \Sigma, P_1, S_1)$ und $G_2 = (V_2, \Sigma, P_2, S_2)$ kontextfreie Grammatiken mit $V_1 \cap V_2 = \emptyset$ und sei S eine neue Variable
- Dann gilt
 - $L(G_1) \cup L(G_2) = L(G_3)$ für die kontextfreie Grammatik

$$G_3 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1, S_2\}, S)$$
 - $L(G_1)L(G_2) = L(G_4)$ für die kontextfreie Grammatik

$$G_4 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1S_2\}, S)$$
 und
 - $L(G_1)^* = L(G_5)$ für die kontextfreie Grammatik

$$G_5 = (V_1 \cup \{S\}, \Sigma, P_1 \cup \{S \rightarrow S_1S, \varepsilon\}, S)$$

□

Für $G_6 = (V_1, \Sigma, P_1 \cup \{S_1 \rightarrow S_1S_1, \varepsilon\}, S_1)$ muss nicht $L(G_6) = L(G_1)^*$ gelten, da $L(G_6)$ z.B. für $P_1 = \{S_1 \rightarrow aS_1b, \varepsilon\}$ das Wort $aababb \notin L(G_1)^*$ enthält

Satz

CFL ist nicht abgeschlossen unter Schnitt und Komplement

Beweis von $L_1, L_2 \in \text{CFL} \not\Rightarrow L_1 \cap L_2 \in \text{CFL}$

- Folgende Sprachen sind kontextfrei (siehe Übungen):

$$L_1 = \{a^n b^m c^m \mid n, m \geq 0\} \quad \text{und} \quad L_2 = \{a^n b^n c^m \mid n, m \geq 0\}$$

- Nicht jedoch ihr Schnitt $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ □

Beweis von $L \in \text{CFL} \not\Rightarrow \bar{L} \in \text{CFL}$

- Wäre CFL unter Komplement abgeschlossen, so wäre CFL wegen de Morgan auch unter Schnitt abgeschlossen
- Mit $A, B \in \text{CFL}$ wären dann nämlich auch $\bar{A}, \bar{B} \in \text{CFL}$, woraus wegen

$$\bar{A}, \bar{B} \in \text{CFL} \Rightarrow \overline{\bar{A} \cap \bar{B}} = \overline{\bar{A} \cap \bar{B}} \in \text{CFL}$$

wiederum $A \cap B \in \text{CFL}$ folgen würde □

Umwandlung in Chomsky-Normalform

Satz

Zu jeder kontextfreien Grammatik G lässt sich eine CNF-Grammatik G' mit $L(G') = L(G) \setminus \{\varepsilon\}$ konstruieren

Beweis

Wir wandeln $G = (V, \Sigma, P, S)$ wie folgt in eine CNF-Grammatik G' um:

- Wir beseitigen zunächst alle Regeln der Form $A \rightarrow \varepsilon$ und danach alle Regeln der Form $A \rightarrow B$ (siehe folgende Folien)
- Dann fügen wir für jedes Terminal $a \in \Sigma$ eine neue Variable X_a und eine neue Regel $X_a \rightarrow a$ hinzu und ersetzen jedes Vorkommen von a , bei dem a nicht alleine auf der rechten Seite einer Regel steht, durch X_a
- Anschließend führen wir für jede Regel $A \rightarrow B_1 \dots B_k$, $k \geq 3$, neue Variablen A_1, \dots, A_{k-2} ein und ersetzen sie durch die $k - 1$ Regeln

$$A \rightarrow B_1 A_1, A_1 \rightarrow B_2 A_2, \dots, A_{k-3} \rightarrow B_{k-2} A_{k-2}, A_{k-2} \rightarrow B_{k-1} B_k \quad \square$$

Falls G Regeln mit vielen Variablen auf der rechten Seite hat, empfiehlt es sich, Regeln der Form $A \rightarrow \varepsilon$ und $A \rightarrow B$ zuletzt zu beseitigen (s. Übungen)

Beseitigung von ε -Regeln

Satz

Zu jeder kontextfreien Grammatik $G = (V, \Sigma, P, S)$ gibt es eine kontextfreie Grammatik $G' = (V, \Sigma, P', S)$ ohne ε -Regeln mit $L(G') = L(G) \setminus \{\varepsilon\}$

Beweis

- Zuerst berechnen wir die Menge $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$ aller Variablen, die nach ε ableitbar sind:

```

1    $E' := \{A \in V \mid A \rightarrow \varepsilon\}$ 
2   repeat
3      $E := E'$ 
4      $E' := E \cup \{A \in V \mid \exists B_1, \dots, B_k \in E : A \rightarrow B_1 \dots B_k\}$ 
5   until  $E = E'$ 
  
```

- Nun bilden wir P' wie folgt:

$$\left\{ A \rightarrow v' \mid \begin{array}{l} \text{es ex. eine Regel } A \rightarrow_G v, \text{ so dass } v' \neq \varepsilon \text{ aus } v \text{ durch} \\ \text{Entfernen von beliebig vielen Variablen } A \in E \text{ entsteht} \end{array} \right\} \quad \square$$

Beseitigung von ε -Regeln

Beispiel

Betrachte die Grammatik $G = (\{S, T, U, X, Y, Z\}, \{a, b, c\}, P, S)$ mit

$$\begin{array}{lll}
 P: & S \rightarrow aY, bX, Z & Y \rightarrow bS, aYY & T \rightarrow U \\
 & X \rightarrow aS, bXX & Z \rightarrow \varepsilon, S, T, cZ & U \rightarrow abc
 \end{array}$$

- Berechnung von E :

E'	$\{Z\}$	$\{Z, S\}$
E	$\{Z, S\}$	$\{Z, S\}$

- Entferne $Z \rightarrow \varepsilon$ und füge die Regeln $Y \rightarrow b$ (wegen $Y \rightarrow bS$), $X \rightarrow a$ (wegen $X \rightarrow aS$) und $Z \rightarrow c$ (wegen $Z \rightarrow cZ$) hinzu:

$$\begin{array}{lll}
 P': & S \rightarrow aY, bX, Z & Y \rightarrow b, bS, aYY & T \rightarrow U \\
 & X \rightarrow a, aS, bXX & Z \rightarrow c, S, T, cZ & U \rightarrow abc
 \end{array}$$

Beseitigung von Variablenumbenennungen

Satz

Zu jeder kontextfreien Grammatik $G = (V, \Sigma, P, S)$ gibt es eine kontextfreie Grammatik $G' = (V, \Sigma, P', S)$ ohne Regeln der Form $A \rightarrow B$ mit $L(G') = L(G)$

Beweis

- Zuerst entfernen wir sukzessive alle Zyklen $A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_k \rightarrow A_1$
- Hierzu entfernen wir diese Regeln aus P und ersetzen alle Vorkommen der Variablen A_2, \dots, A_k in den übrigen Regeln durch A_1
- Befindet sich die Startvariable unter A_1, \dots, A_k , so sei dies o.B.d.A. A_1
- Nun eliminieren wir sukzessive die restlichen Variablenumbenennungen, indem wir
 - eine Regel $A \rightarrow B$ wählen, so dass in P keine Variablenumbenennung $B \rightarrow C$ mit B auf der linken Seite existiert,
 - diese Regel $A \rightarrow B$ aus P entfernen und
 - für jede Regel $B \rightarrow v$ in P die Regel $A \rightarrow v$ zu P hinzunehmen □

Beseitigung von Variablenumbenennungen

Beispiel (Fortsetzung)

$$P: \quad S \rightarrow aY, bX, Z \quad Y \rightarrow b, bS, aYY \quad T \rightarrow U \\ X \rightarrow a, aS, bXX \quad Z \rightarrow c, S, T, cZ \quad U \rightarrow abc$$

- Entferne den Zyklus $S \rightarrow Z \rightarrow S$ und ersetze Z durch S :

$$S \rightarrow aY, bX, c, T, cS \quad Y \rightarrow b, bS, aYY \quad T \rightarrow U \\ X \rightarrow a, aS, bXX \quad U \rightarrow abc$$

- Ersetze die Regel $T \rightarrow U$ durch $T \rightarrow abc$ (wegen $U \rightarrow abc$):

$$S \rightarrow aY, bX, c, T, cS \quad Y \rightarrow b, bS, aYY \quad T \rightarrow abc \\ X \rightarrow a, aS, bXX \quad U \rightarrow abc$$

- Ersetze dann auch die Regel $S \rightarrow T$ durch $S \rightarrow abc$ (wegen $T \rightarrow abc$):

$$S \rightarrow abc, aY, bX, c, cS \quad Y \rightarrow b, bS, aYY \quad T \rightarrow abc \\ X \rightarrow a, aS, bXX \quad U \rightarrow abc$$

- Da T und U nirgends mehr auf der rechten Seite vorkommen, können wir die Regeln $T \rightarrow abc$ und $U \rightarrow abc$ weglassen:

$$S \rightarrow abc, aY, bX, c, cS \quad Y \rightarrow b, bS, aYY \quad X \rightarrow a, aS, bXX$$

Beispiel (Schluss)

Betrachte die Grammatik $G = (\{S, X, Y, Z\}, \{a, b, c\}, P, S)$ mit

$$P: S \rightarrow abc, aY, bX, c, cS \quad Y \rightarrow b, bS, aYY \quad X \rightarrow a, aS, bXX$$

- Ersetze a , b und c durch A , B und C (außer wenn sie alleine auf der rechten Seite einer Regel stehen) und füge die Regeln $A \rightarrow a$, $B \rightarrow b$, $C \rightarrow c$ hinzu:

$$S \rightarrow ABC, AY, BX, c, CS \quad Y \rightarrow b, BS, AYY \quad X \rightarrow a, AS, BXX$$

$$A \rightarrow a \quad B \rightarrow b \quad C \rightarrow c$$

- Ersetze die Regeln $S \rightarrow ABC$, $Y \rightarrow AYY$ und $X \rightarrow BXX$ durch die Regeln $S \rightarrow AS'$, $S' \rightarrow BC$, $Y \rightarrow AY'$, $Y' \rightarrow YY$ und $X \rightarrow BX'$, $X' \rightarrow XX$:

$$S \rightarrow AS', AY, BX, c, CS \quad S' \rightarrow BC \quad Y \rightarrow b, BS, AY' \quad Y' \rightarrow YY$$

$$X \rightarrow a, AS, BX' \quad X' \rightarrow XX \quad A \rightarrow a \quad B \rightarrow b \quad C \rightarrow c$$



Links- und Rechtsableitungen

Definition

Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik

- Eine Ableitung

$$\underline{S} \Rightarrow l_1 \underline{A_1} r_1 \Rightarrow \dots \Rightarrow l_{m-1} \underline{A_{m-1}} r_{m-1} \Rightarrow \alpha_m$$

heißt **Linksableitung** von α_m (kurz $S \Rightarrow_L^* \alpha_m$), falls in jedem Ableitungsschritt die am weitesten links stehende Variable ersetzt wird, d.h. es gilt $l_i \in \Sigma^*$ für $i = 1, \dots, m-1$

- **Rechtsableitungen** $S_0 \Rightarrow_R^* \alpha_m$ sind analog definiert
- G heißt **mehrdeutig**, wenn es ein Wort $x \in L(G)$ gibt, das mindestens zwei verschiedene Linksableitungen hat
- Andernfalls heißt G **eindeutig**

Für alle $x \in \Sigma^*$ gilt: $x \in L(G) \Leftrightarrow S \Rightarrow^* x \Leftrightarrow S \Rightarrow_L^* x \Leftrightarrow S \Rightarrow_R^* x$

Ein- und mehrdeutige Grammatiken

Beispiel

- In $G = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, \varepsilon\}, S)$ gibt es **8** Ableitungen für $aabb$:

$$\underline{S} \Rightarrow_L a\underline{S}bS \Rightarrow_L aa\underline{S}bSbS \Rightarrow_L aab\underline{S}bS \Rightarrow_L aabb\underline{S} \Rightarrow_L aabb$$

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aabSb\underline{S} \Rightarrow aab\underline{S}b \Rightarrow aabb$$

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSbS \Rightarrow aaSbb\underline{S} \Rightarrow aa\underline{S}bb \Rightarrow aabb$$

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSb\underline{S} \Rightarrow aa\underline{S}bSb \Rightarrow aab\underline{S}b \Rightarrow aabb$$

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}bSb\underline{S} \Rightarrow aaSb\underline{S}b \Rightarrow aa\underline{S}bb \Rightarrow aabb$$

$$\underline{S} \Rightarrow aSb\underline{S} \Rightarrow a\underline{S}b \Rightarrow aa\underline{S}bSb \Rightarrow aab\underline{S}b \Rightarrow aabb$$

$$\underline{S} \Rightarrow_R aSb\underline{S} \Rightarrow_R a\underline{S}b \Rightarrow_R aaSb\underline{S}b \Rightarrow_R aa\underline{S}bb \Rightarrow_R aabb$$

- Darunter sind genau eine **Links-** und genau eine **Rechtsableitung**
- In $G' = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, ab, \varepsilon\}, S)$ gibt es **3** Ableitungen für ab :

$$\underline{S} \Rightarrow ab$$

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow ab\underline{S} \Rightarrow ab$$

$$\underline{S} \Rightarrow aSb\underline{S} \Rightarrow a\underline{S}b \Rightarrow ab$$

- Darunter sind **zwei Links-** und **zwei Rechtsableitungen**

Ein- und mehrdeutige Grammatiken

Beispiel

- Die Grammatik $G = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, \varepsilon\}, S)$ ist eindeutig
- Dies liegt daran, dass keine Satzform von G das Teilwort Sa enthält
- Daher kann in einer Linksableitung

$$S \Rightarrow_L^* y\underline{S}\beta \Rightarrow_L^* yz = x$$

auf die aktuelle Satzform $y\underline{S}\beta$ nicht die Regel $S \rightarrow \varepsilon$ angewandt werden, wenn in x auf das Präfix y ein a folgt

- Daher muss auf die aktuelle Satzform $y\underline{S}\beta$ genau dann die Regel $S \rightarrow aSbS$ angewandt werden, wenn in x auf das Präfix y ein a folgt
- Dagegen ist die Grammatik $G' = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, ab, \varepsilon\}, S)$ mehrdeutig, da das Wort $x = ab$ zwei Linksableitungen hat:

$$\underline{S} \Rightarrow ab \quad \text{und} \quad \underline{S} \Rightarrow a\underline{S}bS \Rightarrow ab\underline{S} \Rightarrow ab$$



Gerichtete Bäume und Wälder

Sei $G = (V, E)$ ein Digraph.

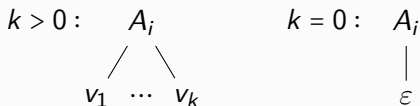
- Ein (gerichteter) v_0 - v_k -Weg in G ist eine Folge von Knoten v_0, \dots, v_k mit $(v_i, v_{i+1}) \in E$ für $i = 0, \dots, k-1$. Seine Länge ist k
- Ein Weg heißt Pfad, falls alle Knoten paarweise verschieden sind
- Ein u - v -Weg der Länge ≥ 1 mit $u = v$ heißt Zyklus
- G heißt azyklisch, wenn es in G keinen Zyklus gibt
- Ein Zyklus heißt Kreis, falls alle Knoten paarweise verschieden sind
- G heißt gerichteter Wald, wenn G azyklisch ist und jeder Knoten $v \in V$ Eingangsgrad $\deg^-(v) \leq 1$ hat
- Ein Knoten $u \in V$ vom Ausgangsgrad $\deg^+(u) = 0$ heißt Blatt
- Ein Knoten $w \in V$ heißt Wurzel, wenn $\deg^-(w) = 0$ ist
- Ein gerichteter Wald mit genau einer Wurzel heißt gerichteter Baum
- Da in einem gerichteten Baum alle Kanten von der Wurzel w wegführen, ist die Angabe der Kantenrichtungen bei Kenntnis von w überflüssig. Man spricht dann auch von einem Wurzelbaum

Wir ordnen einer Ableitung

$$\underline{A_0} \Rightarrow l_1 \underline{A_1} r_1 \Rightarrow \dots \Rightarrow l_{m-1} \underline{A_{m-1}} r_{m-1} \Rightarrow \alpha_m$$

den **Syntaxbaum** (oder **Ableitungsbaum**, engl. *parse tree*) T_m zu, wobei die Bäume T_0, \dots, T_m induktiv wie folgt definiert sind:

- T_0 besteht aus einem einzigen Knoten, der mit A_0 markiert ist
- Wird im $(i+1)$ -ten Ableitungsschritt die Regel $A_i \rightarrow v_1 \dots v_k$ mit $v_1, \dots, v_k \in \Sigma \cup V$ angewandt, so entsteht T_{i+1} aus T_i , indem wir das Blatt A_i durch folgenden Unterbaum ersetzen:



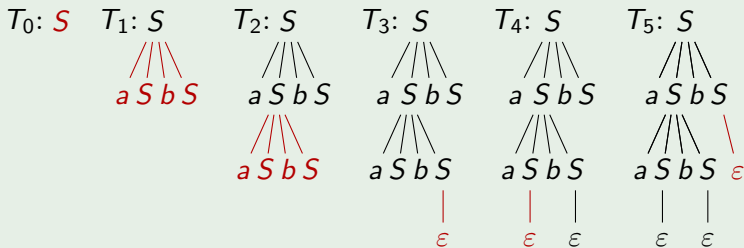
- Hierbei stellen wir uns die Kanten von oben nach unten gerichtet und die Kinder $v_1 \dots v_k$ von links nach rechts geordnet vor
- Syntaxbäume sind also **geordnete** Wurzelbäume

Beispiel

- Betrachte die Grammatik $G = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, \varepsilon\}, S)$ und die Ableitung

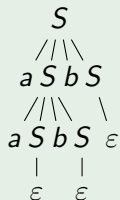
$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aa\underline{S}b\underline{S}bS \Rightarrow aa\underline{S}bbS \Rightarrow aabb\underline{S} \Rightarrow aabb$$

- Die zugehörigen Syntaxbäume sind dann

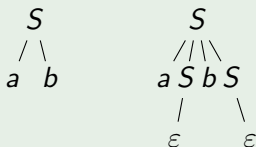


Beispiel

- In $G = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, \varepsilon\}, S)$ führen alle acht Ableitungen des Wortes $aabb$ auf denselben Syntaxbaum:



- Dagegen führen in $G' = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, ab, \varepsilon\}, S)$ die drei Ableitungen des Wortes ab auf zwei unterschiedliche Syntaxbäume:



Syntaxbäume und Linksableitungen

- Seien T_0, \dots, T_m die zu einer Ableitung $S = \alpha_0 \Rightarrow \dots \Rightarrow \alpha_m$ gehörigen Syntaxbäume
- Dann haben alle Syntaxbäume T_0, \dots, T_m die Wurzel S
- Die Satzform α_i ergibt sich aus T_i , indem wir die Blätter von T_i von links nach rechts zu einem Wort zusammensetzen
- Auf den Syntaxbaum T_m führen neben $\alpha_0 \Rightarrow \dots \Rightarrow \alpha_m$ alle Ableitungen, die sich von dieser nur in der Reihenfolge der Regelanwendungen unterscheiden
- Dazu gehört genau eine Linksableitung
- Linksableitungen und Syntaxbäume entsprechen sich also eineindeutig
- Dasselbe gilt für Rechtsableitungen
- Ist T Syntaxbaum einer CNF-Grammatik, so hat jeder Knoten in T höchstens zwei Kinder (d.h. T ist ein **Binärbaum**)

Definition

Die **Tiefe** eines Baumes mit Wurzel w ist die maximale Länge eines Weges von w zu einem Blatt

Lemma

Ein Binärbaum B der Tiefe $\leq k$ hat $\leq 2^k$ Blätter

Beweis durch Induktion über k :

$k = 0$: Ein Baum der Tiefe 0 kann nur einen Knoten haben

$k \rightsquigarrow k + 1$: Sei B ein Binärbaum der Tiefe $\leq k + 1$

Dann hängen an B 's Wurzel maximal zwei Unterbäume

Da deren Tiefe $\leq k$ ist, haben sie nach IV $\leq 2^k$ Blätter

Also hat $B \leq 2^{k+1}$ Blätter



Lemma

Ein Binärbaum B der Tiefe $\leq k$ hat $\leq 2^k$ Blätter

Korollar

Ein Binärbaum B mit $> 2^{k-1}$ Blättern hat eine Tiefe $\geq k$

Beweis

Wäre die Tiefe von B kleiner als k (also $\leq k - 1$), so hätte B nach obigem Lemma $\leq 2^{k-1}$ Blätter (Widerspruch) \square

Satz (Pumping-Lemma für kontextfreie Sprachen)

Zu jeder kontextfreien Sprache $L \in \text{CFL}$ gibt es eine Zahl l , so dass sich alle Wörter $z \in L$ mit $|z| \geq l$ in $z = uvwxy$ zerlegen lassen mit

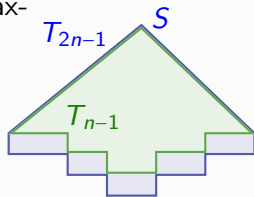
- 1 $vx \neq \varepsilon$,
- 2 $|vwx| \leq l$ und
- 3 $uv^iwx^iy \in L$ für alle $i \geq 0$

Beweis des Pumping-Lemmas für CFL

Beweis

- Sei $G = (V, \Sigma, P, S)$ eine CNF-Grammatik für $L \setminus \{\varepsilon\}$
- Ist nun $z = z_1 \dots z_n \in L$ mit $n \geq 1$, so ex. in G eine Ableitung

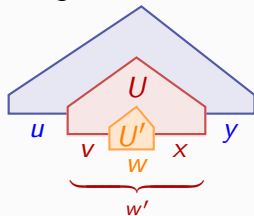
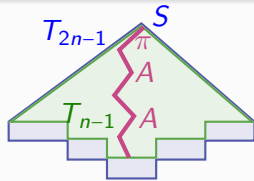
$$S = \alpha_0 \Rightarrow \dots \Rightarrow \alpha_m = z$$
mit zugehörigen Syntaxbäumen T_0, \dots, T_m
- Da G in CNF ist, werden hierbei genau $n - 1$ Regeln der Form $A \rightarrow BC$ und genau n Regeln der Form $A \rightarrow a$ angewandt
- Folglich ist $m = 2n - 1$ und wir können annehmen, dass die Regeln der Form $A \rightarrow BC$ vor den Regeln der Form $A \rightarrow a$ zur Anwendung kommen
- Dann besteht α_{n-1} aus n Variablen und die Syntaxbäume T_{2n-1} und T_{n-1} haben genau n Blätter
- Setzen wir $l = 2^k$, wobei $k = \|V\|$ ist, so hat T_{n-1} im Fall $n \geq l$ mindestens die Tiefe k , da T_{n-1} mindestens $l = 2^k > 2^{k-1}$ Blätter hat



Beweis des Pumping-Lemmas für CFL

Beweis (Fortsetzung)

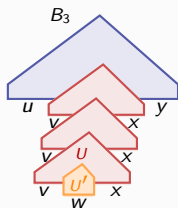
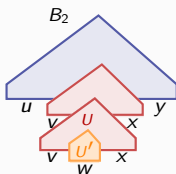
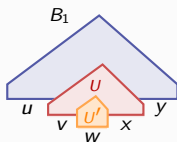
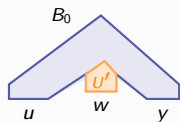
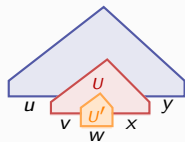
- Setzen wir $l = 2^k$, wobei $k = \|V\|$ ist, so hat T_{n-1} im Fall $n \geq l$ mindestens die Tiefe k , da T_{n-1} mindestens $l = 2^k > 2^{k-1}$ Blätter hat
- Sei π ein von der Wurzel ausgehender Pfad maximaler Länge in T_{n-1}
- Dann hat π mindestens die Länge k und unter den letzten $k + 1 > \|V\|$ Knoten von π müssen zwei mit derselben Variablen A markiert sein
- Seien U und U' die Unterbäume von T_{2n-1} mit diesen Knoten als Wurzel
- Dann hat U höchstens $l = 2^k$ Blätter und U' hat weniger Blätter als U
- Nun zerlegen wir z wie folgt:
 - w' ist das Teilwort von $z = uw'y$, das von U erzeugt wird und
 - w ist das Teilwort von $w' = vwx$, das von U' erzeugt wird



Beweis des Pumping-Lemmas für CFL

Beweis (Schluss)

- Dann ist $vx \neq \varepsilon$ (Bed. 1), da U mehr Blätter als U' hat
- Zudem gilt $|vwx| \leq l$ (Bed. 2), da U höchstens $2^k = l$ Blätter hat (sonst hätte der Baum $U^* = U \cap T_{n-1}$ eine Tiefe größer k und π wäre nicht maximal)
- Schließlich lassen sich Syntaxbäume B_i für die Wörter uv^iwx^iy , $i \geq 0$, wie folgt konstruieren (Bed. 3):
 - B_0 entsteht aus $B_1 = T_{2n-1}$, indem wir U durch U' ersetzen
 - B_{i+1} entsteht aus B_i , indem wir U' durch U ersetzen:



Das Wortproblem für kontextfreie Grammatiken

Gegeben: Eine kontextfreie Grammatik G und ein Wort x

Gefragt: Ist $x \in L(G)$?

Frage

Wie lässt sich das Wortproblem für kontextfreie Grammatiken entscheiden?

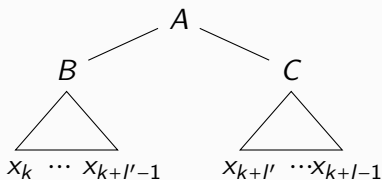
- Sei eine Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $x = x_1 \dots x_n$ gegeben
- Falls $x = \varepsilon$ ist, können wir effizient prüfen, ob $S \Rightarrow^* \varepsilon$ gilt
- Hierzu genügt es, die Menge $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$ aller ε -ableitbaren Variablen zu berechnen und zu prüfen, ob $S \in E$ ist
- Andernfalls bringen wir G in CNF und starten den nach seinen Autoren **Cocke**, **Younger** und **Kasami** benannten **CYK-Algorithmus**
- Dieser bestimmt mittels **dynamischer Programmierung** für $l = 1, \dots, n$ und $k = 1, \dots, n - l + 1$ die Menge $V_{l,k}$ aller Variablen, aus denen das Teilwort $x_k \dots x_{k+l-1}$ ableitbar ist
- Dann gilt $x \in L(G) \Leftrightarrow S \in V_{n,1}$

Berechnung der Mengen $V_{l,k}$

- Sei $G = (V, \Sigma, P, S)$ eine CNF-Grammatik und sei $x \in \Sigma^+$
- Dann lassen sich die Mengen $V_{l,k} = \{A \in V \mid A \Rightarrow^* x_k \dots x_{k+l-1}\}$ wie folgt bestimmen
- Für $l = 1$ gehört A zu $V_{1,k}$, falls die Regel $A \rightarrow x_k$ existiert:

$$V_{1,k} = \{A \in V \mid A \rightarrow x_k\}$$

- Für $l > 1$ gehört A zu $V_{l,k}$, falls eine Regel $A \rightarrow BC$ und eine Zahl $l' \in \{1, \dots, l-1\}$ ex. mit $B \in V_{l',k}$ und $C \in V_{l-l',k+l'}$:



$$V_{l,k} = \{A \in V \mid \exists l' < l, B \in V_{l',k}, C \in V_{l-l',k+l'}: A \rightarrow BC \in P\}$$

Algorithmus CYK(G, x)

```
1  Input: CNF-Grammatik  $G = (V, \Sigma, P, S)$  und Wort  $x = x_1 \dots x_n$ 
2  for  $k := 1$  to  $n$  do
3       $V_{1,k} := \{A \in V \mid A \rightarrow x_k \in P\}$ 
4  for  $l := 2$  to  $n$  do
5      for  $k := 1$  to  $n - l + 1$  do
6           $V_{l,k} := \emptyset$ 
7          for  $l' := 1$  to  $l - 1$  do
8              for all  $A \rightarrow BC \in P$  do
9                  if  $B \in V_{l',k}$  and  $C \in V_{l-l',k+l'}$  then
10                      $V_{l,k} := V_{l,k} \cup \{A\}$ 
11  if  $S \in V_{n,1}$  then accept else reject
```

Der CYK-Algorithmus lässt sich dahingehend erweitern, dass er im Fall $x \in L(G)$ auch einen Syntaxbaum T von x bestimmt

Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX,$
 $Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c$

- Dann erhalten wir für das Wort $x = abb$ folgende Mengen $V_{l,k}$:

$k:$	1	2	3
	a	b	b
$l: 1$	{X, A}	{Y, B}	{Y, B}
2	{S}	{Y'}	
3	{Y}		

- Wegen $S \notin V_{3,1}$ ist $x \notin L(G)$

Der CYK-Algorithmus

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

$P: S \rightarrow AS', AY, BX, CS, c, S' \rightarrow BC, X \rightarrow AS, BX', a, X' \rightarrow XX, Y \rightarrow BS, AY', b, Y' \rightarrow YY, A \rightarrow a, B \rightarrow b, C \rightarrow c$

- Dagegen gehört das Wort $y = aababb$ zu $L(G)$:

	a	a	b	a	b	b
	$\{X, A\}$	$\{X, A\}$	$\{Y, B\}$	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$
	$\{X'\}$	$\{S\}$	$\{S\}$	$\{S\}$	$\{Y'\}$	
	$\{X\}$	$\{X\}$	$\{Y\}$	$\{Y\}$		
	$\{X'\}$	$\{S\}$	$\{Y'\}$			
	$\{X\}$	$\{Y\}$				
	$\{S\}$					

Frage

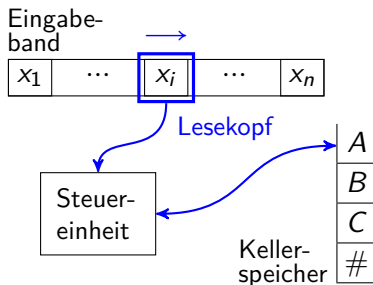
Wie lässt sich das Maschinenmodell des DFA erweitern, um die Sprache

$$L = \{a^n b^n \mid n \geq 0\}$$

und alle anderen kontextfreien Sprachen erkennen zu können?

Antwort

- Ein DFA kann Sprachen wie L nicht erkennen, da er nur seinen Zustand als Speicher benutzen kann und die Anzahl der Zustände zwar von L aber nicht von der Eingabe abhängen darf
- Um kontextfreie Sprachen erkennen zu können, genügt bereits ein **Kellerspeicher** (auch **Stapel**, engl. *stack* oder *pushdown memory*)
- Dieser erlaubt nur den Zugriff auf die höchste belegte Speicheradresse



- verfügt zusätzlich über einen Kellerspeicher
- kann auch ε -Übergänge machen
- hat Lesezugriff auf das aktuelle Eingabezeichen und auf das oberste Kellersymbol
- kann in jedem Rechenschritt das oberste Kellersymbol löschen und durch beliebig viele Symbole ersetzen

Notation

Für eine Menge M bezeichne $\mathcal{P}_e(M)$ die Menge aller endlichen Teilmengen von M , d.h. $\mathcal{P}_e(M) = \{A \subseteq M \mid A \text{ ist endlich}\}$

Definition

Ein **Kellerautomat mit Endzuständen** (auch **FS-PDA** für engl. *Final State PushDown Automaton*) ist ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$. Dabei ist

- $Z \neq \emptyset$ eine endliche Menge von **Zuständen**
- Σ das **Eingabealphabet**
- Γ das **Kelleralphabet**
- $\delta : Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow \mathcal{P}_e(Z \times \Gamma^*)$ die **Überföhrungsfunktion**
- $q_0 \in Z$ der **Startzustand**
- $\# \in \Gamma$ das **Kelleranfangszeichen**
- $E \subseteq Z$ die Menge der **Endzustände**

- Wenn p der momentane Zustand, A das oberste Kellerzeichen und $u \in \Sigma$ das nächste Eingabezeichen (bzw. $u = \varepsilon$) ist, so kann M im Fall $(q, B_1 \dots B_k) \in \delta(p, u, A)$
 - in den Zustand q wechseln,
 - den Lesekopf auf dem Eingabeband um $|u| \in \{0, 1\}$ Positionen vorrücken und
 - das Zeichen A aus- sowie die Zeichenfolge $B_1 \dots B_k$ einkellern (danach ist B_1 das oberste Kellerzeichen)
- Hierfür sagen wir auch, M führt die Anweisung $puA \rightarrow qB_1 \dots B_k$ aus, und sprechen im Fall
 - $k = 0$ von einer **pop-Operation**,
 - $k = 2$ und $B_2 = A$ von einer **push-Operation**, sowie
 - im Fall $u = \varepsilon$ von einem **ε -Übergang**
- Man beachte, dass bei leerem Keller kein Übergang mehr möglich ist

Ein Kellerautomat mit Endzuständen

Beispiel

- Sei $M = (Z, \Sigma, \Gamma, \delta, p, \#, E)$ mit $Z = \{p, q, r\}$, $\Sigma = \{a, b\}$, $\Gamma = \{A, \#\}$, $E = \{r\}$ und der Überföhrungsfunktion

$$\delta : pa\# \rightarrow pA\# \quad (1)$$

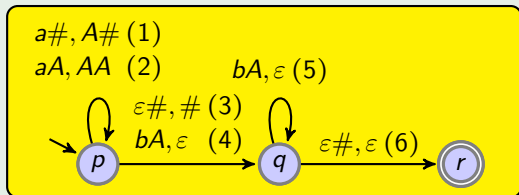
$$paA \rightarrow pAA \quad (2)$$

$$p\epsilon\# \rightarrow q\# \quad (3)$$

$$pbA \rightarrow q \quad (4)$$

$$qbA \rightarrow q \quad (5)$$

$$q\epsilon\# \rightarrow r \quad (6)$$



Konfiguration eines Kellerautomaten

- Eine **Konfiguration** von M wird durch ein Tripel

$$K = (p, x_i \dots x_n, A_1 \dots A_l) \in Z \times \Sigma^* \times \Gamma^*$$

beschrieben und besagt, dass

- p der momentane Zustand,
 - $x_i \dots x_n$ der ungelesene Rest der Eingabe und
 - $A_1 \dots A_l$ der aktuelle Kellerinhalt ist (A_1 ist oberstes Symbol)
- In einer solchen Konfiguration K kann M eine beliebige Anweisung $puA_1 \rightarrow qB_1 \dots B_k$ mit $u \in \{\varepsilon, x_i\}$ ausführen
 - Diese überführt M in die **Folgekonfiguration**

$$K' = (q, x_j \dots x_n, B_1 \dots B_k A_2 \dots A_l) \text{ mit } j = i + |u|$$

- Hierfür schreiben wir auch kurz $K \vdash K'$
- Eine **Rechnung** von M bei Eingabe x ist eine Folge von Konfigurationen

$$K_0, K_1, K_2 \dots \text{ mit } K_0 \vdash K_1 \vdash K_2 \dots ,$$

wobei $K_0 = (q_0, x, \#)$ die **Startkonfiguration** von M bei Eingabe x ist

Notation

Die reflexive, transitive Hülle von \vdash bezeichnen wir wie üblich mit \vdash^*

Definition

Die von einem Kellerautomaten $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ **akzeptierte (oder erkannte) Sprache** ist

$$L(M) = \{x \in \Sigma^* \mid \exists q \in E, \alpha \in \Gamma^* : (q_0, x, \#) \vdash^* (q, \varepsilon, \alpha)\}$$

- Ein Kellerautomat M mit Endzuständen akzeptiert also genau dann ein Wort x , wenn es bei dieser Eingabe eine Rechnung gibt, bei der M
 - alle Eingabezeichen liest und
 - einen Endzustand $q \in E$ erreicht

Ein Kellerautomat mit Endzuständen

Beispiel (Fortsetzung)

- Sei $M = (Z, \Sigma, \Gamma, \delta, p, \#, E)$ mit $Z = \{p, q, r\}$, $\Sigma = \{a, b\}$, $\Gamma = \{A, \#\}$, $E = \{r\}$ und der Überföhrungsfunktion

$$\delta : pa\# \rightarrow pA\# \quad (1)$$

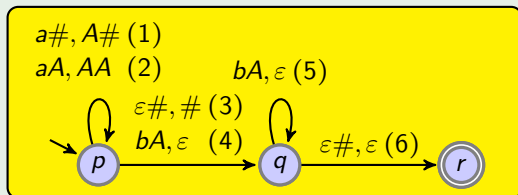
$$paA \rightarrow pAA \quad (2)$$

$$p\epsilon\# \rightarrow q\# \quad (3)$$

$$pbA \rightarrow q \quad (4)$$

$$qbA \rightarrow q \quad (5)$$

$$q\epsilon\# \rightarrow r \quad (6)$$



- Dann akzeptiert M die Eingabe $x = aabb$:

$$\begin{array}{ccccccc}
 (p, aabb, \#) & \vdash_{(1)} & (p, abb, A\#) & \vdash_{(2)} & (p, bb, AA\#) & \vdash_{(4)} & (q, b, A\#) & \vdash_{(5)} & (q, \epsilon, \#) \\
 & & & & & & & & \\
 & & & & & & & & \vdash_{(6)} & (r, \epsilon, \epsilon)
 \end{array}$$

Es gibt auch ein Akzeptanzkriterium, das keine Endzustände erfordert.

Definition

- Sei $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ ein Kellerautomat ohne Endzustandsmenge
- Die von M durch Leeren des Kellers akzeptierte (oder erkannte) Sprache ist

$$L(M) = \{x \in \Sigma^* \mid \exists p \in Z: (q_0, x, \#) \vdash^* (p, \varepsilon, \varepsilon)\}$$

- Wir nennen M auch einen **ES-PDA** (für engl. *Empty Stack PushDown Automaton*) oder einfach **PDA**
- Ein PDA M akzeptiert also genau dann eine Eingabe, wenn es eine Rechnung gibt, bei der M alle Eingabezeichen liest und den Keller leert
- Es gilt (siehe Übungen)

$$\{L(M) \mid M \text{ ist ein FS-PDA}\} = \{L(M) \mid M \text{ ist ein ES-PDA}\}$$

Ein PDA für die Sprache $\{a^n b^n \mid n \geq 0\}$

Beispiel

- Sei $M = (Z, \Sigma, \Gamma, \delta, p, \#)$ mit $Z = \{p, q\}$,
 $\Sigma = \{a, b\}$, $\Gamma = \{A, \#\}$ und

$$\delta : p\varepsilon\# \rightarrow p \quad (1) \quad pa\# \rightarrow pA \quad (2) \quad paA \rightarrow pAA \quad (3) \\ pbA \rightarrow q \quad (4) \quad qbA \rightarrow q \quad (5)$$

- Dann akzeptiert M die Eingabe $x = aabb$:

$$(p, aabb, \#) \xrightarrow{(2)} (p, abb, A) \xrightarrow{(3)} (p, bb, AA) \xrightarrow{(4)} (q, b, A) \xrightarrow{(5)} (q, \varepsilon, \varepsilon)$$

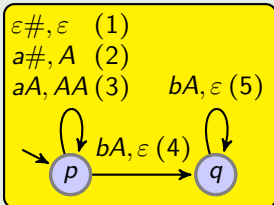
- Allgemeiner akzeptiert M das Wort $x = a^n b^n$ mit folgender Rechnung:

$$n = 0: (p, \varepsilon, \#) \xrightarrow{(1)} (p, \varepsilon, \varepsilon)$$

$$n \geq 1: (p, a^n b^n, \#) \xrightarrow{(2)} (p, a^{n-1} b^n, A) \xrightarrow{(3)^{n-1}} (p, b^n, A^n)$$

$$\xrightarrow{(4)} (q, b^{n-1}, A^{n-1}) \xrightarrow{(5)^{n-1}} (q, \varepsilon, \varepsilon)$$

- Dies zeigt, dass M alle Wörter der Form $a^n b^n$, $n \geq 0$, akzeptiert



Ein PDA für die Sprache $\{a^n b^n \mid n \geq 0\}$

Beispiel (Fortsetzung)

- Sei $M = (Z, \Sigma, \Gamma, \delta, p, \#)$ mit $Z = \{p, q\}$,
 $\Sigma = \{a, b\}$, $\Gamma = \{A, \#\}$ und

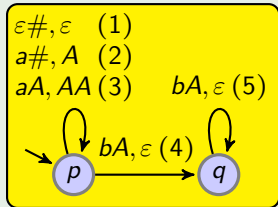
$$\delta : p\epsilon\# \rightarrow p \quad (1) \quad pa\# \rightarrow pA \quad (2) \quad paA \rightarrow pAA \quad (3)$$

$$pbA \rightarrow q \quad (4) \quad qbA \rightarrow q \quad (5)$$

- Es gilt auch die umgekehrte Inklusion:

alle Wörter $x = x_1 \dots x_m \in L(M)$ haben die Form $x = a^n b^n$

- Ausgehend von der Startkonfiguration $(p, x, \#)$ sind nämlich nur die Anweisungen (1) oder (2) ausführbar
- Führt M zuerst Anweisung (1) aus, so wird der Keller geleert
- Daher kann M in diesem Fall nur das leere Wort $x = \epsilon = a^0 b^0$ akzeptieren
- Falls M mit Anweisung (2) beginnt, muss M später mittels Anweisung (4) in den Zustand q gelangen, da sonst der Keller nicht geleert wird



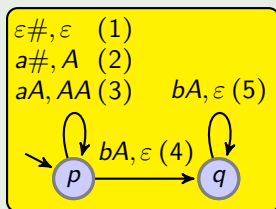
Ein PDA für die Sprache $\{a^n b^n \mid n \geq 0\}$

Beispiel (Schluss)

- Sei $M = (Z, \Sigma, \Gamma, \delta, p, \#)$ mit $Z = \{p, q\}$,
 $\Sigma = \{a, b\}$, $\Gamma = \{A, \#\}$ und

$$\delta : p\epsilon\# \rightarrow p \quad (1) \quad pa\# \rightarrow pA \quad (2) \quad paA \rightarrow pAA \quad (3)$$

$$pbA \rightarrow q \quad (4) \quad qbA \rightarrow q \quad (5)$$



- Falls M mit Anweisung (2) beginnt, muss M später mittels Anweisung (4) in den Zustand q gelangen, da sonst der Keller nicht geleert wird
- Dies geschieht, sobald M nach Lesen von $n \geq 1$ a 's das erste b liest:

$$(p, x_1 x_2 \dots x_m, \#) \stackrel{(2)}{\vdash} (p, x_2 \dots x_n x_{n+1} \dots x_m, A)$$

$$\stackrel{(3)}{\vdash} \dots \stackrel{(3)}{\vdash} (p, x_{n+1} x_{n+2} \dots x_m, A^n) \stackrel{(4)}{\vdash} (q, x_{n+2} \dots x_m, A^{n-1})$$

mit $x_1 = x_2 = \dots = x_n = a$ und $x_{n+1} = b$.

- Damit der Keller nach dem Lesen von x_m leer ist, muss M nun noch genau $n - 1$ b 's lesen, weshalb $m = 2n$ und $x = a^n b^n$ folgt. \triangleleft

Ziel

Als nächstes wollen wir zeigen, dass PDAs genau die kontextfreien Sprachen erkennen

Satz

$$\text{CFL} = \{L(M) \mid M \text{ ist ein PDA}\}$$

Beweis von $\text{CFL} \subseteq \{L(M) \mid M \text{ ist ein PDA}\}$ **Idee:**

Konstruiere zu einer kontextfreien Grammatik $G = (V, \Sigma, P, S)$ einen PDA $M = (\{z\}, \Sigma, \Gamma, \delta, z, S)$ mit $\Gamma = V \cup \Sigma$ und folgenden Anweisungen:

- für jede Regel $A \rightarrow_G \alpha$ die Anweisung $z \in A \rightarrow z\alpha$
- für jedes Zeichen $a \in \Sigma$ die Anweisung $z a a \rightarrow z \varepsilon$

Beweis von $CFL \subseteq \{L(M) \mid M \text{ ist ein PDA}\}$

Beispiel

- Betrachte die Grammatik $G = (\{S\}, \{a, b\}, P, S)$ mit den Regeln

$$P: S \rightarrow aSb \quad (1) \quad S \rightarrow \varepsilon \quad (2)$$

- Der zugehörige PDA besitzt dann die Anweisungen

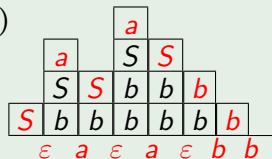
$$\delta: zaa \rightarrow z \quad (0) \quad zbb \rightarrow z \quad (0') \quad z\varepsilon S \rightarrow zaSb \quad (1') \quad z\varepsilon S \rightarrow z \quad (2')$$

- Der Linksableitung $\underline{S} \xRightarrow{(1)} a\underline{S}b \xRightarrow{(1)} aa\underline{S}bb \xRightarrow{(2)} aabb$ in G entspricht dann die Rechnung

$$(z, aabb, S) \underset{(1')}{\vdash} (z, aabb, aSb) \underset{(0)}{\vdash} (z, abb, Sb)$$

$$\underset{(1')}{\vdash} (z, abb, aSbb) \underset{(0)}{\vdash} (z, bb, Sbb)$$

$$\underset{(2')}{\vdash} (z, bb, bb) \underset{(0')}{\vdash} (z, b, b) \underset{(0')}{\vdash} (z, \varepsilon, \varepsilon)$$



von M und umgekehrt

Idee:

Konstruiere zu einer kontextfreien Grammatik $G = (V, \Sigma, P, S)$ einen PDA $M = (\{z\}, \Sigma, \Gamma, \delta, z, S)$ mit $\Gamma = V \cup \Sigma$ und folgenden Anweisungen:

- für jede Regel $A \rightarrow_G \alpha$ die Anweisung $z \varepsilon A \rightarrow z \alpha$
- für jedes Zeichen $a \in \Sigma$ die Anweisung $z a a \rightarrow z \varepsilon$

- M versucht also, eine Linksableitung für die Eingabe x zu finden. Da M hierbei den Syntaxbaum von oben nach unten aufbaut, wird M als *Top-Down Parser* bezeichnet
- Zudem gilt $S \Rightarrow_L^m x_1 \dots x_n$ gdw. $(z, x_1 \dots x_n, S) \vdash^{m+n} (z, \varepsilon, \varepsilon)$
- Daher folgt

$$x \in L(G) \Leftrightarrow S \Rightarrow_L^* x \Leftrightarrow (z, x, S) \vdash^* (z, \varepsilon, \varepsilon) \Leftrightarrow x \in L(M)$$



Beweis von $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \text{CFL}$ **Vorbetrachtung:**

- Obige Konstruktion eines PDA M aus einer kontextfreien Grammatik lässt sich leicht umdrehen, falls M nur einen Zustand hat
- Zu einem solchen PDA $M = (\{z\}, \Sigma, \Gamma, \delta, z, \#)$ lässt sich wie folgt eine kontextfreie Grammatik $G = (V, \Sigma, P, X_{\#})$ mit $L(G) = L(M)$ konstruieren:

- Die Variablenmenge von G ist $V = \{X_A \mid A \in \Gamma\}$
(wir können auch o.B.d.A. $\Sigma \cap \Gamma = \emptyset$ annehmen und $V = \Gamma$ setzen)
- die Startvariable von G ist $X_{\#}$ und
- P enthält für jede Anweisung $zuA \rightarrow zA_1 \dots A_k$ von M die Regel

$$X_A \rightarrow uX_{A_1} \dots X_{A_k}$$

- Dann lässt sich jede akzeptierende Rechnung $(z, x, \#) \vdash^m (z, \varepsilon, \varepsilon)$ von $M(x)$ der Länge m direkt in eine Linksableitung $X_{\#} \Rightarrow_L^m x$ in G der Länge m transformieren und umgekehrt

Beweis von $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \text{CFL}$

Beispiel

- Betrachte den PDA $M = (\{z\}, \{a, b\}, \{S, a, b\}, \delta, z, S)$ mit

$$\delta: zaa \rightarrow z \quad (1) \quad zbb \rightarrow z \quad (2) \quad z \in S \rightarrow zaSb \quad (3) \quad z \in S \rightarrow z \quad (4)$$

den wir aus der Grammatik $G = (\{S\}, \{a, b\}, P, S)$ mit den beiden Regeln $S \rightarrow aSb, \varepsilon$ konstruiert haben

- Dann führt M auf die Grammatik $G' = (\{X_S, X_a, X_b\}, \{a, b\}, P', X_S)$ mit

$$P': X_a \rightarrow a \quad (1') \quad X_b \rightarrow b \quad (2') \quad X_S \rightarrow X_a X_S X_b \quad (3') \quad X_S \rightarrow \varepsilon \quad (4')$$

- Der Rechnung

$$(z, ab, S) \underset{(3)}{\vdash} (z, ab, aSb) \underset{(1)}{\vdash} (z, b, Sb) \underset{(4)}{\vdash} (z, b, b) \underset{(2)}{\vdash} (z, \varepsilon, \varepsilon)$$

von M entspricht dann folgende Linksableitung in G' (und umgekehrt):

$$\underline{X_S} \underset{(3')}{\Rightarrow} \underline{X_a X_S X_b} \underset{(1')}{\Rightarrow} \underline{a X_S X_b} \underset{(4')}{\Rightarrow} \underline{a X_b} \underset{(2')}{\Rightarrow} \underline{ab}$$

- Man beachte, dass G' eine aufgeblähte Variante von G ist.

Idee:

Transformiere einen PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ wie folgt in einen äquivalenten PDA $M' = (\{z\}, \Sigma, \Gamma', \delta', z, S)$ mit nur einem Zustand:

- Das Kellularphabet ist $\Gamma' = \{S\} \cup \{X_{pAq} \mid A \in \Gamma, p, q \in Z\}$ und
- die Überföhrungsfunktion δ' enthalt folgende Anweisungen:
 - fur jeden Zustand $q \in Z$ die Anweisung $z \in S \rightarrow zX_{q_0\#q}$ und
 - fur jede Anweisung $p_0 u A_0 \rightarrow p_1 A_1 \dots A_k$, $k \geq 0$, von M und fur jede Folge von k Zustanden $p_2, \dots, p_{k+1} \in Z$ die Anweisung

$$z u X_{p_0 A_0 p_{k+1}} \rightarrow z X_{p_1 A_1 p_2} \dots X_{p_k A_k p_{k+1}}$$

- Dabei rat M' durch die Wahl der Anweisung
 - $z \in S \rightarrow zX_{q_0\#q}$ den Zustand q , den M mit leerem Keller (also im letzten Rechenschritt) erreicht, und
 - $z u X_{p_0 A_0 p_{k+1}} \rightarrow z X_{p_1 A_1 p_2} \dots X_{p_k A_k p_{k+1}}$ fur $i = 1, \dots, k - 1$ die Zustande p_{i+1} , die M bei Freigabe der mit A_i belegten Speicherzelle erreicht
- Zudem verifiziert M' bei jeder pop-Operation $z u X_{p_0 A_0 p_1} \rightarrow z$, dass M den (geratenen) Zustand p_1 nach Entfernen von A_0 auch erreichen kann

Beispiel

- Betrachte den PDA $M = (\{p, q\}, \{a, b\}, \{A, \#\}, \delta, p, \#)$ mit den Anweisungen

$$\begin{array}{lll} \delta : p\varepsilon\# \rightarrow q & (1) & pa\# \rightarrow pA & (2) & paA \rightarrow pAA & (3) \\ & & pbA \rightarrow q & (4) & qbA \rightarrow q & (5) \end{array}$$

- Der zugehörige PDA $M' = (\{z\}, \{a, b\}, \Gamma', \delta', z, S)$ mit nur einem Zustand hat dann das Kelleralphabet

$$\Gamma' = \{S, X_{p\#\#}, X_{p\#q}, X_{q\#\#}, X_{q\#q}, X_{pAp}, X_{pAq}, X_{qAp}, X_{qAq}\}$$

Beispiel (Fortsetzung)

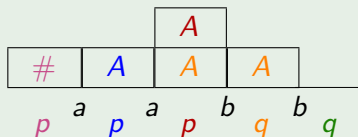
- Zudem enthält M' neben den beiden Anweisungen ~~$z\epsilon S \rightarrow zX_{p\#p}$ (0)~~ und $z\epsilon S \rightarrow zX_{p\#q}$ (0')

Anweisung von M	k	p_2, \dots, p_{k+1}	Anweisungen von M'
$p\epsilon\# \rightarrow q$	(1)	0	$z\epsilon X_{p\#q} \rightarrow z$ (1')
$pa\# \rightarrow pA$	(2)	1	$z\epsilon X_{p\#p} \rightarrow zX_{pAp}$ (2') $z\epsilon X_{p\#q} \rightarrow zX_{pAq}$ (2'')
$paA \rightarrow pAA$	(3)	2	$z\epsilon X_{pAp} \rightarrow zX_{pAp}X_{pAp}$ (3') $z\epsilon X_{pAq} \rightarrow zX_{pAp}X_{pAq}$ (3'') $z\epsilon X_{pAp} \rightarrow zX_{pAq}X_{qAp}$ (3''') $z\epsilon X_{pAq} \rightarrow zX_{pAq}X_{qAq}$ (3''')
$pbA \rightarrow q$	(4)	0	$z\epsilon X_{pAq} \rightarrow z$ (4')
$qbA \rightarrow q$	(5)	0	$z\epsilon X_{qAq} \rightarrow z$ (5')

Beispiel (Schluss)

- Der (akzeptierenden) Rechnung

$$(p, aabb, \#) \xrightarrow{(2)} (p, abb, A) \xrightarrow{(3)} (p, bb, AA) \xrightarrow{(4)} (q, b, A) \xrightarrow{(5)} (q, \varepsilon, \varepsilon)$$



von M entspricht dann folgende Rechnung von M' :

$$(z, aabb, S) \xrightarrow{(0')} (z, aabb, X_{p\#q}) \xrightarrow{(2'')} (z, abb, X_{pAq})$$

$$\xrightarrow{(3''')} (z, bb, X_{pAq}X_{qAq}) \xrightarrow{(4')} (z, b, X_{qAq}) \xrightarrow{(5')} (z, \varepsilon, \varepsilon)$$

