

Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2013/14

Bemerkung

- Wie wir gesehen haben, ist folgende Sprache nicht regulär:

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

- Es ist aber leicht, eine kontextfreie Grammatik für L zu finden:

$$G = (\{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow \varepsilon\}, S).$$

- Damit ist klar, dass die Klasse der regulären Sprachen echt in der Klasse der kontextfreien Sprachen enthalten ist:

$$\text{REG} \subsetneq \text{CFL}.$$

- Als nächstes wollen wir zeigen, dass die Klasse der kontextfreien Sprachen wiederum echt in der Klasse der kontextsensitiven Sprachen enthalten ist:

$$\text{CFL} \subsetneq \text{CSL}.$$

- Kontextfreie Grammatiken sind dadurch charakterisiert, dass sie nur Regeln der Form $A \rightarrow \alpha$ haben.
- Dies lässt die Verwendung von beliebigen ε -Regeln der Form $A \rightarrow \varepsilon$ zu.
- Eine kontextsensitive Grammatik darf dagegen höchstens die ε -Regel $S \rightarrow \varepsilon$ haben.
- Voraussetzung hierfür ist, dass S das Startsymbol ist und dieses nicht auf der rechten Seite einer Regel vorkommt.
- Daher sind nicht alle kontextfreien Grammatiken kontextsensitiv.
- Beispielsweise ist die Grammatik $G = (\{S\}, \{a, b\}, \{S \rightarrow aSb, S \rightarrow \varepsilon\}, S)$ nicht kontextsensitiv, da sie die Regel $S \rightarrow \varepsilon$ enthält, obwohl S auf der rechten Seite der Regel $S \rightarrow aSb$ vorkommt.
- Wir werden jedoch sehen, dass sich zu jeder kontextfreien Grammatik eine äquivalente kontextsensitive Grammatik konstruieren lässt.

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 *ε -ableitbaren* Variablen:

```

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:

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

Entfernen von ε -Regeln

Beispiel

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

$$P: \quad S \rightarrow aY, bX, Z; \quad Y \rightarrow bS, aYY; \quad T \rightarrow U; \\ X \rightarrow aS, bXX; \quad Z \rightarrow \varepsilon, S, T, cZ; \quad U \rightarrow abc.$$

- Berechnung von E :

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

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

$$P': \quad S \rightarrow aY, bX, Z; \quad Y \rightarrow b, bS, aYY; \quad T \rightarrow U; \\ X \rightarrow a, aS, bXX; \quad Z \rightarrow c, S, T, cZ; \quad U \rightarrow abc.$$



Satz

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

Korollar

$\text{REG} \not\subseteq \text{CFL} \subseteq \text{CSL} \subseteq \text{RE}$.

Beweis

- Es ist nur noch die Inklusion $\text{CFL} \subseteq \text{CSL}$ zu zeigen.
- Nach obigem Satz ex. zu $L \in \text{CFL}$ eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ ohne ε -Regeln mit $L(G) = L \setminus \{\varepsilon\}$.
- Da G dann auch kontextsensitiv ist, folgt hieraus im Fall $\varepsilon \notin L$ unmittelbar $L(G) = L \in \text{CSL}$.
- Im Fall $\varepsilon \in L$ erzeugt die kontextsensitive Grammatik

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

die Sprache $L(G') = L$, d.h. $L \in \text{CSL}$.

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 erzeugen die kontextfreien Grammatiken

$$G_3 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1, S_2\}, S)$$

die Vereinigung $L(G_3) = L(G_1) \cup L(G_2)$,

$$G_4 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S)$$

das Produkt $L(G_4) = L(G_1)L(G_2)$ und

$$G_5 = (V_1 \cup \{S\}, \Sigma, P_1 \cup \{S \rightarrow S_1 S, \varepsilon\}, S)$$

die Sternhülle $L(G_1)^*$. □

Satz

CFL ist abgeschlossen unter Vereinigung, Produkt und Sternhülle.

Frage

Ist die Klasse CFL auch abgeschlossen unter

- Schnitt und
- Komplement?

Antwort

Nein.

Hierzu müssen wir für bestimmte Sprachen nachweisen, dass sie nicht kontextfrei sind. Dies gelingt mit einem Pumping-Lemma für kontextfreie Sprachen, für dessen Beweis wir Grammatiken in Chomsky-Normalform benötigen.

Definition

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

Anwendungen der Chomsky-Normalform

- CNF-Grammatiken bilden die Basis für eine effiziente Lösung des Wortproblems für kontextfreie Sprachen.
- Zudem ermöglichen sie den Beweis des Pumping-Lemmas für kontextfreie Sprachen.

Satz (Pumping-Lemma für kontextfreie Sprachen)

Zu jeder kontextfreien Sprache L 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 $i \geq 0$.

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 $|z| \geq 2$ pumpen:
 - Zerlege z in $z = uvwxy$ mit $u = a^{n-1}$, $v = a$, $w = \varepsilon$, $x = b$, $y = b^{n-1}$.
 - Dann ist für alle $i \geq 0$ das Wort $uv^iwx^iy = a^{n-1} a^i b^i b^{n-1} \in L$.



Beispiel

- Die Sprache $\{a^n b^n c^n \mid n \geq 0\}$ ist nicht kontextfrei.
- Für eine vorgegebene Zahl $l \geq 0$ hat nämlich $z = a^l b^l c^l$ 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 $z' = uv^2wx^2y$ nicht zu L :

- Wegen $vx \neq \varepsilon$ ist $|z| < |z'|$.
- Wegen $|vwx| \leq l$ kann in vx nicht jedes der drei Zeichen a, b, c vorkommen.
- Kommt aber in vx beispielsweise kein a vor, so ist

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

- Also kann z' nicht zu L gehören.



Das Wortproblem für kontextfreie Grammatiken

Gegeben: Eine kontextfreie Grammatik G und ein Wort x .

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

Satz

Das Wortproblem für kontextfreie Grammatiken ist effizient entscheidbar.

Um eine kontextfreie Grammatik in Chomsky-Normalform zu bringen, müssen wir neben den ε -Regeln $A \rightarrow \varepsilon$ auch sämtliche Variablenumbenennungen $A \rightarrow B$ loswerden.

Definition

Regeln der Form $A \rightarrow B$ heißen **Variablenumbenennungen**.

Satz

Zu jeder kontextfreien Grammatik G ex. eine kontextfreie Grammatik G' ohne Variablenumbenennungen 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 .

(Sollte sich unter den entfernten Variablen A_2, \dots, A_k die Startvariable S befinden, so sei A_1 die neue Startvariable.)

Beispiel (Fortsetzung)

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

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

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

Entfernen von Variablenumbenennungen

Satz

Zu jeder kontextfreien Grammatik G ex. eine kontextfreie Grammatik G' ohne Variablenumbenennungen mit $L(G') = L(G)$.

Beweis

- Zuerst entfernen wir alle Zyklen

$$A_1 \rightarrow A_2 \rightarrow \dots \rightarrow A_k \rightarrow A_1.$$

- Nun werden wir sukzessive die restlichen Variablenumbenennungen los, 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 $B \rightarrow \alpha$ in P die Regel $A \rightarrow \alpha$ zu P hinzunehmen. \square

Beispiel (Fortsetzung)

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

- Entferne die Regel $T \rightarrow U$ und füge die Regel $T \rightarrow abc$ hinzu (wegen $U \rightarrow abc$):

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

- Entferne dann auch die Regel $S \rightarrow T$ und füge die Regel $S \rightarrow abc$ (wegen $T \rightarrow abc$) hinzu:

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

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

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

Bereits gezeigt:

Korollar

Zu jeder kontextfreien Grammatik G ex. eine kontextfreie Grammatik G' ohne ε -Regeln und ohne Variablenumbenennungen mit $L(G') = L(G) \setminus \{\varepsilon\}$.

Noch zu zeigen:

Satz

Zu jeder kontextfreien Sprache $L \in \text{CFL}$ gibt es eine CNF-Grammatik G' mit $L(G') = L \setminus \{\varepsilon\}$.

Umwandlung in Chomsky-Normalform

Satz

Zu jeder kontextfreien Sprache $L \in \text{CFL}$ gibt es eine CNF-Grammatik G' mit $L(G') = L \setminus \{\varepsilon\}$.

Beweis

- Sei $G = (V, \Sigma, P, S)$ eine kontextfreie Grammatik ohne ε -Regeln und ohne Variablenumbenennungen für $L \setminus \{\varepsilon\}$.
- Wir transformieren G wie folgt in eine CNF-Grammatik.
- Füge für jedes Terminalsymbol $a \in \Sigma$ eine neue Variable X_a zu V und eine neue Regel $X_a \rightarrow a$ zu P hinzu.
- Ersetze alle Vorkommen von a durch X_a , außer wenn a alleine auf der rechten Seite einer Regel steht.
- Ersetze jede Regel $A \rightarrow B_1 \dots B_k$, $k \geq 3$, durch die $k - 1$ Regeln

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

wobei A_1, \dots, A_{k-2} neue Variablen sind. □

Beispiel (Fortsetzung)

- Betrachte die Regeln

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

- Ersetze a , b und c durch A , B und C (außer wenn sie alleine rechts vorkommen) und füge die Regeln $A \rightarrow a$, $B \rightarrow b$, $C \rightarrow c$ hinzu:

$$S \rightarrow ABC, AY, BX, CS, c; \quad X \rightarrow AS, BXX, a;$$

$$Y \rightarrow BS, AYY, b; \quad A \rightarrow a; \quad B \rightarrow b; \quad C \rightarrow c.$$

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

$$S \rightarrow AS', AY, BX, CS, c; \quad S' \rightarrow BC; \quad X \rightarrow AS, BX', a; \quad X' \rightarrow XX;$$

$$Y \rightarrow BS, AY', b; \quad Y' \rightarrow YY; \quad A \rightarrow a; \quad B \rightarrow b; \quad C \rightarrow c.$$


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 zwei verschiedene Linksableitungen hat. Andernfalls heißt G **eindeutig**.

Leicht zu sehen:

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

Beispiel

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

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

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

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

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

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

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

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

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


Beispiel

- Die Grammatik $G = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, \varepsilon\}, S)$ ist eindeutig.
- Dies liegt daran, dass in keiner Satzform von G die Variable S von einem a gefolgt wird.
- Daher muss jede Linksableitung eines Wortes $x \in L(G)$ die am weitesten links stehende Variable der aktuellen Satzform $\alpha S \beta$ genau dann nach $aSbS$ expandieren, falls das Präfix α in x von einem a gefolgt wird.
- Dagegen ist die Grammatik $G = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, ab, \varepsilon\}, S)$ mehrdeutig, da das Wort $x = ab$ 2 verschiedene Linksableitungen hat:

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



- Für jede Satzform $\alpha \in \{a, b, S\}^*$ von G gilt $\#_a(\alpha) = \#_b(\alpha)$ (*).
- Zudem gilt für jedes Präfix u von α : $\#_a(u) \geq \#_b(u)$ (**).
- Dies folgt durch Induktion über die Ableitungslänge l .
 - $l = 0$: Klar, da $\alpha = S$ beide Bedingungen (*, **) erfüllt.
 - $l \rightsquigarrow l + 1$: Gelte $S \Rightarrow^l \alpha \Rightarrow \beta$.
 - Falls β aus α durch Anwendung der Regel $S \rightarrow \varepsilon$ entsteht, ist dies ebenfalls klar.
 - Entsteht β aus α durch die Regel $S \rightarrow aSbS$, so folgt $\#_a(\beta) = \#_a(\alpha) + 1 = \#_b(\alpha) + 1 = \#_b(\beta)$, also (*). Zudem entspricht jedem Präfix u von β ein Präfix u' von α mit $\#_a(u) - \#_b(u) \geq \#_a(u') - \#_b(u')$, wodurch sich (**) von α auf β überträgt.
- Somit erfüllen alle Wörter $x \in L(G)$ die Bedingungen (*, **).
- Tatsächlich sind in G genau die Wörter ableitbar, die (*, **) erfüllen.

- Tatsächlich sind in G genau die Wörter ableitbar, die $(*, **)$ erfüllen.
- Dazu zeigen wir durch Induktion über n folgende Behauptung:
Alle Wörter $x \in \{a, b\}^*$ der Länge $\leq n$, die $(*, **)$ erfüllen, sind in G ableitbar.

$n = 0$: Klar, da $x = \varepsilon$ aus S ableitbar ist.

$n \rightsquigarrow n + 1$: Sei x ein Wort der Länge $n + 1$, das $(*, **)$ erfüllt und sei u das kürzeste Präfix von x mit $\#_a(u) = \#_b(u) \geq 1$.

- Dann muss u die Form $u = avb$ haben, wobei v $(*, **)$ erfüllt. Nach IV gilt daher $S \Rightarrow^* v$.
- Zudem hat x die Form $x = uw$, wobei auch w $(*, **)$ erfüllt. Nach IV gilt daher $S \Rightarrow^* w$.
- Nun ist x aus S wie folgt ableitbar:
 $S \Rightarrow aSbS \Rightarrow^* avbS = uS \Rightarrow^* uw = x$.

Gerichtete Bäume und Wälder

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

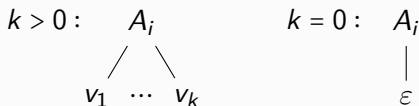
- Ein (**gerichteter**) v_0 - v_k -**Weg** in G ist eine Folge von Knoten v_0, \dots, v_k mit $(v_i, v_{i+1}) \in E$ für $i = 0, \dots, k-1$. Seine **Länge** ist k .
- Ein Weg heißt **Pfad**, falls alle Knoten paarweise verschieden sind.
- Ein u - v -Weg der Länge ≥ 1 mit $u = v$ heißt **Zyklus**.
- G heißt **azyklisch**, wenn es in G keinen Zyklus gibt.
- 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** von G , falls alle Knoten $v \in V$ von w aus erreichbar sind (d.h. es gibt einen gerichteten w - v -Weg in G).
- Ein **gerichteter Wald**, der eine Wurzel hat, heißt **gerichteter Baum**.
- Da die Kantenrichtungen durch die Wahl der Wurzel eindeutig bestimmt sind, kann auf ihre Angabe verzichtet werden. 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:



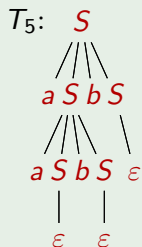
- Hierbei stellen wir uns die Kanten von oben nach unten gerichtet und die Kinder $v_1 \dots v_k$ von links nach rechts geordnet vor.
- Syntaxbäume sind also **geordnete** Wurzelbäume.

Beispiel

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

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

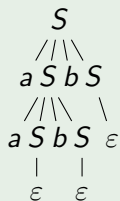
- Die zugehörigen Syntaxbäume sind dann



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

Beispiel

- In $G = (\{S\}, \{a, b\}, \{S \rightarrow aSbS, \epsilon\}, S)$ führen alle 8 Ableitungen auf denselben Syntaxbaum



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

Hätte B eine Tiefe $\leq k - 1$, so hätte B nach obigem Lemma $\leq 2^{k-1}$ Blätter (Widerspruch). \square

Als erste Anwendung der Chomsky-Normalform beweisen wir das Pumping-Lemma für kontextfreie Sprachen.

Satz (Pumping-Lemma für kontextfreie Sprachen)

Zu jeder kontextfreien Sprache L 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

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

- 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

$$S = \alpha_0 \Rightarrow \alpha_1 \dots \Rightarrow \alpha_m = z.$$

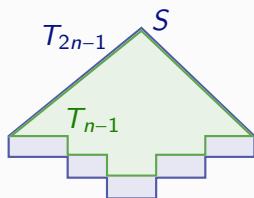
- Da G in CNF ist, werden hierbei genau $n - 1$ Regeln der Form $A \rightarrow BC$ und genau n Regeln der Form $A \rightarrow a$ angewandt.

Beweis des Pumping-Lemmas

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

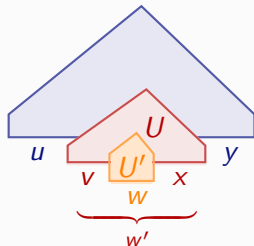
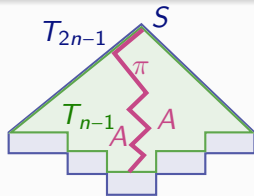
$$S = \alpha_0 \Rightarrow \alpha_1 \dots \Rightarrow \alpha_m = z.$$
- Da G in CNF ist, werden hierbei genau $n - 1$ Regeln der Form $A \rightarrow BC$ und genau n Regeln der Form $A \rightarrow a$ angewandt.
- Folglich ist $m = 2n - 1$ und z hat den Syntaxbaum T_{2n-1} .
- Wir können annehmen, dass die $n - 1$ Regeln der Form $A \rightarrow BC$ vor den n Regeln der Form $A \rightarrow a$ zur Anwendung kommen.
- Dann besteht α_{n-1} aus n Variablen und T_{n-1} hat wie T_{2n-1} genau n Blätter.
- Setzen wir $l = 2^k$, wobei $k = \|V\|$ ist, so hat T_{n-1} im Fall $n \geq l$ mindestens $l = 2^k > 2^{k-1}$ Blätter und daher mindestens die Tiefe k .



Beweis des Pumping-Lemmas

Beweis (Fortsetzung)

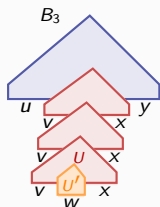
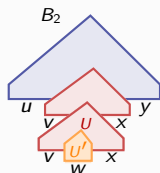
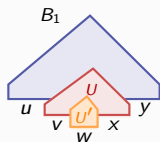
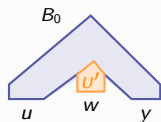
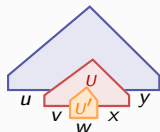
- Setzen wir $l = 2^k$, wobei $k = \|V\|$ ist, so hat T_{n-1} im Fall $n \geq l$ mindestens $l = 2^k > 2^{k-1}$ Blätter und daher mindestens die Tiefe k .
- Sei π ein von der Wurzel ausgehender Pfad maximaler Länge in T_{n-1} .
- Dann hat π mindestens die Länge k und unter den letzten $k + 1$ Knoten von π müssen zwei mit derselben Variablen A markiert sein.
- Seien U und U' die von diesen Knoten ausgehenden Unterbäume des vollständigen Syntaxbaums T_{2n-1} .
- 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

Beweis (Schluss)

- Da U mehr Blätter hat als U' , ist $vx \neq \varepsilon$ (Bed. 1).
- Da der Baum $U^* = U \cap T_{n-1}$ höchstens die Tiefe k hat (andernfalls wäre π nicht maximal), hat U^* (und damit U) höchstens $2^k = l$ Blätter.
- Folglich ist $|vwx| \leq l$ (Bed. 2).
- 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:



Satz (Pumping-Lemma für kontextfreie Sprachen)

Zu jeder kontextfreien Sprache L 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 $i \geq 0$.

Beispiel

- Die Sprache $\{a^n b^n c^n \mid n \geq 0\}$ ist nicht kontextfrei.
- Für eine vorgegebene Zahl $l \geq 0$ hat nämlich $z = a^l b^l c^l$ 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 $z' = uv^2wx^2y$ nicht zu L :

- Wegen $vx \neq \varepsilon$ ist $|z| < |z'|$.
- Wegen $|vwx| \leq l$ kann in vx nicht jedes der drei Zeichen a, b, c vorkommen.
- Kommt aber in vx beispielsweise kein a vor, so ist

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

- Also kann z' nicht zu L gehören.



Wie wir gesehen haben, ist die Klasse CFL abgeschlossen unter

- Vereinigung,
- Produkt und
- Sternhülle.

Satz

CFL ist nicht abgeschlossen unter

- Schnitt und
- Komplement.

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

- Die beiden Sprachen

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

sind kontextfrei (siehe Übungen).

- Nicht jedoch ihr Schnitt $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$.
- Also ist CFL nicht unter Schnitt abgeschlossen.

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

Da CFL zwar unter Vereinigung aber nicht unter Schnitt abgeschlossen ist, kann CFL wegen de Morgan nicht unter Komplement abgeschlossen sein.

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?

Satz

Das Wortproblem für kontextfreie Grammatiken ist effizient entscheidbar.

Beweis

- 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 $V_{l,k}$

Beweis (Fortsetzung)

- Sei $G = (V, \Sigma, P, S)$ eine CNF-Grammatik und sei $x \in \Sigma^+$.
- Dann lassen sich die Mengen

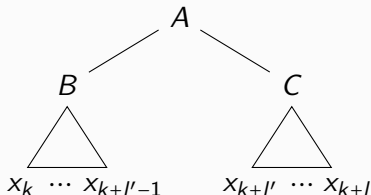
$$V_{l,k} = \{A \in V \mid A \Rightarrow^* x_k \dots x_{k+l-1}\}$$

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., so dass $B \in V_{l',k}$ und $C \in V_{l-l',k+l'}$ sind:



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

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 leicht dahingehend modifizieren, 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

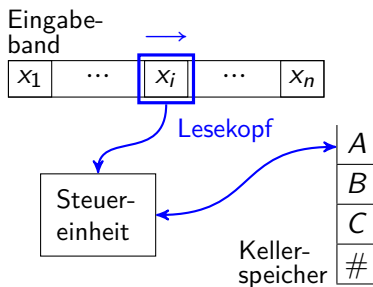
Wie lässt sich das Maschinenmodell des DFA erweitern, um die Sprache

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

und alle anderen kontextfreien Sprachen erkennen zu können?

Antwort

- Dass ein DFA die Sprache L nicht erkennen kann, liegt an seinem beschränkten Speichervermögen, das zwar von L aber nicht von der Eingabe abhängen darf.
- Um L erkennen zu können, genügt bereits ein so genannter **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 das oberste Kellersymbol löschen (durch eine **pop-Operation**) und
- durch beliebig viele Symbole ersetzen (durch eine **push-Operation**).

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** (kurz: **PDA**, engl. *pushdown automaton*) wird durch ein 6-Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ beschrieben, wobei

- $Z \neq \emptyset$ 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** und
- $\# \in \Gamma$ das **Kelleranfangszeichen** ist.

Arbeitsweise eines PDA

- 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.
- Im Fall $u = \varepsilon$ spricht man auch von einem **ε -Übergang**.

- Eine **Konfiguration** 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 der Konfiguration $K = (p, x_i \dots x_n, A_1 \dots A_l)$ kann M eine bel. 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 $K_0, K_1, K_2 \dots$ mit $K_0 = (q_0, x, \#)$ und $K_0 \vdash K_1 \vdash K_2 \dots$.
 K_0 heißt **Startkonfiguration** von M bei Eingabe x .

Notation

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

Definition

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

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

Bemerkung

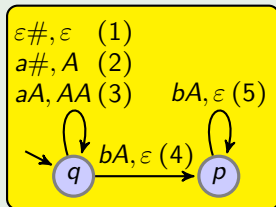
- Ein PDA M akzeptiert also genau dann eine Eingabe x , wenn es eine Rechnung gibt, bei der M
 - das gesamte Eingabewort bis zum Ende liest und
 - den Keller leert.
- Man beachte, dass bei leerem Keller kein weiterer Übergang mehr möglich ist.

Ein Kellerautomat

Beispiel

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

$$\delta : \begin{array}{l} q\varepsilon\# \rightarrow q \quad (1) \quad qa\# \rightarrow qA \quad (2) \quad qaA \rightarrow qAA \quad (3) \\ qbA \rightarrow p \quad (4) \quad pbA \rightarrow p \quad (5) \end{array}$$



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

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

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

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

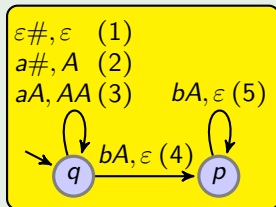
$$\begin{aligned} n \geq 1: (q, a^n b^n, \#) &\xrightarrow{(2)} (q, a^{n-1} b^n, A) \xrightarrow{(3)} (q, b^n, A^n) \\ &\xrightarrow{(4)} (p, b^{n-1}, A^{n-1}) \xrightarrow{(5)} (p, \varepsilon, \varepsilon). \end{aligned}$$

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

Ein Kellerautomat

Beispiel

- Sei $M = (Z, \Sigma, \Gamma, \delta, q, \#)$ mit $Z = \{q, p\}$,
 $\Sigma = \{a, b\}$, $\Gamma = \{A, \#\}$ und
 $\delta : q\epsilon\# \rightarrow q$ (1) $qa\# \rightarrow qA$ (2) $qaA \rightarrow qAA$ (3)
 $qbA \rightarrow p$ (4) $pbA \rightarrow p$ (5)

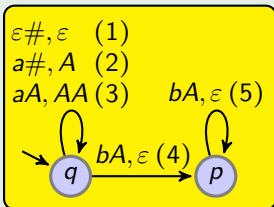


- Als nächstes zeigen wir, dass jede von M akzeptierte Eingabe $x = x_1 \dots x_n \in L(M)$ die Form $x = a^m b^m$ haben muss.
- Ausgehend von der Startkonfiguration $(q, x, \#)$ sind nur die Anweisungen (1) oder (2) ausführbar.
- Führt M zuerst Anweisung (1) aus, so wird der Keller geleert.
- Daher 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 p gelangen, da sonst der Keller nicht geleert wird.

Ein Kellerautomat

Beispiel

- Sei $M = (Z, \Sigma, \Gamma, \delta, q, \#)$ mit $Z = \{q, p\}$,
 $\Sigma = \{a, b\}$, $\Gamma = \{A, \#\}$ und
 $\delta : q\varepsilon\# \rightarrow q$ (1) $qa\# \rightarrow qA$ (2) $qaA \rightarrow qAA$ (3)
 $qbA \rightarrow p$ (4) $pbA \rightarrow p$ (5)



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

$$(q, x_1 \dots x_n, \#) \xrightarrow{(2)} (q, x_2 \dots x_n, A) \xrightarrow{(3)^{m-1}} (q, x_{m+1} \dots x_n, A^m) \\ \xrightarrow{(4)} (p, x_{m+2} \dots x_n, A^{m-1})$$

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

- Um den Keller leeren zu können, muss M nun noch genau $m - 1$ b 's lesen, weshalb x auch in diesem Fall die Form $a^m b^m$ haben muss. \triangleleft

Ziel

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

Satz

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

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 $M = (\{q\}, \Sigma, \Gamma, \delta, q, S)$ mit $\Gamma = V \cup \Sigma$, so dass folgende Äquivalenz gilt:

$$S \Rightarrow^* x_1 \dots x_n \text{ gdw. } (q, x_1 \dots x_n, S) \vdash^* (q, \varepsilon, \varepsilon).$$

- Hierzu fügen wir folgende Anweisungen zu δ hinzu:

für jede Regel $A \rightarrow_G \alpha$: $q \varepsilon A \rightarrow q \alpha$,

für jedes Zeichen $a \in \Sigma$: $q a a \rightarrow q \varepsilon$.

- 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.
- Dann gilt $S \Rightarrow_L^I x_1 \dots x_n$ gdw. $(q, x_1 \dots x_n, S) \vdash^{I+n} (q, \varepsilon, \varepsilon)$.
- Daher folgt

$$x \in L(G) \Leftrightarrow S \Rightarrow_L^* x \Leftrightarrow (q, x, S) \vdash^* (q, \varepsilon, \varepsilon) \Leftrightarrow x \in L(M). \quad \square$$

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

$$P: S \rightarrow aSbS \quad (1), \quad S \rightarrow a \quad (2).$$

- Der zugehörige PDA besitzt dann die Anweisungen

$$\delta: qaa \rightarrow q\varepsilon \quad (0) \quad qbb \rightarrow q\varepsilon \quad (0')$$

$$q\varepsilon S \rightarrow qaSbS \quad (1') \quad q\varepsilon S \rightarrow qa \quad (2')$$

- Der Linksableitung $\underline{S} \xRightarrow{(1)} aSbS \xRightarrow{(2)} aabS \xRightarrow{(2)} aaba$ in G entspricht dann die Rechnung

$$(q, aaba, S) \underset{(1')}{\vdash} (q, aaba, aSbS) \underset{(0)}{\vdash} (q, aba, SbS) \underset{(2')}{\vdash} (q, aba, abS)$$

$$\underset{(0)}{\vdash} (q, ba, bS) \underset{(0')}{\vdash} (q, a, S) \underset{(2')}{\vdash} (q, a, a) \underset{(0)}{\vdash} (q, \varepsilon, \varepsilon)$$

von M und umgekehrt.



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

Konstruiere zu einem PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit Variablen $X_{pAp'}$, $A \in \Gamma$, $p, p' \in Z$, so dass folgende Äquivalenz gilt:

$$X_{pAp'} \Rightarrow^* x \text{ gdw. } (p, x, A) \vdash^* (p', \varepsilon, \varepsilon).$$

- Ein Wort x soll also genau dann in G aus $X_{pAp'}$ ableitbar sein, wenn M ausgehend vom Zustand p bei Lesen von x in den Zustand p' gelangen kann und dabei das Zeichen A aus dem Keller entfernt.
- Hierzu fügen wir für jede Anweisung $puA \rightarrow p_0A_1 \dots A_k$, $k \geq 0$, die folgenden Regeln zu P hinzu:

$$\text{Für jede Zustandsfolge } p_1, \dots, p_k: X_{pAp_k} \rightarrow uX_{p_0A_1p_1} \dots X_{p_{k-1}A_kp_k}.$$

- Um damit alle Wörter $x \in L(M)$ aus S ableiten zu können, benötigen wir jetzt nur noch die Regeln

$$S \rightarrow X_{q_0\#p'}, p' \in Z.$$

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

Beispiel

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

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

- Dann erhalten wir die Grammatik $G = (V, \Sigma, P, S)$ mit der Variablenmenge

$$V = \{S, X_{p\#p}, X_{p\#q}, X_{q\#p}, X_{q\#q}, X_{pAp}, X_{pAq}, X_{qAp}, X_{qAq}\}.$$

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

Beispiel (Fortsetzung)

- P enthält neben den beiden Startregeln $S \rightarrow X_{p\#p}, X_{p\#q}$ ($0, 0'$) die folgenden Produktionen:

Anweisung	k	p_1, \dots, p_k	zugehörige Regeln
$p\epsilon\# \rightarrow q\epsilon$	(1)	0	- $X_{p\#q} \rightarrow \epsilon$ (1')
$pa\# \rightarrow pA$	(2)	1	p $X_{p\#p} \rightarrow aX_{pAp}$ (2') q $X_{p\#q} \rightarrow aX_{pAq}$ (2'')
$paA \rightarrow pAA$	(3)	2	p, p $X_{pAp} \rightarrow aX_{pAp}X_{pAp}$ (3') p, q $X_{pAq} \rightarrow aX_{pAp}X_{pAq}$ (3'') q, p $X_{pAp} \rightarrow aX_{pAq}X_{qAp}$ (3''') q, q $X_{pAq} \rightarrow aX_{pAq}X_{qAq}$ (3'''')
$pbA \rightarrow q\epsilon$	(4)	0	- $X_{pAq} \rightarrow b$ (4')
$qbA \rightarrow q\epsilon$	(5)	0	- $X_{qAq} \rightarrow b$ (5')

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

Beispiel (Schluss)

- Die Anweisungen

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

von M führen also auf die folgenden Regeln von G :

$$\begin{array}{ll} S \rightarrow X_{p\#p}, X_{p\#q} & (0, 0') \quad X_{p\#q} \rightarrow \varepsilon & (1') \\ X_{p\#p} \rightarrow aX_{pAp} & (2') \quad X_{p\#q} \rightarrow aX_{pAq} & (2'') \\ X_{pAp} \rightarrow aX_{pAp}X_{pAp} & (3') \quad X_{pAq} \rightarrow aX_{pAp}X_{pAq} & (3'') \\ X_{pAp} \rightarrow aX_{pAq}X_{qAp} & (3''') \quad X_{pAq} \rightarrow aX_{pAq}X_{qAq} & (3''''') \\ X_{pAq} \rightarrow b & (4') \quad X_{qAq} \rightarrow b & (5') \end{array}$$

- Der akzeptierenden Rechnung

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

von M entspricht dann in G die Linksableitung

$$\underline{S} \underset{(0')}{\Rightarrow} \underline{X_{p\#q}} \underset{(2'')}{\Rightarrow} \underline{aX_{pAq}} \underset{(3''''')}{\Rightarrow} \underline{aaX_{pAq}X_{qAq}} \underset{(4')}{\Rightarrow} \underline{aabX_{qAq}} \underset{(5')}{\Rightarrow} \underline{aabb}.$$

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

- Für einen PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ sei G die Grammatik (V, Σ, P, S) mit $V = \{S\} \cup \{X_{pAq} \mid p, q \in Z, A \in \Gamma\}$, wobei P neben den Regeln $S \rightarrow X_{q_0\#p'}$, $p' \in Z$, für jede Anweisung

$$p u A \rightarrow p_0 A_1 \dots A_k, \quad k \geq 0$$

von M und jede Zustandsfolge p_1, \dots, p_k die folgende Regel enthält:

$$X_{pA p_k} \rightarrow u X_{p_0 A_1 p_1} \dots X_{p_{k-1} A_k p_k}.$$

- Dann lässt sich mit Hilfe der Äquivalenz

$$X_{pA p'} \Rightarrow^* x \text{ gdw. } (p, x, A) \vdash^* (p', \varepsilon, \varepsilon),$$

deren Beweis wir später nachholen, leicht die Korrektheit von G zeigen:

$$x \in L(M) \Leftrightarrow (q_0, x, \#) \vdash^* (p', \varepsilon, \varepsilon) \text{ für ein } p' \in Z$$

$$\Leftrightarrow S \Rightarrow X_{q_0\#p'} \Rightarrow^* x \text{ für ein } p' \in Z$$

$$\Leftrightarrow x \in L(G).$$

Beweis von $X_{pAp'} \Rightarrow^* x$ gdw. $(p, x, A) \vdash^* (p', \varepsilon, \varepsilon)$

- Es bleibt zu zeigen, dass für alle $p, p' \in Z$, $A \in \Gamma$ und $x \in \Sigma^*$ folgende Äquivalenz gilt:

$$X_{pAp'} \Rightarrow^* x \text{ gdw. } (p, x, A) \vdash^* (p', \varepsilon, \varepsilon). \quad (*)$$

- Hierzu zeigen wir durch Induktion über m folgende stärkere Behauptung:

$$X_{pAp'} \Rightarrow^m x \text{ gdw. } (p, x, A) \vdash^m (p', \varepsilon, \varepsilon). \quad (**)$$

Beweis von $(**)$ durch Induktion über m :

$m = 0$: Da weder $X_{pAp'} \Rightarrow^0 x$ noch $(p, x, A) \vdash^0 (p', \varepsilon, \varepsilon)$ gelten, ist die Äquivalenz $(**)$ für $m = 0$ erfüllt.

Beweis von $X_{pAp'} \Rightarrow^m x$ gdw. $(p, x, A) \vdash^m (p', \varepsilon, \varepsilon)$

$m \rightsquigarrow m + 1$: Wir zeigen zuerst die Implikation von links nach rechts.

- Gelte also $X_{pAp'} \Rightarrow^{m+1} x$ und sei α die im ersten Schritt abgeleitete Satzform, d.h. $X_{pAp'} \Rightarrow \alpha \Rightarrow^m x$.
- Wegen $X_{pAp'} \rightarrow_G \alpha$ gibt es eine Anweisung $puA \rightarrow p_0 A_1 \dots A_k$, $k \geq 0$, und Zustände $p_1, \dots, p_k \in Z$ mit

$$\alpha = u X_{p_0 A_1 p_1} \dots X_{p_{k-1} A_k p_k}, \text{ wobei } p_k = p' \text{ ist.}$$

- Wegen $\alpha \Rightarrow^m x$ ex. eine Zerlegung $x = uu_1 \dots u_k$ und Zahlen $m_i \geq 1$ mit $m_1 + \dots + m_k = m$ und

$$X_{p_i A_{i+1} p_{i+1}} \Rightarrow^{m_{i+1}} u_{i+1} \quad (i = 0, \dots, k-1).$$

- Nach IV gibt es somit Rechnungen

$$(p_i, u_{i+1}, A_{i+1}) \vdash^{m_{i+1}} (p_{i+1}, \varepsilon, \varepsilon), \quad i = 0, \dots, k-1.$$

Beweis von $X_{pAp'} \Rightarrow^m x$ gdw. $(p, x, A) \vdash^m (p', \varepsilon, \varepsilon)$

Induktionsschritt für die Implikation von links nach rechts

- Nach IV gibt es somit Rechnungen

$$(p_i, u_{i+1}, A_{i+1}) \vdash^{m_{i+1}} (p_{i+1}, \varepsilon, \varepsilon), \quad i = 0, \dots, k-1,$$

aus denen sich die gesuchte Rechnung der Länge $m+1$ zusammensetzen lässt:

$$\begin{aligned} (p, uu_1 \dots u_k, A) \vdash & (p_0, u_1 \dots u_k, A_1 \dots A_k) \\ & \vdash^{m_1} (p_1, u_2 \dots u_k, A_2 \dots A_k) \\ & \vdots \\ & \vdash^{m_{k-1}} (p_{k-1}, u_k, A_k) \\ & \vdash^{m_k} (p_k, \varepsilon, \varepsilon). \end{aligned}$$

Beweis von $X_{pAp'} \Rightarrow^m x$ gdw. $(p, x, A) \vdash^m (p', \varepsilon, \varepsilon)$

Induktionsschritt für die Implikation von rechts nach links

- Sei nun umgekehrt eine Rechnung der Länge $m + 1$ gegeben:

$$(p, x, A) \vdash (p_0, x', A_1 \dots A_k) \vdash^m (p', \varepsilon, \varepsilon).$$

- Sei $puA \rightarrow p_0A_1 \dots A_k$, $k \geq 0$, die im ersten Rechenschritt ausgeführte Anweisung (d.h. $x = ux'$).
- Für $i = 1, \dots, k$ sei p_i der Zustand, in den M mit Kellerinhalt $A_{i+1} \dots A_k$ gelangt (d.h. $p_k = p'$).
- Dann enthält P die Regel $X_{pAp_k} \rightarrow uX_{p_0A_1p_1} \dots X_{p_{k-1}A_kp_k}$.
- Zudem sei u_i für $i = 1, \dots, k$ das Teilwort von x' , das M zwischen den Besuchen von p_{i-1} und p_i liest.
- Dann ex. Zahlen $m_i \geq 1$ mit $m_1 + \dots + m_k = m$ und

$$(p_i, u_{i+1}, A_{i+1}) \vdash^{m_{i+1}} (p_{i+1}, \varepsilon, \varepsilon) \text{ für } i = 0, \dots, k - 1.$$

- Nach IV gibt es daher Ableitungen

$$X_{p_iA_{i+1}p_{i+1}} \Rightarrow^{m_{i+1}} u_{i+1}, \quad i = 0, \dots, k - 1.$$

Beweis von $X_{pAp'} \Rightarrow^m x$ gdw. $(p, x, A) \vdash^m (p', \varepsilon, \varepsilon)$

Induktionsschritt für die Implikation von rechts nach links

- Nach IV gibt es daher Ableitungen

$$X_{p_i A_{i+1} p_{i+1}} \Rightarrow^{m_{i+1}} u_{i+1}, \quad i = 0, \dots, k-1,$$

die wir zu der gesuchten Ableitung zusammensetzen können:

$$\begin{aligned} X_{pAp_k} &\Rightarrow uX_{p_0A_1p_1} \cdots X_{p_{k-2}A_{k-1}p_{k-1}} X_{p_{k-1}A_kp_k} \\ &\Rightarrow^{m_1} uu_1X_{p_1A_2p_2} \cdots X_{p_{k-2}A_{k-1}p_{k-1}} X_{p_{k-1}A_kp_k} \\ &\quad \vdots \\ &\Rightarrow^{m_{k-1}} uu_1 \cdots u_{k-1} X_{p_{k-1}A_kp_k} \\ &\Rightarrow^{m_k} uu_1 \cdots u_k = x. \end{aligned}$$

□