Einführung in die Theoretische Informatik

Johannes Köbler

Institut für Informatik
Humboldt-Universität zu Berlin

WS 2017/18

Kontextfreie Sprachen

Bemerkung

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

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

- Wir können aber eine kontextfreie Grammatik für L angeben:

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

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

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

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

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

Kontextfreie Sprachen sind auch kontextsensitiv

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

Entfernen 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:

$$
\begin{array}{l}
E' := \{A \in V \mid A \to \varepsilon\} \\
\textbf{repeat} \\
\quad E := E' \\
\quad E' := E \cup \{A \in V \mid \exists\, B_1, \ldots, B_k \in E : A \to B_1 \ldots B_k\} \\
\textbf{until}\ \ E = E'
\end{array}
$$

- Nun bilden wir P' wie folgt:

$$
\left\{ A \to v' \;\middle|\; \begin{array}{l} \text{es ex. eine Regel } A \to_G v \text{, so dass } v' \neq \varepsilon \text{ aus } v \text{ durch} \\ \text{Entfernen von beliebig vielen Variablen } A \in E \text{ entsteht} \end{array} \right\}
$$

□

Entfernen von ε-Regeln

Beispiel

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

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

- Berechnung von E:

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

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

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

◁

Die Chomsky-Hierarchie

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

$\mathrm{REG} \subsetneq \mathrm{CFL} \subseteq \mathrm{CSL} \subseteq \mathrm{RE}$.

Beweis

- Es ist nur noch die Inklusion $\mathrm{CFL} \subseteq \mathrm{CSL}$ zu zeigen.
- Nach obigem Satz ex. zu $L \in \mathrm{CFL}$ 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 $L(G) = L \in \mathrm{CSL}$.
- Im Fall $\varepsilon \in L$ erzeugt die kontextsensitive Grammatik

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

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

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 $V_1 \cap V_2 = \varnothing$ und sei S eine neue Variable.
Dann erzeugen die kontextfreien Grammatiken

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

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

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

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

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

die Sternhülle $L(G_1)^*$. □

Abschlusseigenschaften von CFL

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.

Das Pumping-Lemma für kontextfreie Sprachen

Satz (Pumping-Lemma für kontextfreie Sprachen)

Zu jeder kontextfreien Sprache $L \in \mathrm{CFL}$ 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 Pumping-Lemma für kontextfreie Sprachen

Beispiel

- Betrachte die Sprache $L = \{a^n b^n | n \geq 0\}$.
- Dann lässt sich jedes Wort $z = a^n b^n = a^{n-1} ab b^{n-1}$ 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 $i \geq 0$ $\triangleleft$

Anwendung des Pumping-Lemmas

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. $\triangleleft$

Abschlusseigenschaften von CFL

Wie wir gesehen haben, ist die Klasse CFL abgeschlossen unter

- Vereinigung,
- Produkt und
- Sternhülle.

Satz

CFL ist nicht abgeschlossen unter

- Schnitt und
- Komplement.

Abschlusseigenschaften von CFL

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

- 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 $L \in \mathrm{CFL} \not\Rightarrow \overline{L} \in \mathrm{CFL}$

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

$$\overline{A}, \overline{B} \in \mathrm{CFL} \Rightarrow \overline{A} \cup \overline{B} = \overline{A \cap B} \in \mathrm{CFL}$$

  wiederum $A \cap B \in \mathrm{CFL}$ folgen würde.

Chomsky-Normalform

Definition

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

Satz

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

Anwendungen der Chomsky-Normalform

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

Das Wortproblem für CFL

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 $A \to \varepsilon$ auch sämtliche Variablenumbenennungen $A \to B$ loswerden.

Definition

Regeln der Form $A \to B$ heißen Variablenumbenennungen.

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 sukzessive alle Zyklen

  $$A_1 \to A_2 \to \cdots \to A_k \to A_1.$$

  Hierzu entfernen wir diese Regeln aus P und ersetzen alle Vorkommen der Variablen $A_2, \ldots, A_k$ in den übrigen Regeln durch A_1.
  (Sollte sich unter den entfernten Variablen $A_2, \ldots, A_k$ die Startvariable S befinden, so sei A_1 die neue Startvariable.)

Entfernen von Variablenumbenennungen

Beispiel (Fortsetzung)

$$P\colon S \to aY, bX, Z \qquad Y \to b, bS, aYY \qquad T \to U$$
$$X \to a, aS, bXX \qquad Z \to c, S, T, cZ \qquad U \to abc$$

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

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

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

$$A_1 \to A_2 \to \cdots \to A_k \to A_1.$$

- Nun werden wir sukzessive die restlichen Variablenumbenennungen los, indem wir
  - eine Regel $A \to B$ wählen, so dass in P keine Variablenumbenennung $B \to C$ mit B auf der linken Seite existiert,
  - diese Regel $A \to B$ aus P entfernen und
  - für jede Regel $B \to v$ in P die Regel $A \to v$ zu P hinzunehmen. □

Entfernen von Variablenumbenennungen

Beispiel (Fortsetzung)

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

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

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

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

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

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

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

Entfernen von ε-Regeln und Variablenumbenennungen

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 $L \in \mathrm{CFL}$ gibt es eine CNF-Grammatik G' mit $L(G') = L \setminus \{\varepsilon\}$.

Umwandlung in Chomsky-Normalform

Satz

Zu jeder kontextfreien Sprache $L \in \mathrm{CFL}$ 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 $X_a \to a$ 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 $A \to B_1 \dots B_k$, $k \geq 3$, durch die $k-1$ Regeln

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

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

Umwandlung in Chomsky-Normalform

Beispiel (Fortsetzung)

- Betrachte die Regeln

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

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

$$S \to ABC, AY, BX, CS, c \qquad X \to AS, BXX, a$$
$$Y \to BS, AYY, b \qquad A \to a, \; B \to b, \; C \to c$$

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

$$S \to AS', AY, BX, CS, c, \quad S' \to BC \qquad X \to AS, BX', a, \quad X' \to XX$$
$$Y \to BS, AY', b \quad Y' \to YY \qquad A \to a, \; B \to b, \; C \to c$$

◁

Links- und Rechtsableitungen

Definition

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

- Eine Ableitung

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

  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 $i = 1, \ldots, m-1$.

- 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.

Es gilt:

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

$$
\begin{array}{lllllllllll}
\underline{S} & \Rightarrow_L & a\underline{S}bS & \Rightarrow_L & aa\underline{S}bSbS & \Rightarrow_L & aab\underline{S}bS & \Rightarrow_L & aabb\underline{S} & \Rightarrow_L & aabb \\
\underline{S} & \Rightarrow & a\underline{S}bS & \Rightarrow & aa\underline{S}bSbS & \Rightarrow & aabSb\underline{S} & \Rightarrow & aab\underline{S}b & \Rightarrow & aabb \\
\underline{S} & \Rightarrow & a\underline{S}bS & \Rightarrow & aaSb\underline{S}bS & \Rightarrow & aa\underline{S}bbS & \Rightarrow & aabb\underline{S} & \Rightarrow & aabb \\
\underline{S} & \Rightarrow & a\underline{S}bS & \Rightarrow & aaSb\underline{S}bS & \Rightarrow & aaSbb\underline{S} & \Rightarrow & aa\underline{S}bb & \Rightarrow & aabb \\
\underline{S} & \Rightarrow & a\underline{S}bS & \Rightarrow & aaSbSb\underline{S} & \Rightarrow & aa\underline{S}bSb & \Rightarrow & aab\underline{S}b & \Rightarrow & aabb \\
\underline{S} & \Rightarrow & a\underline{S}bS & \Rightarrow & aaSbSb\underline{S} & \Rightarrow & aaSb\underline{S}b & \Rightarrow & aa\underline{S}bb & \Rightarrow & aabb \\
\underline{S} & \Rightarrow & aSb\underline{S} & \Rightarrow & a\underline{S}b & \Rightarrow & aa\underline{S}bSb & \Rightarrow & aab\underline{S}b & \Rightarrow & aabb \\
\underline{S} & \Rightarrow_R & aSb\underline{S} & \Rightarrow_R & a\underline{S}b & \Rightarrow_R & aaSb\underline{S}b & \Rightarrow_R & aa\underline{S}bb & \Rightarrow_R & aabb
\end{array}
$$

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

$$
\underline{S} \Rightarrow ab, \qquad \underline{S} \Rightarrow a\underline{S}bS \Rightarrow ab\underline{S} \Rightarrow ab, \qquad \underline{S} \Rightarrow aSb\underline{S} \Rightarrow a\underline{S}b \Rightarrow ab
$$

- Darunter sind 2 Links- und 2 Rechtsableitungen. $\triangleleft$

Ein- und mehrdeutige Grammatiken

Beispiel

- Die Grammatik $G = (\{S\}, \{a, b\}, \{S \to aSbS, \varepsilon\}, S)$ ist eindeutig.
- Dies liegt daran, dass keine Satzform von G das Teilwort Sa enthält.
- Daher muss auf die aktuelle Satzform $y\underline{S}\beta$ einer Linksableitung

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

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

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

$\triangleleft$

Gerichtete Bäume und Wälder

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

- Ein (gerichteter) v_0-v_k-Weg in G ist eine Folge von Knoten $v_0, \ldots, v_k$ mit $(v_i, v_{i+1}) \in E$ für $i = 0, \ldots, k-1$. 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 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.

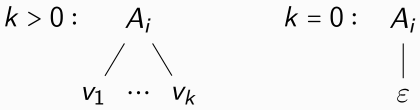
Syntaxbäume

Wir ordnen einer Ableitung

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

den Syntaxbaum (oder Ableitungsbaum, engl. *parse tree*) T_m zu, wobei die Bäume $T_0, \ldots, T_m$ 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 $A_i \to v_1 \ldots v_k$ mit $v_1, \ldots, v_k \in \Sigma \cup V$ angewandt, so ensteht 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 $v_1 \ldots v_k$ von links nach rechts geordnet vor.
- Syntaxbäume sind also geordnete Wurzelbäume.

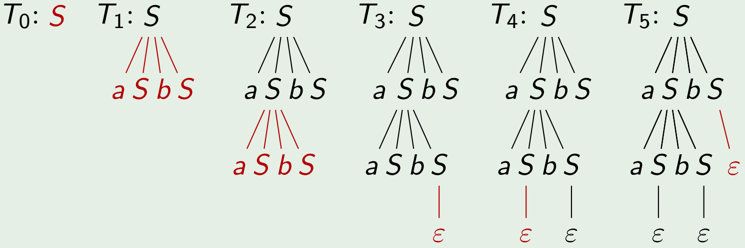
Syntaxbäume

Beispiel

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



Syntaxbäume

Beispiel

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

```
      S
     /|\\
    a S b S
   /|\\    \
  a S b S  ε
    |   |
    ε   ε
```

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

```
    S            S
   / \          /|\\
  a   b        a S b S
                 /     \
                ε       ε
```

Syntaxbäume und Linksableitungen

- Seien $T_0, \ldots, T_m$ die zu einer Ableitung $S = \alpha_0 \Rightarrow \cdots \Rightarrow \alpha_m$ gehörigen Syntaxbäume.
- Dann haben alle Syntaxbäume $T_0, \ldots, T_m$ 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 $\alpha_0 \Rightarrow \cdots \Rightarrow \alpha_m$ 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).

Abschätzung der Blätterzahl bei Binärbäumen

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. $\square$

Mindesttiefe von Binärbäumen

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). $\square$

Beweis des Pumping-Lemmas für CFL

Satz (Pumping-Lemma für kontextfreie Sprachen)

Zu jeder kontextfreien Sprache $L \in \mathrm{CFL}$ 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 $z = z_1 \ldots z_n \in L$ mit $n \geq 1$, so ex. in G eine Ableitung

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

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

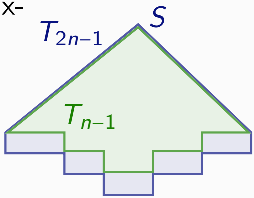
Beweis des Pumping-Lemmas für CFL

Beweis

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

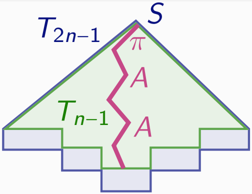
$$S = \alpha_0 \Rightarrow \cdots \Rightarrow \alpha_m = z \text{ mit zugehörigen Syntaxbäumen } T_0, \ldots, T_m$$

- Da G in CNF ist, werden hierbei genau $n - 1$ Regeln der Form $A \to BC$ und genau n Regeln der Form $A \to a$ angewandt.
- Folglich ist $m = 2n - 1$ und wir können annehmen, dass die Regeln der Form $A \to BC$ vor den Regeln der Form $A \to a$ 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.

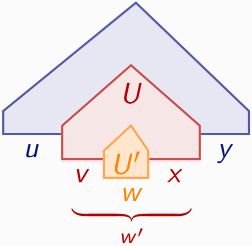


Beweis des Pumping-Lemmas für CFL

Beweis (Fortsetzung)

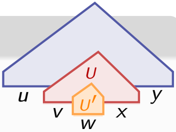


- Setzen wir $l = 2^k$, wobei $k = \|V\|$ ist, so hat T_{n-1} im Fall $n \geq l$ mindestens die Tiefe k, da T_{n-1} mindestens $l = 2^k > 2^{k-1}$ Blätter hat.
- Sei π ein von der Wurzel ausgehender Pfad maximaler Länge in T_{n-1}.
- Dann hat π mindestens die Länge k und unter den letzten $k+1$ Knoten 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.

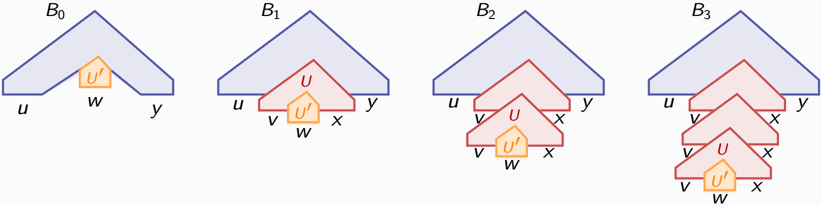


Beweis des Pumping-Lemmas für CFL

Beweis (Schluss)



- Dann ist $vx \neq \varepsilon$ (Bed. 1), da U mehr Blätter hat als U'
- Zudem hat U höchstens $2^k = l$ Blätter, da der Baum $U^* = U \cap T_{n-1}$ höchstens die Tiefe k hat (sonst wäre π nicht maximal)
- Folglich ist $|vwx| \leq l$ (Bed. 2)
- Schließlich lassen sich Syntaxbäume B_i für die Wörter $uv^i wx^i y$, $i \geq 0$, 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 CFL

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?

Der CYK-Algorithmus

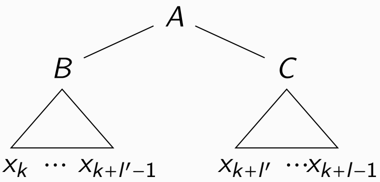
- Sei eine Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $x = x_1 \dots x_n$ gegeben.
- Falls $x = \varepsilon$ ist, können wir effizient prüfen, ob $S \Rightarrow^* \varepsilon$ gilt.
- Hierzu genügt es, die Menge $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$ aller ε-ableitbaren Variablen zu berechnen 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 $A \to x_k$ existiert:

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

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



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

Der CYK-Algorithmus

Algorithmus $\texttt{CYK}(G, x)$

Input: CNF–Grammatik $G = (V, \Sigma, P, S)$ und Wort $x = x_1 \ldots x_n$
  for $k := 1$ **to** n **do**
    $V_{1,k} := \{A \in V \mid A \to x_k \in P\}$
  for $l := 2$ **to** n **do**
    for $k := 1$ **to** $n - l + 1$ **do**
      $V_{l,k} := \varnothing$
      for $l' := 1$ **to** $l - 1$ **do**
        for all $A \to BC \in P$ **do**
          if $B \in V_{l',k}$ **and** $C \in V_{l-l',k+l'}$ **then**
            $V_{l,k} := V_{l,k} \cup \{A\}$
  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.

Der CYK-Algorithmus

Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

$$P\colon\ S \to AS', AY, BX, CS, c,\quad S' \to BC,\quad X \to AS, BX', a,\quad X' \to XX,$$
$$Y \to BS, AY', b,\quad Y' \to YY,\quad A \to a,\quad B \to b,\quad C \to c.$$

- 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

$$P\colon\ S \to AS', AY, BX, CS, c, \quad S' \to BC, \quad X \to AS, BX', a, \quad X' \to XX,$$
$$Y \to BS, AY', b, \quad Y' \to YY, \quad A \to a, \quad B \to b, \quad C \to c.$$

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

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

◁

Ein Maschinenmodell für die kontextfreien Sprachen

Frage

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

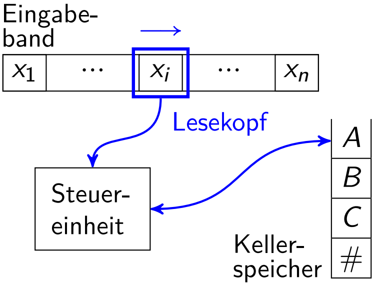
$$L = \{a^n b^n \mid n \geq 0\}$$

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.

Der Kellerautomat



- 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).

Formale Definition des Kellerautomaten

Notation

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

Definition

Ein Kellerautomat wird durch ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ beschrieben, wobei

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

Der Kellerautomat

Arbeitsweise eines Kellerautomaten

- 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

  $$puA \to qB_1 \dots B_k$$

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

Konfiguration eines Kellerautomaten

- 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 $puA_1 \to qB_1 \dots B_k$ 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 $K_0 \vdash K_1 \vdash K_2 \cdots$.

  K_0 heißt Startkonfiguration von M bei Eingabe x.

Die von einem Kellerautomaten erkannte Sprache

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 akzeptiert also genau dann eine Eingabe x, wenn es eine Rechnung gibt, bei der M

- das gesamte Eingabewort liest und
- einen Endzustand $q \in E$ erreicht

Akzeptanz durch Leeren des Kellers

Es gibt noch ein weiteres Akzeptanzkriterium, das die Angabe von Endzuständen überflüssig macht.

Definition

- Ein PDA (engl. *pushdown automaton*) ist ein Kellerautomat $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ ohne Endzustandsmenge.
- Die von einem PDA M akzeptierte oder erkannte Sprache ist

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

- Ein PDA M akzeptiert also genau dann eine Eingabe x, wenn es eine Rechnung gibt, bei der M
  - das gesamte Eingabewort liest und den Keller leert
- Man beachte, dass bei leerem Keller kein Übergang mehr möglich ist
- Es gilt (siehe Übungen)

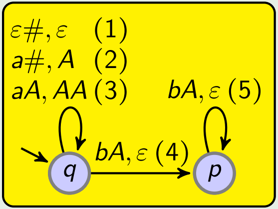
$$\{L(M) \mid M \text{ ist ein PDA}\} = \{L(M) \mid M \text{ ist ein Kellerautomat}\}$$

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

Beispiel

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

$$\delta : q\varepsilon\# \to q \ (1) \quad qa\# \to qA \ (2) \quad qaA \to qAA \ (3)$$
$$qbA \to p \ (4) \quad pbA \to p \quad (5)$$



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

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

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

$n = 0$: $(q, \varepsilon, \#) \underset{(1)}{\vdash} (q, \varepsilon, \varepsilon)$

$n \geq 1$: $(q, a^n b^n, \#) \underset{(2)}{\vdash} (q, a^{n-1} b^n, A) \underset{(3)}{\vdash}^{n-1} (q, b^n, A^n)$

$$\underset{(4)}{\vdash} (p, b^{n-1}, A^{n-1}) \underset{(5)}{\vdash}^{n-1} (p, \varepsilon, \varepsilon)$$

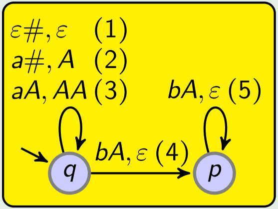
- Dies zeigt, dass M alle Wörter der Form $a^n b^n$, $n \geq 0$, akzeptiert.

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

Beispiel

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

$$\delta: q\varepsilon\# \to q \ (1) \quad qa\# \to qA \ (2) \quad qaA \to qAA \ (3)$$
$$qbA \to p \ (4) \quad pbA \to p \quad (5)$$



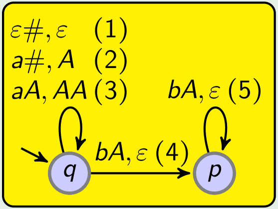
- Als nächstes zeigen wir, dass jede von M akzeptierte Eingabe $x = x_1 \dots x_m \in L(M)$ die Form $x = a^n b^n$ haben muss.
- Ausgehend von der Startkonfiguration $(q, x, \#)$ sind nur die Anweisungen (1) oder (2) ausführbar.
- Führt M zuerst Anweisung (1) aus, so wird der Keller geleert.
- Daher kann M in diesem Fall nur das leere Wort $x = \varepsilon = a^0 b^0$ 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 PDA für die Sprache $\{a^n b^n \mid n \geq 0\}$

Beispiel

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

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



- 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 $n \geq 1$ a's das erste b liest:

$$\begin{array}{rl} (q, x_1 \ldots x_n, \#) \underset{(2)}{\vdash} (q, x_2 \ldots x_n, A) \underset{(3)}{\vdash}^{n-1} & (q, x_{n+1} \ldots x_m, A^n) \\ \underset{(4)}{\vdash} & (p, x_{n+2} \ldots x_m, A^{n-1}) \end{array}$$

  mit $x_1 = x_2 = \cdots = x_n = a$ und $x_{n+1} = b$.
- Damit der Keller nach dem Lesen von x leer ist, muss M nun noch genau $n - 1$ b's lesen, weshalb $x = a^n b^n$ folgt. $\triangleleft$

Ein Maschinenmodell für die Klasse CFL

Ziel

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

Satz

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

Beweis von CFL $\subseteq \{L(M) \mid M$ 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 jedes Zeichen $a \in \Sigma$ die Anweisung $zaa \to z\varepsilon$
- für jede Regel $A \to_G \alpha$ die Anweisung $z\varepsilon A \to z\alpha$

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

Beispiel

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

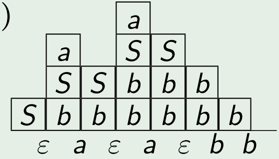
$$P\colon S \to aSb \;\; (1) \quad S \to \varepsilon \;\; (2)$$

- Der zugehörige PDA besitzt dann die Anweisungen

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

- Der Linksableitung $\underline{S} \underset{(1)}{\Rightarrow} a\underline{S}b \underset{(1)}{\Rightarrow} aa\underline{S}bb \underset{(2)}{\Rightarrow} aabb$ 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. $\triangleleft$

Beweis von CFL $\subseteq \{L(M) \mid M$ 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 jedes Zeichen $a \in \Sigma$ die Anweisung $zaa \to z\varepsilon$
- für jede Regel $A \to_G \alpha$ die Anweisung $z\varepsilon A \to z\alpha$

- 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 $S \Rightarrow_L^l x_1 \dots x_n$ gdw. $(z, x_1 \dots x_n, S) \vdash^{l+n} (z, \varepsilon, \varepsilon)$
- Daher folgt

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

$\square$

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

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 $V = \{X_A \mid A \in \Gamma\}$ (im Fall $\Sigma \cap \Gamma = \varnothing$ können wir auch einfach $V = \Gamma$ setzen)
  - die Startvariable von G ist $X_\#$ und
  - P enthält für jede Anweisung $zuA \to zA_1 \dots A_k$ von M die Regel

    $$X_A \to uX_{A_1} \dots X_{A_k}$$
- Dann lässt sich jede akzeptierende Rechnung von $M(x)$ direkt in eine Linksableitung $X_\# \Rightarrow_L x$ in G transformieren und umgekehrt.

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

Beispiel

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

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

  den wir aus der Grammatik $G = (\{S\}, \{a,b\}, P, S)$ mit den beiden Regeln $S \to aSb, \varepsilon$ konstruiert haben.
- Dann führt M auf die Grammatik $G' = (\{X_S, X_a, X_b\}, \{a,b\}, P', X_S)$ mit

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

- 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):

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

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

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

Idee:

Um zu einem PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ eine kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit $L(G) = L(M)$ zu konstruieren, genügt es, M wie folgt in einen äquivalenten PDA $M' = (\{z\}, \Sigma, \Gamma', \delta', z, \#)$ mit nur einem Zustand z zu transformieren:

- Das Kelleralphabet von M' ist $\Gamma' = \{\#\} \cup \{X_{pAq} \mid A \in \Gamma,\ p, q \in Z\}$.
- Zudem fügen wir die folgenden Anweisungen zu δ' hinzu:
  - für jede Anweisung $puA \to p_1 A_1 \dots A_k$ von M und für jede Folge $p_2, \dots, p_{k+1} \in Z$ von k Zuständen die Anweisung

    $$zuX_{pAp_{k+1}} \to zX_{p_1 A_1 p_2} \dots X_{p_k A_k p_{k+1}}$$

  - für jeden Zustand $q \in Z$ die Anweisung

    $$z\varepsilon\# \to zX_{q_0 \# q}$$

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

Beispiel

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

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

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

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

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

Beispiel (Fortsetzung)

- Zudem enthält M' neben den beiden Anweisungen $z\varepsilon\# \to zX_{p\#p}$ (0) und $z\varepsilon\# \to zX_{p\#q}$ (0′) die folgenden Anweisungen:

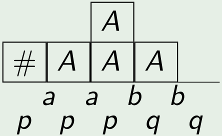
Anweisung von M		k	$p_2, \ldots, p_{k+1}$	Anweisungen von M'	
$p\varepsilon\# \to q$	(1)	0	-	$z\varepsilon X_{p\#q} \to z$	(1′)
$pa\# \to pA$	(2)	1	p	$zaX_{p\#p} \to zX_{pAp}$	(2′)
			q	$zaX_{p\#q} \to zX_{pAq}$	(2″)
$paA \to pAA$	(3)	2	p, p	$zaX_{pAp} \to zX_{pAp}X_{pAp}$	(3′)
			p, q	$zaX_{pAq} \to zX_{pAp}X_{pAq}$	(3″)
			q, p	$zaX_{pAp} \to zX_{pAq}X_{qAp}$	(3‴)
			q, q	$zaX_{pAq} \to pX_{pAq}X_{qAq}$	(3⁗)
$pbA \to q$	(4)	0	-	$zbX_{pAq} \to z$	(4′)
$qbA \to q$	(5)	0	-	$zbX_{qAq} \to z$	(5′)

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

Beispiel (Schluss)

- 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 folgende Rechnung von M':

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

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

◁

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

Zusammenfassung:

Fassen wir die beiden Schritte

- PDA $M \to$ PDA M' mit nur einem Zustand und
- PDA M' mit nur einem Zustand $\to$ kontextfreie Grammatik G

zu einem Schritt zusammen, so können wir zu einem PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ wie folgt eine äquivalente kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit der Variablenmenge $V = \{S\} \cup \{X_{pAq} \mid A \in \Gamma, p, q \in Z\}$ und folgenden Regeln konstruieren:

- für jeden Zustand $q \in Z$ die Startregel

$$S \to X_{q_0 \# q}$$

- für jede Anweisung $puA \to p_1 A_1 \dots A_k$ von M und jede Zustandsfolge $p_2, \dots, p_{k+1}$ die Regel

$$X_{pAp_{k+1}} \to uX_{p_1 A_1 p_2} \dots X_{p_k A_k p_{k+1}}$$

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

Beispiel

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

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

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

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

- Die Regelmenge P enthält die beiden Startregeln

$$S \to X_{p\#p}, X_{p\#q} \quad (0, 0')$$

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

Beispiel (Fortsetzung)

- Zudem enthält P die folgenden Produktionen:

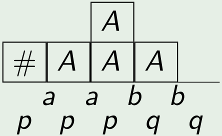
Anweisung		k	$p_2, \ldots, p_{k+1}$	zugehörige Regeln	
$p\varepsilon\# \to q$	(1)	0	-	$X_{p\#q} \to \varepsilon$	$(1')$
$pa\# \to pA$	(2)	1	p	$X_{p\#p} \to aX_{pAp}$	$(2')$
			q	$X_{p\#q} \to aX_{pAq}$	$(2'')$
$paA \to pAA$	(3)	2	p, p	$X_{pAp} \to aX_{pAp}X_{pAp}$	$(3')$
			p, q	$X_{pAq} \to aX_{pAp}X_{pAq}$	$(3'')$
			q, p	$X_{pAp} \to aX_{pAq}X_{qAp}$	$(3''')$
			q, q	$X_{pAq} \to aX_{pAq}X_{qAq}$	$(3'''')$
$pbA \to q$	(4)	0	-	$X_{pAq} \to b$	$(4')$
$qbA \to q$	(5)	0	-	$X_{qAq} \to b$	$(5')$

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

Beispiel (Schluss)

- 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

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

$\triangleleft$

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

- Es bleibt noch zu zeigen, dass die zu einem PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ konstruierte kontextfreie Grammatik $G = (V, \Sigma, P, S)$ mit der Variablenmenge $V = \{S\} \cup \{X_{pAq} \mid A \in \Gamma, p, q \in Z\}$, die
  - für jeden Zustand $q \in Z$ die Startregel $S \to X_{q_0\#q}$ sowie
  - für jede Anweisung $puA \to p_1A_1 \dots A_k$ von M und jede Zustandsfolge $p_2, \dots, p_{k+1}$ die Regel $X_{pAp_{k+1}} \to uX_{p_1A_1p_2} \dots X_{p_kA_kp_{k+1}}$

  enthält, die Sprache $L(M)$ erzeugt.
- Hierzu zeigen wir, dass sich aus einer Rechnung $(p, x, A) \vdash^m (q, \varepsilon, \varepsilon)$ der Länge m von M eine Ableitung $X_{pAq} \Rightarrow^m x$ der Länge m gewinnen lässt und umgekehrt.
- Aus dieser Äquivalenz folgt dann sofort $L(G) = L(M)$:

$$\begin{aligned}
x \in L(M) \quad &\Leftrightarrow \quad (q_0, x, \#) \vdash^m (q, \varepsilon, \varepsilon) \text{ für ein } m \geq 1 \text{ und ein } q \in Z \\
&\Leftrightarrow \quad S \Rightarrow X_{q_0\#q} \Rightarrow^m x \text{ für ein } m \geq 1 \text{ und ein } q \in Z \\
&\Leftrightarrow \quad x \in L(G)
\end{aligned}$$

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

Es bleibt noch zu zeigen, dass für alle $p, q \in Z$, $A \in \Gamma$, $x \in \Sigma^*$ und $m \geq 0$ gilt:

$$X_{pAq} \Rightarrow^m x \text{ gdw. } (p, x, A) \vdash^m (q, \varepsilon, \varepsilon) \qquad (*)$$

Induktionsanfang ($m = 0$):

Da sowohl $X_{pAq} \Rightarrow^0 x$ als auch $(p, x, A) \vdash^0 (q, \varepsilon, \varepsilon)$ falsch sind, gilt die Äquivalenz $(*)$ im Fall $m = 0$.

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

Induktionsschritt ($m \rightsquigarrow m+1$):

- Wir zeigen zuerst die Implikation von links nach rechts.
- Sei x aus X_{pAq} in $m+1$ Schritten ableitbar und sei α die im ersten Schritt abgeleitete Satzform: $X_{pAq} \Rightarrow \alpha \Rightarrow^m x$
- Wegen $X_{pAq} \to_G \alpha$ gibt es eine Anweisung $puA \to p_1 A_1 \dots A_k$ und Zustände $p_2, \dots, p_{k+1} \in Z$ mit $p_{k+1} = q$ und

$$\alpha = u\, X_{p_1 A_1 p_2} \dots X_{p_k A_k p_{k+1}}$$

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

$$X_{p_i A_i p_{i+1}} \Rightarrow^{m_i} u_i \text{ für } i = 1, \dots, k$$

- Nach IV gibt es somit Rechnungen

$$(p_i, u_i, A_i) \vdash^{m_i} (p_{i+1}, \varepsilon, \varepsilon) \text{ für } i = 1, \dots, k$$

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

Induktionsschritt ($m \rightsquigarrow m+1$):

- Nach IV gibt es somit Rechnungen

$$(p_i, u_i, A_i) \vdash^{m_i} (p_{i+1}, \varepsilon, \varepsilon) \text{ für } i = 1, \ldots, k$$

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

$$\begin{array}{lll} (p, x, A) & = & (p, uu_1 \ldots u_k, A) \\ & \vdash & (p_1, u_1 \ldots u_k, A_1 \ldots A_k) \\ & \vdash^{m_1} & (p_2, u_2 \ldots u_k, A_2 \ldots A_k) \\ & \vdots & \\ & \vdash^{m_{k-1}} & (p_k, u_k, A_k) \\ & \vdash^{m_k} & (p_{k+1}, \varepsilon, \varepsilon) \; = \; (q, \varepsilon, \varepsilon) \end{array}$$

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

Induktionsschritt ($m \rightsquigarrow m+1$):

- Nun zeigen wir die Implikation von rechts nach links.
- Sei eine Rechnung der Länge $m+1$ gegeben und sei $puA \to p_1A_1 \dots A_k$ die erste Anweisung:

$$(p, x, A) \vdash (p_1, x', A_1 \dots A_k) \vdash^m (q, \varepsilon, \varepsilon)$$

- Im Fall $k \geq 2$ sei p_i für $i = 2, \dots, k$ der Zustand, in den M mit dem Kellerinhalt $A_i \dots A_k$ gelangt.
- Zudem sei u_i für $i = 1, \dots, k$ das zwischen den Besuchen von p_i und p_{i+1} gelesene Teilwort von x, wobei $p_{k+1} = q$ ist.
- Dann gilt $x = ux'$ und $x' = u_1 \dots u_k$ sowie

$$(p_1, x', A_1 \dots A_k) \vdash^* (p_i, u_i \dots u_k, A_i \dots A_k) \vdash^* (q, \varepsilon, \varepsilon)$$

- Für $i = 1, \dots, k$ ex. daher Zahlen $m_i \geq 1$ mit

$$(p_i, u_i, A_i) \vdash^{m_i} (p_{i+1}, \varepsilon, \varepsilon) \text{ und } m_1 + \dots + m_k = m$$

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

Induktionsschritt ($m \rightsquigarrow m+1$):

- Für $i = 1, \ldots, k$ ex. daher Zahlen $m_i \geq 1$ mit

  $$(p_i, u_i, A_i) \vdash^{m_i} (p_{i+1}, \varepsilon, \varepsilon) \text{ und } m_1 + \cdots + m_k = m$$

- Nach IV ex. daher für $i = 1, \ldots, k$ die Ableitungen

  $$X_{p_i A_i p_{i+1}} \Rightarrow^{m_i} u_i$$

- Aufgrund der Anweisung $puA \to p_1 A_1 \ldots A_k$ enthält P zudem die Regel $X_{pAq} \to u X_{p_1 A_1 p_2} \ldots X_{p_{k-1} A_{k-1} p_k} X_{p_k A_k q}$, die sich wie folgt mit obigen Ableitungen zusammensetzen lässt:

$$\begin{array}{lll}
X_{pAq} & \Rightarrow & u X_{p_1 A_1 p_2} \ldots X_{p_{k-1} A_{k-1} p_k} X_{p_k A_k q} \\
 & \Rightarrow^{m_1} & u u_1 \ldots X_{p_{k-1} A_{k-1} p_k} X_{p_{k-1} A_k q} \\
 & \vdots & \\
 & \Rightarrow^{m_{k-1}} & u u_1 \ldots u_{k-1} X_{p_k A_k q} \\
 & \Rightarrow^{m_k} & u u_1 \ldots u_k = x
\end{array}$$

$\square$

Deterministische Kellerautomaten

In der Praxis spielen det. Kellerautomaten eine wichtige Rolle.

Definition

- Ein Kellerautomat heißt deterministisch, falls $\vdash$ rechtseindeutig ist:

  $$K \vdash K_1 \wedge K \vdash K_2 \Rightarrow K_1 = K_2$$

  Äquivalent hierzu ist, dass die Überführungsfunktion δ für alle $(q, a, A) \in Z \times \Sigma \times \Gamma$ folgende Bedingung erfüllt:

  $$\|\delta(q, a, A)\| + \|\delta(q, \varepsilon, A)\| \leq 1$$

- Ein Kellerautomat $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ ist ein DPDA (engl. *deterministic pushdown automaton*), falls M für alle $(q, a, A) \in Z \times \Sigma \times \Gamma$ zusätzlich folgende Bedingung erfüllt:

  $$\|\delta(q, a, A)\| + \|\delta(q, \varepsilon, A)\| \leq 1$$

- Weiter sei DCFL $= \{L(M) \,|\, M \text{ ist ein DPDA}\}$.