Einführung in die Theoretische Informatik

Johannes Köbler

Institut für Informatik
Humboldt-Universität zu Berlin

WS 2018/19

Bemerkung

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

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

- Also gilt REG $\subsetneq$ CFL
- Allerdings sind nicht alle kontextfreien Grammatiken kontextsensitiv
- Z.B. ist obige Grammatik G nicht kontextsensitiv, da sie die Regel $S \to \varepsilon$ enthält und S auf der rechten Seite der Regel $S \to aSb$ vorkommt
- Wir können G jedoch wie folgt in eine Grammatik G' umwandeln:
  - ersetze die Regel $S \to \varepsilon$ durch die Regel $S \to ab$ und
  - füge ein neues Startsymbol S' sowie die Regeln $S' \to S, \varepsilon$ hinzu
- Tatsächlich lässt sich jede kontextfreie Grammatik G in eine äquivalente kontextfreie Grammatik G' umwandeln, die auch kontextsensitiv ist

Chomsky-Normalform

Definition

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

Satz

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

Anwendungen der Chomsky-Normalform

Korollar

CFL $\subseteq$ CSL.

Beweis

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

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

□

Weitere Anwendungen der Chomsky-Normalform

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

Das Pumping-Lemma für kontextfreie Sprachen

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

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

Das Wortproblem für kontextfreie Grammatiken

Gegeben: Eine kontextfreie Grammatik G und ein Wort x.

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

Das Pumping-Lemma für kontextfreie Sprachen

Beispiel

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

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

dann gilt

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

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

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 gilt
  - $L(G_1) \cup L(G_2) = L(G_3)$ für die kontextfreie Grammatik
    $$G_3 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1, S_2\}, S)$$
  - $L(G_1)L(G_2) = L(G_4)$ für die kontextfreie Grammatik
    $$G_4 = (V_1 \cup V_2 \cup \{S\}, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 S_2\}, S) \text{ und}$$
  - $L(G_1)^* = L(G_5)$ für die kontextfreie Grammatik
    $$G_5 = (V_1 \cup \{S\}, \Sigma, P_1 \cup \{S \to S_1 S, \varepsilon\}, S)$$

□

Abschlusseigenschaften von CFL

Satz

CFL ist nicht abgeschlossen unter Schnitt und Komplement.

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

- Folgende Sprachen sind kontextfrei (siehe Übungen):

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

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

Beweis von $L \in \mathrm{CFL} \not\Rightarrow \overline{L} \in \mathrm{CFL}$

- 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 □

Umwandlung in Chomsky-Normalform

Satz

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

Beweis

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

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

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

□

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

Beseitigung von ε-Regeln

Satz

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

Beweis

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

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

- Nun bilden wir P' wie folgt:

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

$\square$

Beseitigung 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 die Regeln $Y \to b$ (wegen $Y \to bS$), $X \to a$ (wegen $X \to aS$) und $Z \to c$ (wegen $Z \to cZ$) hinzu:

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

Beseitigung von Variablenumbenennungen

Satz

Zu jeder kontextfreien Grammatik $G = (V, \Sigma, P, S)$ gibt es eine kontextfreie Grammatik $G' = (V, \Sigma, P', S)$ ohne Regeln der Form $A \to B$ 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
- Befindet sich die Startvariable unter $A_1, \ldots, A_k$, so sei dies o.B.d.A. A_1
- Nun eliminieren wir sukzessive die restlichen Variablenumbenennungen, 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 $\square$

Beseitigung von Variablenumbenennungen

Beispiel (Fortsetzung)

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

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

$$\begin{array}{lll} S \to aY, bX, c, T, cS & Y \to b, bS, aYY & T \to U \\ X \to a, aS, bXX & U \to abc & \end{array}$$

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

$$\begin{array}{lll} S \to aY, bX, c, T, cS & Y \to b, bS, aYY & T \to abc \\ X \to a, aS, bXX & U \to abc & \end{array}$$

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

$$\begin{array}{lll} S \to abc, aY, bX, c, cS & Y \to b, bS, aYY & T \to abc \\ X \to a, aS, bXX & U \to abc & \end{array}$$

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

$$\begin{array}{lll} S \to abc, aY, bX, c, cS & Y \to b, bS, aYY & X \to a, aS, bXX \end{array}$$

Bringe alle Regeln in die Form $A \to a$ und $A \to BC$

Beispiel (Schluss)

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

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

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

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

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

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

◁

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

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 $aabb$:

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

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

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

- Darunter sind zwei Links- und zwei Rechtsableitungen ◁

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:

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

◁

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 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:

$$k > 0: \quad \begin{array}{c} A_i \\ \diagup \quad \diagdown \\ v_1 \;\; \cdots \;\; v_k \end{array} \qquad\qquad k = 0: \quad \begin{array}{c} A_i \\ | \\ \varepsilon \end{array}$$

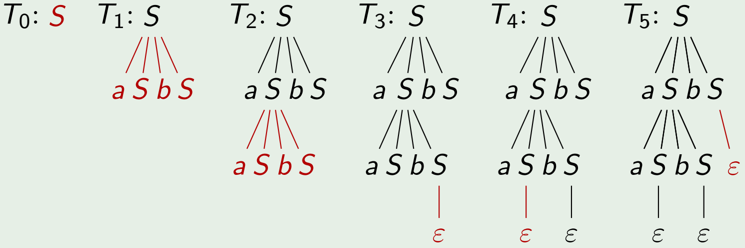
- Hierbei stellen wir uns die Kanten von oben nach unten gerichtet und die Kinder $v_1 \ldots v_k$ von links nach rechts geordnet vor.
- Syntaxbäume sind also geordnete Wurzelbäume.

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. $uv^i wx^i y \in L$ 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

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

- 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

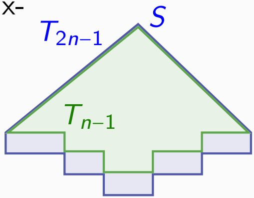
Beweis des Pumping-Lemmas für CFL

Beweis

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

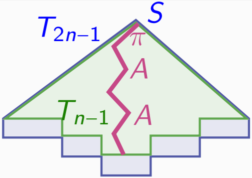
$$S = \alpha_0 \Rightarrow \cdots \Rightarrow \alpha_m = z \text{ mit zugehörigen Syntaxbäumen } T_0, \dots, T_m$$

- 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

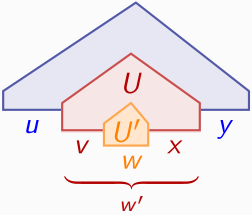


Beweis des Pumping-Lemmas für CFL

Beweis (Fortsetzung)

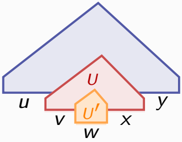


- Setzen wir $l = 2^k$, wobei $k = \|V\|$ ist, so hat T_{n-1} im Fall $n \geq l$ mindestens die Tiefe k, da T_{n-1} mindestens $l = 2^k > 2^{k-1}$ Blätter hat
- Sei π ein von der Wurzel ausgehender Pfad maximaler Länge in T_{n-1}
- Dann hat π mindestens die Länge k und unter den letzten $k+1$ Knoten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

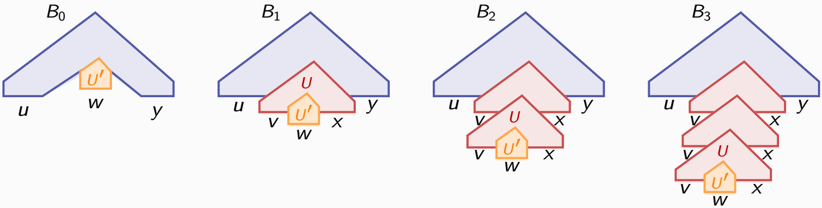


Beweis des Pumping-Lemmas für CFL

Beweis (Schluss)



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



□

Das Wortproblem für 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

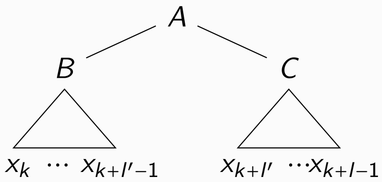
- Sei eine Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $x = x_1 \dots x_n$ gegeben.
- Falls $x = \varepsilon$ ist, können wir effizient prüfen, ob $S \Rightarrow^* \varepsilon$ gilt.
- Hierzu genügt es, die Menge $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$ aller ε-ableitbaren Variablen zu berechnen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

Input: CNF–Grammatik $G = (V, \Sigma, P, S)$ und Wort $x = x_1 \dots x_n$
  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.

Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

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

- 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 $x \notin L(G)$.

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

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

- 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

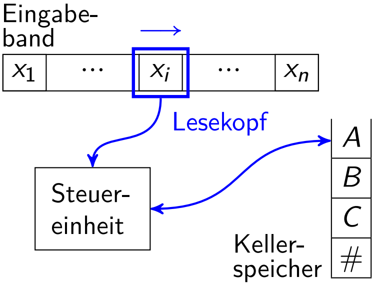
$$L = \{a^n b^n \mid n \geq 0\}$$

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 in jedem Rechenschritt das oberste Kellersymbol löschen und durch beliebig viele Symbole ersetzen

Kellerautomaten mit Endzuständen

Notation

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

Definition

Ein Kellerautomat mit Endzuständen (auch FS-PDA für engl. *final state pushdown automaton*) ist ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$. Dabei ist

- $Z \neq \varnothing$ 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
- $E \subseteq Z$ die Menge der Endzustände

Arbeitsweise eines Kellerautomaten

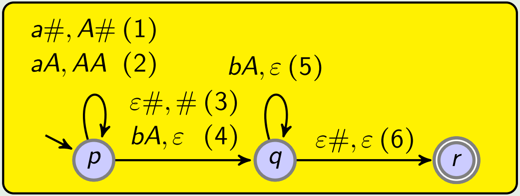
- Wenn p der momentane Zustand, A das oberste Kellerzeichen und $u \in \Sigma$ das nächste Eingabezeichen (bzw. $u = \varepsilon$) ist, so kann M im Fall $(q, B_1 \dots B_k) \in \delta(p, u, A)$
  - in den Zustand q wechseln,
  - den Lesekopf auf dem Eingabeband um $|u| \in \{0, 1\}$ Positionen vorrücken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

Ein Kellerautomat mit Endzuständen

Beispiel

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

$$\begin{aligned}
\delta : pa\# &\to pA\# && (1)\\
paA &\to pAA && (2)\\
p\varepsilon\# &\to q\# && (3)\\
pbA &\to q && (4)\\
qbA &\to q && (5)\\
q\varepsilon\# &\to r && (6)
\end{aligned}$$



◁

Konfiguration eines Kellerautomaten

- Eine Konfiguration von M wird durch ein Tripel

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

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

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

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

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

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

Akzeptanz durch Erreichen eines Endzustands

Notation

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

Definition

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

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

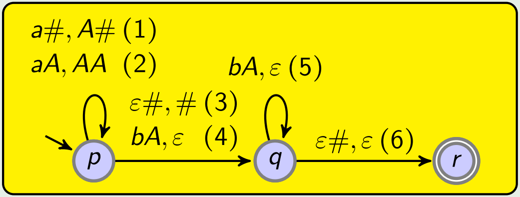
- Ein Kellerautomat M mit Endzuständen akzeptiert also genau dann ein Wort x, wenn es bei dieser Eingabe eine Rechnung gibt, bei der M
  - alle Eingabezeichen liest und
  - einen Endzustand $q \in E$ erreicht

Ein Kellerautomat mit Endzuständen

Beispiel (Fortsetzung)

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

$$\begin{aligned}
\delta : pa\# &\to pA\# && (1)\\
paA &\to pAA && (2)\\
p\varepsilon\# &\to q\# && (3)\\
pbA &\to q && (4)\\
qbA &\to q && (5)\\
q\varepsilon\# &\to r && (6)
\end{aligned}$$



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

$$(p, aabb, \#) \underset{(1)}{\vdash} (p, abb, A\#) \underset{(2)}{\vdash} (p, bb, AA\#) \underset{(4)}{\vdash} (q, b, A\#) \underset{(5)}{\vdash} (q, \varepsilon, \#) \underset{(6)}{\vdash} (r, \varepsilon, \varepsilon)$$

◁

Akzeptanz durch Leeren des Kellers

Es gibt auch ein Akzeptanzkriterium, das keine Endzustände erfordert.

Definition

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

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

- Wir nennen M auch einen ES-PDA (für engl. *empty stack pushdown automaton*) oder einfach PDA.

- Ein PDA M akzeptiert also genau dann eine Eingabe, wenn es eine Rechnung gibt, bei der M alle Eingabezeichen liest und den Keller leert
- Es gilt (siehe Übungen)

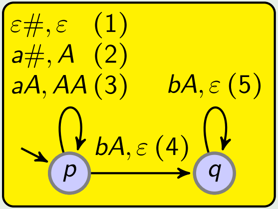
$$\{L(M) \mid M \text{ ist ein FS-PDA}\} = \{L(M) \mid M \text{ ist ein ES-PDA}\}$$

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

Beispiel

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

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



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

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

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

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

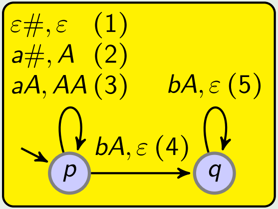
$n \geq 1$: $\quad (p, a^n b^n, \#) \underset{(2)}{\vdash} (p, a^{n-1} b^n, A) \underset{(3)}{\vdash}^{n-1} (p, b^n, A^n)$

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

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

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

Beispiel (Fortsetzung)



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

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

- Es gilt auch die umgekehrte Inklusion:

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

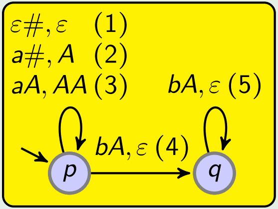
- Ausgehend von der Startkonfiguration $(q, x, \#)$ sind nämlich nur die Anweisungen (1) oder (2) ausführbar.
- Führt M zuerst Anweisung (1) aus, so wird der Keller geleert.
- Daher kann M in diesem Fall nur das leere Wort $x = \varepsilon = a^0 b^0$ akzeptieren.
- Falls M mit Anweisung (2) beginnt, 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 (Schluss)

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

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



- 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:

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

mit $x_1 = x_2 = \dots = x_n = a$ und $x_{n+1} = b$.
- Damit der Keller nach dem Lesen von x_m leer ist, muss M nun noch genau $n - 1$ b's lesen, weshalb $m = 2n$ und $x = a^n b^n$ folgt. $\triangleleft$

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

Idee:

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

- für jede Regel $A \to_G \alpha$ die Anweisung $z\varepsilon A \to z\alpha$
- für jedes Zeichen $a \in \Sigma$ die Anweisung $zaa \to z\varepsilon$

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

Beispiel

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

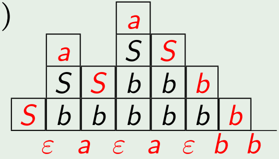
$$P\colon S \to aSb \;\; (1) \quad S \to \varepsilon \;\; (2)$$

- 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 ◁

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 jede Regel $A \to_G \alpha$ die Anweisung $z\varepsilon A \to z\alpha$
- für jedes Zeichen $a \in \Sigma$ die Anweisung $zaa \to z\varepsilon$

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

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

$\square$

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

Vorbetrachtung:

- Obige Konstruktion eines PDA M aus einer kontextfreien Grammatik lässt sich leicht umdrehen, falls M nur einen Zustand hat
- Zu einem solchen PDA $M = (\{z\}, \Sigma, \Gamma, \delta, z, \#)$ lässt sich wie folgt eine kontextfreie Grammatik $G = (V, \Sigma, P, X_{\#})$ mit $L(G) = L(M)$ konstruieren:
  - Die Variablenmenge von G ist $V = \{X_A \mid A \in \Gamma\}$ (wir können auch o.B.d.A. $\Sigma \cap \Gamma = \varnothing$ annehmen und $V = \Gamma$ 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 $(z, x, \#) \vdash^m (z, \varepsilon, \varepsilon)$ von $M(x)$ der Länge m direkt in eine Linksableitung $X_{\#} \Rightarrow_L^m x$ in G der Länge m transformieren und umgekehrt

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

Beispiel

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

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

  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

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

- 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):

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

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

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

Idee:

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

- Das Kelleralphabet ist $\Gamma' = \{S\} \cup \{X_{pAq} \mid A \in \Gamma,\ p, q \in Z\}$ und
- die Überführungsfunktion δ' enthält folgende Anweisungen:
  - für jeden Zustand $q \in Z$ die Anweisung

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

  - für jede Anweisung $puA \to p_1A_1 \dots A_k$ von M und für jede Folge von k Zuständen $p_2, \dots, p_{k+1} \in Z$ die Anweisung

    $$zuX_{pAp_{k+1}} \to zX_{p_1A_1p_2} \dots X_{p_kA_kp_{k+1}}$$

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

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

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

Beispiel (Fortsetzung)

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

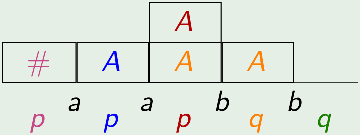
Anweisung von M		k	$p_2, \ldots, p_{k+1}$	Anweisungen von M'	
$p\varepsilon\# \to q$	(1)	0	-	$z\varepsilon X_{p\#q} \to z$	$(1')$
$pa\# \to pA$	(2)	1	p	$\cancel{zaX_{p\#p} \to zX_{pAp}}$	$\cancel{(2')}$
			q	$zaX_{p\#q} \to zX_{pAq}$	$(2'')$
$paA \to pAA$	(3)	2	p, p	$zaX_{pAp} \to zX_{pAp}X_{pAp}$	$(3')$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

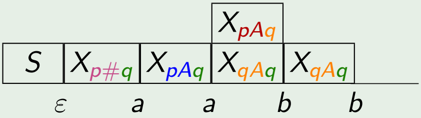
- Der (akzeptierenden) Rechnung

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



von M entspricht dann folgende Rechnung von M':

$$(z, aabb, S) \underset{(0')}{\vdash} (z, aabb, X_{p\#q}) \underset{(2'')}{\vdash} (z, abb, X_{pAq})$$
$$\underset{(3'''')}{\vdash} (z, bb, X_{pAq}X_{qAq}) \underset{(4')}{\vdash} (z, b, X_{qAq}) \underset{(5')}{\vdash} (z, \varepsilon, \varepsilon)$$



◁

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

- Es bleibt noch zu zeigen, dass der zu einem PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ konstruierte PDA $M' = (\{z\}, \Sigma, \Gamma', \delta', z, S)$ mit dem Kelleralphabet $\Gamma' = \{S\} \cup \{X_{pAq} \mid p, q \in Z, A \in \Gamma\}$, der
  - für jeden Zustand $q \in Z$ die Anweisung $z\varepsilon S \to zX_{q_0\#q}$ sowie
  - für jede Anweisung $puA \to p_1 A_1 \dots A_k$ von M und jede Zustandsfolge $p_2, \dots, p_{k+1}$ die Anweisung $zuX_{pAp_{k+1}} \to zX_{p_1 A_1 p_2} \dots X_{p_k A_k p_{k+1}}$

  enthält, die gleiche Sprache wie M akzeptiert.
- Hierzu zeigen wir, dass jede Rechnung $(p, x, A) \vdash^m (q, \varepsilon, \varepsilon)$ von M einer Rechnung $(z, x, X_{pAq}) \vdash^m (z, \varepsilon, \varepsilon)$ von M' entspricht und umgekehrt.
- Aus dieser Äquivalenz folgt dann sofort $L(M) = L(M')$:

$$\begin{aligned} x \in L(M) \quad &\Leftrightarrow \quad M \text{ hat für ein } q \in Z \text{ eine akzeptierende Rechnung} \\ &\qquad (q_0, x, \#) \vdash^m (q, \varepsilon, \varepsilon) \text{ der Länge } m \geq 1 \\ &\Leftrightarrow \quad M' \text{ hat für ein } q \in Z \text{ eine akzeptierende Rechnung} \\ &\qquad (z, x, S) \vdash (z, x, X_{q_0\#q}) \vdash^m (z, \varepsilon, \varepsilon) \text{ mit } m \geq 1 \\ &\Leftrightarrow \quad x \in L(M') \end{aligned}$$

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

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

$$(p, x, A) \vdash_M^m (q, \varepsilon, \varepsilon) \text{ gdw. } (z, x, X_{pAq}) \vdash_{M'}^m (z, \varepsilon, \varepsilon) \qquad (*)$$

Induktionsanfang ($m = 0$):

Da weder M noch M' in $m = 0$ Rechenschritten den Keller leeren können, gilt die Äquivalenz $(*)$ für $m = 0$.

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

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

- Sei eine Rechnung $(p, x, A) \vdash^{m+1} (q, \varepsilon, \varepsilon)$ der Länge $m+1$ von M gegeben und sei $puA \to p_1 A_1 \ldots A_k$ die im ersten Rechenschritt ausgeführte Anweisung:

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

- Im Fall $k \geq 2$ sei p_i für $i = 2, \ldots, k$ der Zustand, in den M mit dem Kellerinhalt $A_i \ldots A_k$ gelangt
- Zudem sei u_i für $i = 1, \ldots, k$ 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 $x' = u_1 \ldots u_k$ sowie

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

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

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

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

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

- Für $i = 1, \ldots, k$ ex. daher Zahlen $m_i \geq 1$ mit
$$(p_i, u_i, A_i) \vdash^{m_i} (p_{i+1}, \varepsilon, \varepsilon) \text{ und } m_1 + \cdots + m_k = m$$
- Daher hat M' nach IV die Rechnungen $(z, u_i, X_{p_i A_i p_{i+1}}) \vdash^{m_i} (z, \varepsilon, \varepsilon)$
- Zudem hat M' wegen $puA \rightarrow_M p_1 A_1 \ldots A_k$ die Anweisung $zuX_{pAq} \rightarrow zX_{p_1 A_1 p_2} \ldots X_{p_{k-1} A_{k-1} p_k} X_{p_k A_k q}$, so dass wir die gesuchte Rechnung der Länge $m+1$ von M' wie folgt erhalten:
$$\begin{array}{rcl} (z, x, X_{pAq}) & = & (z, uu_1 \ldots u_k, X_{pAq}) \\ & \vdash & (z, u_1 \ldots u_k, X_{p_1 A_1 p_2} \ldots X_{p_{k-1} A_{k-1} p_k} X_{p_k A_k q}) \\ & \vdash^{m_1} & (z, u_2 \ldots u_k, X_{p_2 A_2 p_3} \ldots X_{p_{k-1} A_{k-1} p_k} X_{p_k A_k q}) \\ & \vdots & \\ & \vdash^{m_{k-1}} & (z, u_k, X_{p_k A_k q}) \\ & \vdash^{m_k} & (z, \varepsilon, \varepsilon) \end{array}$$
- Entsprechend lässt sich umgekehrt aus jeder solchen Rechnung von M' eine Rechnung $(p, x, A) \vdash^{m+1} (q, \varepsilon, \varepsilon)$ von M gewinnen. $\square$

Direkte Konstruktion einer Grammatik aus einem PDA

- Wir können 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 zusammenfassen
- Dazu konstruieren wir wie folgt zu einem PDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#)$ eine äquivalente kontextfreie Grammatik $G = (V, \Sigma, P, S)$
- Die Variablenmenge von G ist $V = \{S\} \cup \{X_{pAq} \mid A \in \Gamma, p, q \in Z\}$
- Zudem fügen wir für jeden Zustand $q \in Z$ die Startregel

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

  und 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

  $$X_{pAp_{k+1}} \to uX_{p_1A_1p_2} \dots X_{p_kA_kp_{k+1}}$$

  zu P hinzu

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

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

- 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

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

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

Beispiel (Fortsetzung)

- Zudem enthält P die folgenden Produktionen:

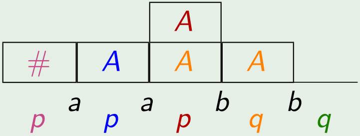
Anweisung		k	$p_2, \ldots, p_{k+1}$	zugehörige Regeln	
$p\varepsilon\# \to q$	(1)	0	-	$X_{p\#q} \to \varepsilon$	(1′)
$pa\# \to pA$	(2)	1	p	$\cancel{X_{p\#p} \to aX_{pAp}}$	~~(2′)~~
			q	$X_{p\#q} \to aX_{pAq}$	(2″)
$paA \to pAA$	(3)	2	p, p	$X_{pAp} \to aX_{pAp}X_{pAp}$	(3′)
			p, q	$X_{pAq} \to aX_{pAp}X_{pAq}$	(3″)
			q, p	$\cancel{X_{pAp} \to aX_{pAq}X_{qAp}}$	~~(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 $\{L(M) \mid M \text{ ist ein PDA}\} \subseteq \mathrm{CFL}$

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 in G die Linksableitung

$$\underline{S} \underset{(0')}{\Rightarrow} \underline{X_{p\#q}} \underset{(2'')}{\Rightarrow} a\underline{X_{pAq}} \underset{(3'''')}{\Rightarrow} aa\underline{X_{pAq}}X_{qAq} \underset{(4')}{\Rightarrow} aab\underline{X_{qAq}} \underset{(5')}{\Rightarrow} aabb$$

$\triangleleft$

Deterministische Kellerautomaten

In der Praxis spielen det. Kellerautomaten eine wichtige Rolle.

Definition

Ein Kellerautomat M heißt deterministisch, falls die Relation $\vdash_M$ rechtseindeutig ist:

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

- Die Anzahl $N(K)$ der Folgekonfigurationen einer Konfiguration $K = (q, x_i \dots x_n, A_1 \dots A_k)$, $1 \le i \le n+1$, ist

$$N(K) = \begin{cases} 0, & k = 0 \\ \|\delta(q, \varepsilon, A_1)\|, & i = n+1 \text{ und } k \ge 1 \\ \|\delta(q, x_i, A_1)\| + \|\delta(q, \varepsilon, A_1)\|, & i \le n \text{ und } k \ge 1 \end{cases}$$

- Daher ist ein Kellerautomat $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ genau dann deterministisch, wenn die Überführungsfunktion δ für alle $(q, a, A) \in Z \times \Sigma \times \Gamma$ folgende Bedingung erfüllt:

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

Deterministische Kellerautomaten

Beispiel

- Betrachte den PDA $M = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{A, B, \#\}, \delta, q_0, \#)$ mit

$$\delta\colon \begin{array}{llll} q_0 a\# \to q_0 A\# & q_0 b\# \to q_0 B\# & q_0 aA \to q_0 AA & q_0 bA \to q_0 BA \\ q_0 aB \to q_0 AB & q_0 bB \to q_0 BB & q_0 cA \to q_1 A & q_0 cB \to q_1 B \\ q_1 aA \to q_1 & q_1 bB \to q_1 & q_1 \varepsilon\# \to q_2 & \end{array}$$

Darstellung von δ in Tabellenform

δ	$q_0, \#$	q_0, A	q_0, B	$q_1, \#$	q_1, A	q_1, B	$q_2, \#$	q_2, A	q_2, B
ε				q_2					
a	$q_0 A\#$	$q_0 AA$	$q_0 AB$		q_1				
b	$q_0 B\#$	$q_0 BA$	$q_0 BB$			q_1			
c		$q_1 A$	$q_1 B$						

- Man beachte, dass jedes Tabellenfeld höchstens eine Anweisung enthält und jede Spalte mit einem ε-Eintrag keine weiteren Einträge enthält.
- Daher ist die Bedingung $\|\delta(q, a, A)\| + \|\delta(q, \varepsilon, A)\| \le 1$ für alle $q \in Z$, $a \in \Sigma$ und $A \in \Gamma$ erfüllt. $\triangleleft$

Wie mächtig sind deterministische ES-PDAs?

Frage

- Können deterministische ES-PDAs (also empty stack PDAs) zumindest alle regulären Sprachen (durch Leeren des Kellers) akzeptieren?
- Kann z.B. die Sprache $L = \{a, aa\}$ von einem deterministischen ES-PDA M akzeptiert werden?

Antwort: Nein

- Um $x = a$ zu akzeptieren, muss M den Keller nach Lesen von a leeren und kann somit keine anderen Wörter mit dem Präfix a akzeptieren
- Deterministische ES-PDAs können also nur präfixfreie Sprachen L akzeptieren (d.h. kein Wort $x \in L$ darf Präfix eines anderen Wortes $y \in L$ sein)

Deterministisch kontextfreie Sprachen

Definition

- Die Klasse der deterministisch kontextfreien Sprachen ist

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

  (für *engl. Deterministic Context Free Languages*).
- Ein deterministischer FS-PDA wird auch als FS-DPDA (für *engl. Finite State Deterministic Push Down Automaton*) oder einfach als DPDA bezeichnet.

Abschlusseigenschaften von DCFL

Frage

Ist DCFL unter Komplementbildung abgeschlossen?

Antwort

Ja. Allerdings ergeben sich beim Versuch, einfach die End- und Nichtendzustände eines DPDA M zu vertauschen, um einen DPDA $\overline{M}$ für $\overline{L(M)}$ zu erhalten, folgende Schwierigkeiten:

1. Falls M eine Eingabe x nicht zu Ende liest, wird x weder von M noch von $\overline{M}$ akzeptiert.
2. Falls M nach dem Lesen von x noch ε-Übergänge ausführt und dabei End- und Nichtendzustände besucht, wird x von M und von $\overline{M}$ akzeptiert.

DPDAs, die ihre Eingabe zu Ende lesen

Satz

Jede Sprache $L \in \mathsf{DCFL}$ wird von einem DPDA M' erkannt, der alle Eingaben zu Ende liest (und bei allen Eingaben hält).

Beweisskizze

Falls ein DPDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ eine Eingabe $x = x_1 \ldots x_n$ nicht zu Ende liest, muss einer der folgenden drei Gründe vorliegen:

1. M gerät in eine Konfiguration $(q, x_i \ldots x_n, \varepsilon)$, $i \leq n$, mit leerem Keller.
2. M gerät in eine Konfiguration $(q, x_i \ldots x_n, A\gamma)$, $i \leq n$, in der wegen $\delta(q, x_i, A) = \delta(q, \varepsilon, A) = \varnothing$ keine Anweisung ausführbar ist.
3. M gerät in eine Konfiguration $(q, x_i \ldots x_n, A\gamma)$, $i \leq n$, so dass M ausgehend von (q, ε, A) eine unendliche Folge von ε-Anweisungen ausführt.

Dies lässt sich vermeiden, indem zu Beginn der Rechnung ein neues Kellerzeichen auf dem Kellerboden platziert wird und ein neuer Fehlerzustand hinzugefügt wird, in dem der Rest der Eingabe gelesen wird. □

Komplementabschluss von DCFL

Satz

Die Klasse DCFL ist unter Komplement abgeschlossen.

Beweisskizze

- Sei $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ ein DPDA, der alle Eingaben zu Ende liest, und sei $L(M) = L$
- Wir konstruieren einen DPDA $\overline{M}$ für $\overline{L}$, der M simuliert
- $\overline{M}$ hat $Z \times \{1, 2, 3\}$ als Zustands- und $Z \times \{3\}$ als Endzustandsmenge
- $\overline{M}$ merkt sich in seinem Zustand (q, i) neben dem aktuellen Zustand q von M, ob M nach Lesen des letzten Zeichens (bzw. seit Beginn der Rechnung) einen Endzustand besucht hat ($i = 1$) oder nicht ($i = 2$)
- Möchte M das nächste Zeichen lesen und befindet sich $\overline{M}$ im Zustand $(q, 2)$, so macht $\overline{M}$ noch einen Umweg über den Endzustand $(q, 3)$, bevor die Simulation von M fortgesetzt wird □

Man beachte, dass $\overline{M}$ in einem Endzustand keine ε-Übergänge macht.

Komplementabschluss von DCFL

Definition

Für eine Sprachklasse $\mathcal{C}$ bezeichne $\mathrm{co}\text{-}\mathcal{C}$ die Klasse $\{\bar{L} \mid L \in \mathcal{C}\}$ aller Komplemente von Sprachen in $\mathcal{C}$.

Korollar

- REG $= \mathrm{co}$-REG,
- DCFL $= \mathrm{co}$-DCFL,
- CFL $\neq \mathrm{co}$-CFL.

Weitere Abschlusseigenschaften von DCFL

Satz

Die Klasse DCFL ist nicht abgeschlossen unter Schnitt, Vereinigung, Produkt und Sternhülle.

$A, B \in \mathrm{DCFL} \not\Rightarrow A \cap B \in \mathrm{DCFL}$

- 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 sogar deterministisch kontextfrei (siehe Übungen).
- Da $L_1 \cap L_2 = \{a^n b^n c^n \mid n \geq 0\}$ nicht kontextfrei ist, liegt der Schnitt dieser Sprachen natürlich auch nicht in DCFL.

$A, B \in \mathrm{DCFL} \not\Rightarrow A \cup B \in \mathrm{DCFL}$

- Da DCFL unter Komplementbildung abgeschlossen ist, kann DCFL wegen de Morgan dann auch nicht unter Vereinigung abgeschlossen sein.
- Beispielsweise sind die Sprachen

  $$L_3 = \{a^i b^j c^k \mid i \neq j \wedge i, j, k \geq 1\} \text{ und } L_4 = \{a^i b^j c^k \mid j \neq k \wedge i, j, k \geq 1\}.$$

  deterministisch kontextfrei (siehe Übungen).
- Ihre Vereinigung gehört aber nicht zu DCFL, d.h.

  $$L_3 \cup L_4 = \{a^i b^j c^k \mid (i \neq j \vee j \neq k) \wedge i, j, k \geq 1\} \in \mathrm{CFL} \setminus \mathrm{DCFL}.$$

- DCFL ist nämlich unter Schnitt mit regulären Sprachen abgeschlossen (siehe Übungen).
- Daher wäre mit $L_3 \cup L_4$ auch die Sprache

  $$(\overline{L_3 \cup L_4}) \cap L(a^+ b^+ c^+) = \{a^n b^n c^n \mid n \geq 1\}$$

  (deterministisch) kontextfrei.

$A, B \in \mathrm{DCFL} \not\Rightarrow AB \in \mathrm{DCFL}$

- Betrachte die DCFL Sprachen

$$L_3 = \{a^i b^j c^k \mid i \neq j \wedge i, j, k \geq 1\} \text{ und } L_4 = \{a^i b^j c^k \mid j \neq k \wedge i, j, k \geq 1\}$$

- Wir wissen bereits, dass $L = L_3 \cup L_4 \notin \mathrm{DCFL}$ ist
- Dann ist aber auch die Sprache

$$0L = 0L_3 \cup 0L_4 \notin \mathrm{DCFL},$$

da sich ein DPDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ für $0L$ leicht zu einem DPDA M' für L umbauen ließe:
  - Sei (p, ε, γ) die Konfiguration, die M nach Lesen der Eingabe 0 erreicht
  - Dann erkennt der DPDA $M' = (Z \cup \{s\}, \Sigma, \Gamma, \delta', s, \#, E)$ die Sprache L, wobei δ' wie folgt definiert ist:

$$\delta'(q, u, A) = \begin{cases} (p, \gamma), & (q, u, A) = (s, \varepsilon, \#) \\ \delta(q, u, A), & (q, u, A) \in Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \end{cases}$$

$A, B \in \mathrm{DCFL} \not\Rightarrow AB \in \mathrm{DCFL}$

- Betrachte die DCFL Sprachen

  $$L_3 = \{a^i b^j c^k \mid i \neq j \wedge i, j, k \geq 1\} \text{ und } L_4 = \{a^i b^j c^k \mid j \neq k \wedge i, j, k \geq 1\}$$

- In den Übungen wird gezeigt, dass auch die beiden Sprachen $\{\varepsilon, 0\}$ und $L_5 = L_3 \cup 0L_4$ in DCFL sind
- Ihr Produkt $\{\varepsilon, 0\}L_5 = L_5 \cup 0L_5 = L_3 \cup 0L_4 \cup 0L_3 \cup 00L_4$ gehört aber nicht zu DCFL
- Da DCFL unter Schnitt mit regulären Sprachen abgeschlossen ist, wäre andernfalls auch die Sprache

  $$\{\varepsilon, 0\}L_5 \cap 0\{a, b, c\}^* = 0L_3 \cup 0L_4$$

  in DCFL, was wir bereits ausgeschlossen haben

Bemerkung

Dass DCFL auch nicht unter Sternhüllenbildung abgeschlossen ist, lässt sich ganz ähnlich zeigen (siehe Übungen)

Abschlusseigenschaften

Abschlusseigenschaften der Klassen REG, DCFL und CFL

	Vereinigung	Schnitt	Komplement	Produkt	Sternhülle
REG	*ja*	*ja*	*ja*	*ja*	*ja*
DCFL	*nein*	*nein*	*ja*	*nein*	*nein*
CFL	*ja*	*nein*	*nein*	*ja*	*ja*