Theoretische Informatik 2

Johannes Köbler

Institut für Informatik
Humboldt-Universität zu Berlin

WS 2009/10

Kontextfreie Sprachen

Bemerkung

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

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

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

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

- 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 $A \to \alpha$ 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.
- 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 *ε-ableitbaren* Variablen:

```
E' := {A ∈ V | A → ε}
repeat
    E := E'
    E' := E ∪ {A ∈ V | ∃ B_1, ..., B_k ∈ E : A → B_1 ··· B_k}
until E = E'
```

- Nun bilden wir P' wie folgt:

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

$\square$

Entfernen von ε-Regeln

Beispiel

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

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

- Berechnung von E:

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

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

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

$\triangleleft$

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

REG $\subsetneq$ CFL $\subseteq$ CSL $\subseteq$ RE.

Beweis

- Es ist nur noch die Inklusion CFL $\subseteq$ CSL zu zeigen.
- Nach obigem Satz ex. zu $L \in$ 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 $L(G) = L \in$ 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. $L \in$ 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 $V_1 \cap V_2 = \emptyset$ und sei S eine neue Variable.
Dann erzeugen die kontextfreien Grammatiken

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

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

- Durchschnitt und
- Komplement?

Antwort

Nein.

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

Chomsky-Normalform

Definition

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

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)

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

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

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

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 $B \to \alpha$ in P die Regel $A \to \alpha$ zu P hinzunehmen. □

Entfernen von Variablenumbenennungen

Beispiel (Fortsetzung)

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

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

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

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

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

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

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

◁

Chomsky-Normalform

Definition

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

Anwendungen der Chomsky-Normalform

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

Entfernen von ε-Regeln und von 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 $A \to B_1 \cdots B_k$, $k \geq 3$, durch die $k - 1$ Regeln

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

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

Umwandlung in Chomsky-Normalform

Beispiel (Fortsetzung)

- Betrachte die Regeln

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

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

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

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

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

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

◁

Syntaxbäume

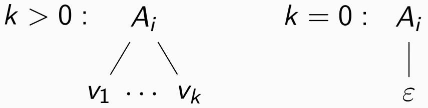
Definition

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 $A_i \rightarrow v_1 \cdots v_k$ 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 $v_1 \cdots v_k$ von links nach rechts geordnet vor.

Syntaxbäume

Beispiel

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

Syntaxbäume

Beispiel

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

$$S$$

Syntaxbäume

Beispiel

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

$$\underline{S} \Rightarrow aSbS$$

Syntaxbäume

Beispiel

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

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

Syntaxbäume

Beispiel

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

$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aaSbbS$$

Syntaxbäume

Beispiel

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

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

Syntaxbäume

Beispiel

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

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

Syntaxbäume

Beispiel

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

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

- Die zugehörigen Syntaxbäume sind dann

Syntaxbäume

Beispiel

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

T_0: S

S

Syntaxbäume

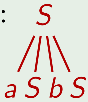
Beispiel

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

T_1:



$\underline{S} \Rightarrow aSbS$

Syntaxbäume

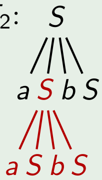
Beispiel

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

T_2:



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

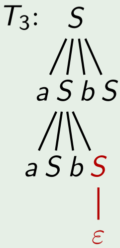
Syntaxbäume

Beispiel

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



$$\underline{S} \Rightarrow a\underline{S}bS \Rightarrow aaSb\underline{S}bS \Rightarrow aaSbbS$$

Syntaxbäume

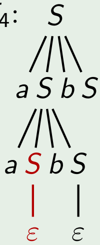
Beispiel

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

T_4:



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

Syntaxbäume

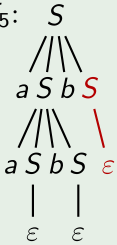
Beispiel

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

T_5:



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

Syntaxbäume

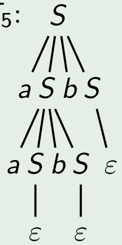
Beispiel

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

T_5:



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

◁

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.

Leicht zu sehen:

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

Syntaxbäume

Beispiel

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

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

◁

Syntaxbäume

Beispiel

- In $G = (\{S\}, \{a, b, c\}, \{S \to aSbS, \varepsilon\}, S)$ führen insgesamt 8 Ableitungen auf den Syntaxbaum

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

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

Syntaxbäume und Linksableitungen

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

Das Pumping-Lemma für kontextfreie Sprachen

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

Satz (Pumping-Lemma für kontextfreie Sprachen)

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

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

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

Abschätzung der Blätterzahl bei Binärbäumen

Definition

Die Tiefe eines Baumes mit Wurzel w ist die maximale Pfadlänge 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 mehr als 2^{k-1} Blättern hat mindestens die Tiefe k.

Beweis

Würde ein Binärbaum B der Tiefe $\leq k-1$ mehr als 2^{k-1} Blätter besitzen, so stünde dies im Widerspruch zu obigem Lemma. $\square$

Beweis des Pumping-Lemmas

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 $z = z_1 \cdots z_n \in L$ 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.

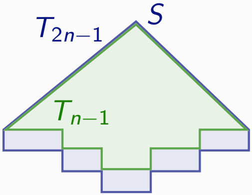
Beweis des Pumping-Lemmas

Beweis

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

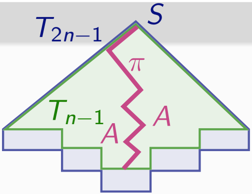
$$S = \alpha_0 \Rightarrow \alpha_1 \cdots \Rightarrow \alpha_m = z.$$

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

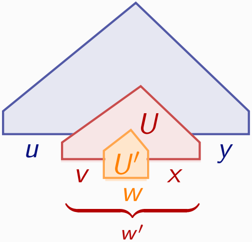


Beweis des Pumping-Lemmas

Beweis (Fortsetzung)



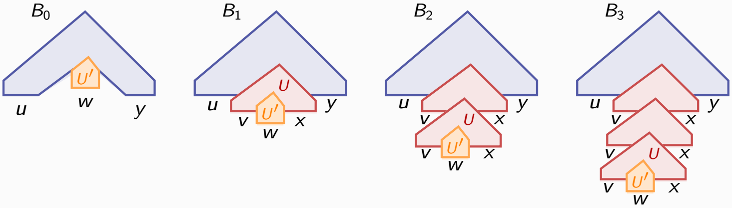
- Setzen wir $l = 2^k$, wobei $k = \|V\|$ ist, so hat T_{n-1} im Fall $n \geq l$ mindestens $l = 2^k > 2^{k-1}$ Blätter und daher mindestens die Tiefe k.
- Sei π ein von der Wurzel ausgehender Pfad maximaler Länge in T_{n-1}.
- Dann hat π mindestens die Länge k und unter den letzten $k+1$ Knoten von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



Beweis des Pumping-Lemmas

Beweis (Schluss)

- Da U mehr Blätter hat als U', ist $vx \neq \varepsilon$ (Bedingung 1).
- Da der Baum $U^* = U \cap T_{n-1}$ höchstens die Tiefe k hat (andernfalls wäre π nicht maximal), hat U^* (und damit U) höchstens $2^k = l$ Blätter.
- Folglich ist $|vwx| \leq l$ (Bedingung 2).
- Schließlich lassen sich Syntaxbäume B_i für die Wörter uv^iwx^iy, $i \geq 0$, wie folgt konstruieren (Bedingung 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:



□

Anwendung des Pumping-Lemmas

Beispiel

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

  Für jede Zerlegung $z = uvwxy$ mit $vx \neq \varepsilon$ und $|vwx| \leq l$ gehört $z' = uv^2wx^2y$ nicht zu L:

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

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

  - Also kann z' nicht zu L gehören. $\triangleleft$

Abschlusseigenschaften von CFL

Wie wir gesehen haben, ist die Klasse CFL abgeschlossen unter

- Vereinigung,
- Produkt und
- Sternhülle.

Satz

CFL ist nicht abgeschlossen unter

- Durchschnitt 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 Durchschnitt abgeschlossen.

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

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

Das Wortproblem für CFL

Das Wortproblem für kontextfreie Grammatiken

Gegeben: Eine kontextfreie Grammatik G und ein Wort x.

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

Frage

Ist das Wortproblem für kontextfreie Grammatiken effizient entscheidbar?

Der CYK-Algorithmus

Satz

Das Wortproblem für kontextfreie Grammatiken ist effizient entscheidbar.

Beweis

- Sei eine Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $x = x_1 \cdots x_n$ 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 $l = 1, \ldots, n$ und $k = 1, \ldots, n - l + 1$ die Menge $V_{l,k}$ aller Variablen, aus denen das Teilwort $x_k \cdots x_{k+l-1}$ ableitbar ist.
- Dann gilt $x \in L(G) \Leftrightarrow S \in V_{n,1}$.

Berechnung der Mengen $V_{l,k}$

Beweis (Schluss)

- Sei $G = (V, \Sigma, P, S)$ eine CNF-Grammatik und sei $x \in \Sigma^+$.
- Dann lassen sich die Mengen
  $$V_{l,k} = \{A \in V \mid A \Rightarrow^* x_k \cdots x_{k+l-1}\}$$
  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 Zahl $l' \in \{1, \ldots, l-1\}$ und eine Regel $A \to BC$ ex., so dass $B \in V_{l',k}$ und $C \in V_{l-l',k+l'}$ sind,
  $$V_{l,k} = \{A \in V \mid \exists l' < l \, \exists B \in V_{l',k} \, \exists C \in V_{l-l',k+l'} : A \to BC\}.$$
- Jede Menge $V_{l,k}$ lässt sich in Zeit $O(l|G|) = O(n|G|)$ bestimmen.
- Da insgesamt $n(n+1)/2 = O(n^2)$ Mengen $V_{l,k}$ zu bestimmen sind, ist dies in Zeit $O(n^3|G|)$ möglich. $\square$

Der CYK-Algorithmus

Algorithmus `CYK`(G, x)

Input: `CNF-Grammatik` $G = (V, \Sigma, P, S)$ `und Wort` $x = x_1 \cdots x_n$

  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**

      $V_{l,k} := \emptyset$

      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 leicht dahingehend modifizieren, 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

Der CYK-Algorithmus

Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

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

Der CYK-Algorithmus

Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

  $$\begin{aligned} P\colon\ & S \rightarrow AS', AY, BX, CS, c, \quad S' \rightarrow BC, \quad X \rightarrow AS, BX', a, \quad X' \rightarrow XX, \\ & Y \rightarrow BS, AY', b, \quad Y' \rightarrow YY, \quad A \rightarrow a, \quad B \rightarrow b, \quad C \rightarrow c. \end{aligned}$$

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

Der CYK-Algorithmus

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			
2			
3			

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	$\{X, A\}$		
2			
3			

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	$\{X, A\}$	$\{Y, B\}$	
2			
3			

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			
3			

Der CYK-Algorithmus

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	$\{S\}$		
3			

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			

Der CYK-Algorithmus

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	$\{Y\}$		

Der CYK-Algorithmus

Beispiel

- Betrachte die CNF-Grammatik mit den Regeln

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

- 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

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

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

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
$\{X, A\}$	$\{X, A\}$	$\{Y, B\}$	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$

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
$\{X, A\}$	$\{X, A\}$	$\{Y, B\}$	$\{X, A\}$	$\{Y, B\}$	$\{Y, B\}$
$\{X'\}$					

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

Der CYK-Algorithmus

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

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

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

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

Der CYK-Algorithmus

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

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

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

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

◁

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

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

◁

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

◁

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

Der CYK-Algorithmus

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

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

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

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

Der CYK-Algorithmus

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

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

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

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

◁

Der CYK-Algorithmus

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

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

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

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

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

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

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

Der CYK-Algorithmus

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

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

- 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

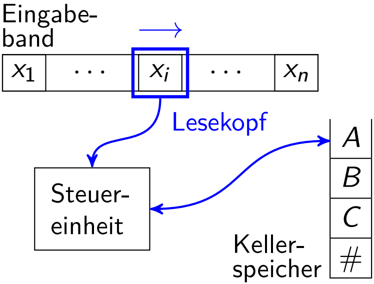
$$L = \{a^n b^n \mid n \geq 0\}$$

und alle anderen kontextfreien Sprachen erkennen zu können?

Antwort

- Dass ein DFA die Sprache L nicht erkennen kann, liegt an seinem beschränkten Speichervermögen, das zwar von L aber nicht von der Eingabe abhängen darf.
- Um L erkennen zu können, genügt bereits ein so genannter Kellerspeicher (auch Stapel, engl. *stack* oder *pushdown memory*).
- Dieser erlaubt nur den Zugriff auf die höchste belegte Speicheradresse.

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
- danach beliebig viele Symbole einkellern (mittels push-Operationen).

Formale Definition des Kellerautomaten

Notation

Für eine Menge M bezeichne $\mathcal{P}_e(M)$ die Menge aller endlichen Teilmengen von M, d.h.

$$\mathcal{P}_e(M) = \{A \subseteq M \mid A \text{ ist endlich}\}.$$

Definition

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

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

Der Kellerautomat

Arbeitsweise eines PDA

- Wenn p der momentane Zustand, A das oberste Kellerzeichen und $u \in \Sigma$ das nächste Eingabezeichen (bzw. $u = \varepsilon$) ist, so kann M im Fall $(q, B_1 \cdots B_k) \in \delta(p, u, A)$
  - 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 $B_1 \cdots B_k$ einkellern (danach ist B_1 das oberste Kellerzeichen).
- Hierfür sagen wir auch, M führt die Anweisung

  $$puA \rightarrow qB_1 \cdots B_k$$

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

Formale Definition der Konfiguration eines PDA

Definition

- Eine Konfiguration wird durch ein Tripel

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

  beschrieben und besagt, dass

  - p der momentane Zustand,
  - $x_i \cdots x_n$ der ungelesene Rest der Eingabe und
  - $A_1 \cdots A_l$ der aktuelle Kellerinhalt ist (A_1 ist oberstes Symbol).

- In der Konfiguration $K = (p, x_i \cdots x_n, A_1 \cdots A_l)$ kann M eine bel. Anweisung $puA_1 \to qB_1 \cdots B_k$ mit $u \in \{\varepsilon, x_i\}$ ausführen.

  Diese überführt M in die Folgekonfiguration

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

  Hierfür schreiben wir auch kurz $K \vdash K'$.

- Eine Rechnung von M bei Eingabe x ist eine Folge von Konfigurationen $K_0, K_1, K_2 \ldots$ 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.

Definition der von einem PDA erkannten Sprache

Notation

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

Definition

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

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

Bemerkung

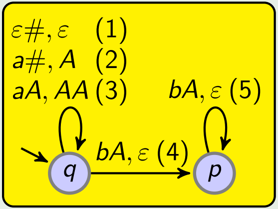
- Ein PDA M akzeptiert also genau dann eine Eingabe x, wenn es eine Rechnung gibt, bei der M
  - das gesamte Eingabewort bis zum Ende liest und
  - den Keller leert.
- Man beachte, dass bei leerem Keller kein weiterer Übergang mehr möglich ist.

Ein Kellerautomat

Beispiel

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

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



- 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:

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

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

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

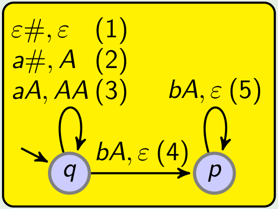
- Dies zeigt, dass M alle Wörter der Form $a^n b^n$, $n \geq 0$, akzeptiert.

Ein Kellerautomat

Beispiel

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

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



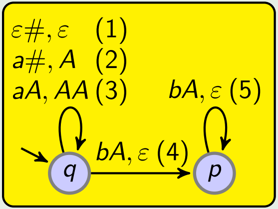
- Als nächstes zeigen wir, dass jede von M akzeptierte Eingabe $x = x_1 \dots x_n \in L(M)$ die Form $x = a^m b^m$ haben muss.
- Ausgehend von der Startkonfiguration $(q, x, \#)$ sind nur die Anweisungen (1) oder (2) ausführbar.
- Führt M zuerst Anweisung (1) aus, so wird der Keller geleert.
- Daher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

Beispiel

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

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



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

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

  mit $x_1 = x_2 = \cdots = x_m = a$ und $x_{m+1} = b$.
- Um den Keller leeren zu können, muss M nun noch genau $m - 1$ b's lesen, weshalb x auch in diesem Fall die Form $a^m b^m$ haben muss. $\triangleleft$

Ein Maschinenmodell für die Klasse CFL

Ziel

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

Satz

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

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

Idee:

- Konstruiere zu einer kontextfreien Grammatik $G = (V, \Sigma, P, S)$ einen PDA $M = (\{q\}, \Sigma, \Gamma, \delta, q, S)$ mit $\Gamma = V \cup \Sigma$, so dass folgende Äquivalenz gilt:

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

- Hierzu fügen wir folgende Anweisungen zu δ hinzu:

$$\begin{aligned} &\text{für jede Regel } A \rightarrow_G \alpha: && q\varepsilon A \rightarrow q\alpha, \\ &\text{für jedes Zeichen } a \in \Sigma: && qaa \rightarrow q\varepsilon. \end{aligned}$$

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

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

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

Beispiel

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

$$P\colon S \to aSbS \ \ (1), \quad S \to a \ \ (2).$$

- Der zugehörige PDA besitzt dann die Anweisungen

$$\begin{aligned} \delta: qaa &\to q\varepsilon \quad (0) & qbb &\to q\varepsilon \ \ (0') \\ q\varepsilon S &\to qaSbS \ \ (1') & q\varepsilon S &\to qa \ \ (2') \end{aligned}$$

- Der Linksableitung $\underline{S} \underset{(1)}{\Rightarrow} a\underline{S}bS \underset{(2)}{\Rightarrow} aab\underline{S} \underset{(2)}{\Rightarrow} aaba$ in G entspricht dann die Rechnung

$$\begin{aligned} (q, aaba, S) &\underset{(1')}{\vdash} (q, aaba, aSbS) \underset{(0)}{\vdash} (q, aba, SbS) \underset{(2')}{\vdash} (q, aba, abS) \\ &\underset{(0)}{\vdash} (q, ba, bS) \underset{(0')}{\vdash} (q, a, S) \underset{(2')}{\vdash} (q, a, a) \underset{(0)}{\vdash} (q, \varepsilon, \varepsilon) \end{aligned}$$

von M und umgekehrt. $\triangleleft$

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

Idee:

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

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

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

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

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

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

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

Beispiel

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

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

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

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

- Die Regelmenge P enthält neben den beiden Startregeln

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

die folgenden Produktionen:

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

Beispiel (Fortsetzung)

- Die Regelmenge P enthält neben den beiden Startregeln

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

die folgenden Produktionen:

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

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

Beispiel (Schluss)

- Die Anweisungen

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

  von M führen also auf die folgenden Regeln von G:

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

- Der akzeptierenden Rechnung

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

  von M entspricht dann in G die Linksableitung

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

◁

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

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

  $$puA \to p_0 A_1 \cdots A_k,\ k \geq 0$$

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

  $$X_{pAp_k} \to uX_{p_0A_1p_1} \cdots X_{p_{k-1}A_kp_k}.$$

- Dann lässt sich mit Hilfe der Äquivalenz

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

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

  $$\begin{aligned} x \in L(M) \quad &\Leftrightarrow \quad (q_0, x, \#) \vdash^* (p', \varepsilon, \varepsilon) \text{ für ein } p' \in Z \\ &\Leftrightarrow \quad S \Rightarrow X_{q_0\#p'} \Rightarrow^* x \text{ für ein } p' \in Z \\ &\Leftrightarrow \quad x \in L(G). \end{aligned}$$

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

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

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

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

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

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

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

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

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

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

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

- Sei $puA \to p_0 A_1 \cdots A_k$, $k \geq 0$, die im ersten Rechenschritt ausgeführte Anweisung (d.h. $x = ux'$).
- Für $i = 1, \ldots, k$ sei p_i der Zustand, in den M mit Kellerinhalt $A_{i+1} \cdots A_k$ gelangt (d.h. $p_k = p'$).
- Dann enthält P die Regel $X_{pAp_k} \to uX_{p_0 A_1 p_1} \cdots X_{p_{k-1} A_k p_k}$.
- Zudem sei u_i für $i = 1, \ldots, k$ das Teilwort von x', das M zwischen den Besuchen von p_{i-1} und p_i liest.
- Dann ex. Zahlen $m_i \geq 1$ mit $m_1 + \cdots + m_k = m$ und

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

- Nach IV gibt es daher Ableitungen

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

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

Induktionsschritt für die Implikation von links nach rechts

- Nach IV gibt es daher Ableitungen

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

  die wir zu der gesuchten Ableitung zusammensetzen können:

  $$\begin{aligned} X_{pAp_k} &\Rightarrow && uX_{p_0 A_1 p_1} \cdots X_{p_{k-2} A_{k-1} p_{k-1}} X_{p_{k-1} A_k p_k} \\ &\Rightarrow^{m_1} && uu_1 X_{p_1 A_2 p_2} \cdots X_{p_{k-2} A_{k-1} p_{k-1}} X_{p_{k-1} A_k p_k} \\ &\vdots && \\ &\Rightarrow^{m_{k-1}} && uu_1 \cdots u_{k-1} X_{p_{k-1} A_k p_k} \\ &\Rightarrow^{m_k} && uu_1 \cdots u_k = x. \end{aligned}$$

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

Induktionsschritt für die Implikation von rechts nach links

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

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

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

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

- Nach IV gibt es somit Rechnungen

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

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

Induktionsschritt für die Implikation von rechts nach links

- Nach IV gibt es somit Rechnungen

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

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

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

$\square$

Deterministische Kellerautomaten

Von besonderem Interesse sind kontextfreie Sprachen, die von einem deterministischen Kellerautomaten erkannt werden.

Definition

- Ein Kellerautomat heißt deterministisch, falls $\vdash$ eine rechtseindeutige Relation 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 (siehe Übungen):

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

Deterministische Kellerautomaten

Beispiel

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

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

  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	–	$q_1 A$	$q_1 B$	–	–	–	–	–	–

- Offenbar erkennt M die Sprache $L(M) = \{xcx^R \mid x \in \{a, b\}^+\}$.
- Man beachte, dass jedes Tabellenfeld höchstens eine Anweisung enthält und jede Spalte mit einem ε-Übergang keine weiteren Anweisungen enthält.
- Daher ist die Bedingung $\|\delta(q, a, A)\| + \|\delta(q, \varepsilon, A)\| \leq 1$ für alle $q \in Z$, $a \in \Sigma$ und $A \in \Gamma$ erfüllt. $\triangleleft$

Deterministische Kellerautomaten

Frage

- Können deterministische Kellerautomaten zumindest alle regulären Sprachen durch Leeren des Kellers akzeptieren?
- Kann z.B. die Sprache $L = \{a, aa\}$ von einem deterministischen Kellerautomaten M durch Leeren des Kellers 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 Kellerautomaten können also durch Leeren des Kellers nur präfixfreie Sprachen L akzeptieren (d.h. kein Wort $x \in L$ ist Präfix eines anderen Wortes in L).

Lösung des Problems

Wir vereinbaren, dass deterministische Kellerautomaten ihre Eingabe durch Erreichen eines Endzustands akzeptieren dürfen.

Deterministische Kellerautomaten

Definition

- Ein Kellerautomat mit Endzuständen wird durch ein 7-Tupel $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ beschrieben.
- Dabei sind die Komponenten $Z, \Sigma, \Gamma, \delta, q_0, \#$ wie bei einem PDA.
- Zusätzlich ist $E \subseteq Z$ eine Menge von Endzuständen.
- Die von M akzeptierte oder erkannte Sprache ist

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

- M ist ein det. Kellerautomat mit Endzuständen (kurz: DPDA), 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

$$\text{DCFL} = \{L(M) \mid M \text{ ist ein DPDA}\}$$

(*deterministic context free languages*).

Charakterisierung von DCFL mittels Grammatiken

- Die Klasse DCFL lässt sich auch mit Hilfe von speziellen kontextfreien Grammatiken charakterisieren, den so genannten $LR(k)$-Grammatiken.
- Der erste Buchstabe L steht für die Leserichtung bei der Syntaxanalyse, d.h. das Eingabewort x wird von links (nach rechts) gelesen.
- Der zweite Buchstabe R bedeutet, dass bei der Syntaxanalyse eine Rechtsableitung entsteht.
- Schließlich gibt der Parameter k an, wieviele Zeichen man in der Eingabe vorauslesen muss, damit die nächste anzuwendende Regel eindeutig feststeht (k wird auch als *Lookahead* bezeichnet).
- Durch $LR(0)$-Grammatiken lassen sich nur die präfixfreien Sprachen in DCFL erzeugen.
- Dagegen erzeugen die $LR(k)$-Grammatiken für jedes $k \geq 1$ genau die Sprachen in DCFL.
- Daneben gibt es noch $LL(k)$-Grammatiken, die für wachsendes k immer mehr deterministisch kontextfreie Sprachen erzeugen.

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

Der nächste Satz zeigt, wie sich Problem 1 beheben lässt.

Satz

Jede Sprache $L \in \text{DCFL}$ wird von einem DPDA M' erkannt, der alle Eingaben zu Ende liest.

Beweis.

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

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

DPDAs, die ihre Eingabe zu Ende lesen

Beweis (Fortsetzung)

- Die erste Ursache schließen wir aus, indem wir ein neues Zeichen $\square$ auf dem Kellerboden platzieren:

  (a) $s\varepsilon\# \rightarrow q_0\#\square$ (dabei sei s ein neuer Startzustand).

- Die zweite Ursache schließen wir durch Hinzunahme eines Fehlerzustands f sowie folgender Anweisungen aus:

  (b) $qaA \rightarrow fA$, für alle $(q, a, A) \in Z \times \Sigma \times \Gamma'$ mit $A = \square$ oder $\delta(q, a, A) = \delta(q, \varepsilon, A) = \emptyset$ (hierbei ist $\Gamma' = \Gamma \cup \{\square\}$),

  (c) $faA \rightarrow fA$, für alle $a \in \Sigma$ und $A \in \Gamma'$.

DPDAs, die ihre Eingabe zu Ende lesen

Beweis (Fortsetzung)

- Als nächstes verhindern wir die Ausführung einer unendlichen Folge von ε-Übergängen.

  Dabei unterscheiden wir die beiden Fälle, ob M hierbei auch Endzustände besucht oder nicht.

  (d) $q\varepsilon A \rightarrow fA$, für alle $q \in Z$ und $A \in \Gamma$, so dass M ausgehend von der Konfiguration (q, ε, A) unendlich viele ε-Übergänge ausführt ohne dabei einen Endzustand zu besuchen.

  Falls ja, sehen wir einen Umweg über den neuen Endzustand e vor.

  (e) $q\varepsilon A \rightarrow eA$, $e\varepsilon A \rightarrow fA$, für alle $q \in Z$ und $A \in \Gamma$, so dass M ausgehend von der Konfiguration (q, ε, A) unendlich viele ε-Übergänge ausführt und dabei auch Endzustände besucht.

- Schließlich übernehmen wir von M noch

  (f) alle Anweisungen aus δ, soweit sie nicht durch Anweisungen vom Typ (d) oder (e) überschrieben wurden.

DPDAs, die ihre Eingabe zu Ende lesen

Beweis (Schluss)

Um dies zu verhindern, transformieren wir $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ in den DPDA

$$M' = (Z \cup \{s, e, f\}, \Sigma, \Gamma', \delta', s, \#, E \cup \{e\}) \text{ mit } \Gamma' = \Gamma \cup \{\square\},$$

wobei δ' folgende Anweisungen enthält:

(a) $s\varepsilon\# \to q_0\#\square$,

(b) $qaA \to fA$, für alle $(q, a, A) \in Z \times \Sigma \times \Gamma'$ mit $A = \square$ oder $\delta(q, a, A) = \delta(q, \varepsilon, A) = \emptyset$,

(c) $faA \to fA$, für alle $a \in \Sigma$ und $A \in \Gamma'$,

(d) $q\varepsilon A \to fA$, für alle $q \in Z$ und $A \in \Gamma$, so dass ausgehend von der Konfiguration (q, ε, A) unendlich viele ε-Übergänge ausgeführt werden, ohne dass dabei ein Endzustand besucht wird.

(e) $q\varepsilon A \to eA$
$e\varepsilon A \to fA$, für alle $q \in Z$ und $A \in \Gamma$, so dass ausgehend von der Konfiguration (q, ε, A) unendlich viele ε-Übergänge ausgeführt und dabei auch Endzustände besucht werden,

(f) alle Anweisungen aus δ, soweit sie nicht durch Anweisungen vom Typ (c) oder (e) überschrieben wurden. $\square$

DPDAs, die ihre Eingabe zu Ende lesen

Beispiel

Wenden wir diese Konstruktion auf den DPDA

$$M = (\{q_0, q_1, q_2\}, \{a, b, c\}, \{A, B, \#\}, \delta, q_0, \#, \{q_2\})$$

mit der Überführungsfunktion

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	–	–	$q_2\#$	–	–
a	$q_0A\#$	q_0AA	q_0AB	–	q_1	–	–	–	–
b	$q_0B\#$	q_0BA	q_0BB	–	–	q_1	–	–	–
c	–	q_1A	q_1B	–	–	–	–	–	–

an, so erhalten wir den DPDA

$$M' = (\{q_0, q_1, q_2, s, e, f\}, \{a, b, c\}, \{A, B, \#, \square\}, \delta', s, \#, \{q_2, e\})$$

mit folgender Überführungsfunktion δ':

DPDAs, die ihre Eingabe zu Ende lesen

Beispiel (Schluss)

δ'	$q_0,\#$	q_0,A	q_0,B	$q_0,\Box$	$q_1,\#$	q_1,A	q_1,B	$q_1,\Box$	$q_2,\#$	q_2,A	q_2,B	$q_2,\Box$
ε	–	–	–	–	q_2	–	–	–	$e\#$	–	–	–
a	$q_0A\#$	q_0AA	q_0AB	$f\Box$	–	q_1	fB	$f\Box$	–	fA	fB	$f\Box$
b	$q_0B\#$	q_0BA	q_0BB	$f\Box$	–	fA	q_1	$f\Box$	–	fA	fB	$f\Box$
c	$f\#$	q_1A	q_1B	$f\Box$	–	fA	fB	$f\Box$	–	fA	fB	$f\Box$
Typ	(f,b)	(f)	(f)	(b)	(f)	(f,b)	(f,b)	(b)	(e)	(b)	(b)	(b)

	$s,\#$	s,A	s,B	$s,\Box$	$f,\#$	f,A	f,B	$f,\Box$	$e,\#$	e,A	e,B	$e,\Box$
ε	$q_0\#\Box$	–	–	–	–	–	–	–	$f\#$	–	–	–
a	–	–	–	–	$f\#$	fA	fB	$f\Box$	–	–	–	–
b	–	–	–	–	$f\#$	fA	fB	$f\Box$	–	–	–	–
c	–	–	–	–	$f\#$	fA	fB	$f\Box$	–	–	–	–
Typ	(a)				(c)	(c)	(c)	(c)	(e)			

◁

Komplementabschluss von DCFL

Satz

Die Klasse DCFL ist unter Komplement abgeschlossen.

Beweis

- 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.
- Dabei merkt sich $\overline{M}$ in seinem Zustand (q, i) neben dem aktuellen Zustand q von M in der Komponente i, ob M nach Lesen des letzten Zeichens (bzw. seit Rechnungsbeginn) einen Endzustand besucht hat $(i = 2)$ oder nicht $(i = 1)$.
- Möchte M das nächste Zeichen lesen und befindet sich $\overline{M}$ im Zustand $(q, 1)$, so macht $\overline{M}$ noch einen Umweg über den Endzustand $(q, 3)$.

Komplementabschluss von DCFL

Beweis (Schluss)

- Konkret sei $\overline{M} = (Z \times \{1, 2, 3\}, \Sigma, \Gamma, \delta', s, \#, Z \times \{3\})$ mit

$$s = \begin{cases} (q_0, 1), & q_0 \notin E, \\ (q_0, 2), & \text{sonst}, \end{cases}$$

wobei δ' für jede Anweisung $q\varepsilon A \to_M p\gamma$ die Anweisungen

$$\begin{aligned} &(q, 1)\varepsilon A \to (p, 1)\gamma, && \text{falls } p \notin E, \\ &(q, 1)\varepsilon A \to (p, 2)\gamma, && \text{falls } p \in E \text{ und} \\ &(q, 2)\varepsilon A \to (p, 2)\gamma, \end{aligned}$$

sowie für jede Anweisung $qaA \to_M p\gamma$ folgende Anweisungen enthält:

$$\begin{aligned} &(q, 1)\varepsilon A \to (q, 3)A, \\ &(q, 2)aA \to (p, 1)\gamma, && \text{falls } p \notin E, \\ &(q, 2)aA \to (p, 2)\gamma, && \text{falls } p \in E, \\ &(q, 3)aA \to (p, 1)\gamma, && \text{falls } p \notin E \text{ und} \\ &(q, 3)aA \to (p, 2)\gamma, && \text{falls } p \in E. \end{aligned}$$

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

Komplementabschluss von DCFL

Beispiel

- Angenommen, ein DPDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ führt bei Eingabe $x = a$ folgende Rechnung aus:

$$(q_0, a, \#) \vdash (q_1, \varepsilon, \gamma_1) \vdash (q_2, \varepsilon, \gamma_2).$$

- Dann würde $\overline{M}$ im Fall $E = \{q_0, q_2\}$ (d.h. $x \in L(M)$) die Rechnung

$$((q_0, 2), a, \#) \vdash ((q_1, 1), \varepsilon, \gamma_1) \vdash ((q_2, 2), \varepsilon, \gamma_2)$$

  ausführen und das Wort a verwerfen, da $(q_1, 1), (q_2, 2) \notin Z \times \{3\}$ sind.

- Im Fall $E = \{q_0\}$ (d.h. $x \notin L(M)$) würde $\overline{M}$ dagegen die Rechnung

$$((q_0, 2), a, \#) \vdash ((q_1, 1), \varepsilon, \gamma_1) \vdash ((q_2, 1), \varepsilon, \gamma_2) \vdash ((q_2, 3), \varepsilon, \gamma_2)$$

  ausführen und das Wort a akzeptieren, da $(q_2, 3) \in Z \times \{3\}$ ist. ◁

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

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

Weitere Abschlusseigenschaften von DCFL

Satz

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

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

- 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.

Beweis von $L_1, L_2 \in \mathsf{DCFL} \not\Rightarrow L_1 \cup L_2 \in \mathsf{DCFL}$

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

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

- Ihre Vereinigung gehört aber nicht zu DCFL, d.h.

  $$L_3 \cup L_4 = \{a^i b^j c^k \mid i \neq j \text{ oder } j \neq k\} \in \mathsf{CFL} \setminus \mathsf{DCFL}.$$

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

  $$(\overline{L_3 \cup L_4}) \cap L(a^* b^* c^*) = \{a^n b^n c^n \mid n \geq 0\}$$

  (deterministisch) kontextfrei.

Beweis von $L_1, L_2 \in \mathrm{DCFL} \not\Rightarrow L_1L_2 \in \mathrm{DCFL}$

- Betrachte die DCFL Sprachen

$$L_0 = \{0\},\ L_3 = \{a^ib^jc^k \mid i \neq j\} \text{ und } L_4 = \{a^ib^jc^k \mid j \neq k\}.$$

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

$$L_5 = L_0(L_3 \cup L_4) = \{0a^ib^jc^k \mid i \neq j \vee j \neq k\} \notin \mathrm{DCFL},$$

  da sich ein DPDA $M = (Z, \Sigma, \Gamma, \delta, q_0, \#, E)$ für L_5 leicht zu einem DPDA für $L_3 \cup L_4$ 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 $L_3 \cup L_4$, wobei δ' wie folgt definiert ist:

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

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

- Betrachte die DCFL Sprachen

$$L_0 = \{0\},\ L_3 = \{a^i b^j c^k \mid i \neq j\} \text{ und } L_4 = \{a^i b^j c^k \mid j \neq k\}.$$

- Es ist leicht zu sehen, dass auch die beiden Sprachen L_0^* und $L = L_0 L_3 \cup L_4$ in DCFL sind (siehe Übungen).
- Ihr Produkt $L_0^* L$ gehört aber nicht zu DCFL.
- Da DCFL unter Schnitt mit regulären Sprachen abgeschlossen ist, wäre andernfalls auch die Sprache

$$L_0^* L \cap L_0 L(a^* b^* c^*) = \{0a^i b^j c^k \mid i \neq j \vee j \neq k\} = L_0(L_3 \cup L_4) = L_5$$

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*