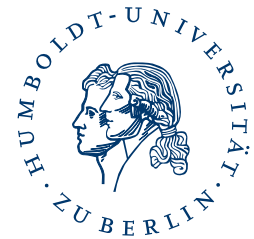
Einführung in die Theoretische Informatik

Johannes Köbler



Institut für Informatik
Humboldt-Universität zu Berlin

WS 2016/17

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:

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

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

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

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

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

Kontextfreie 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.
- Beispielsweise ist die Grammatik $G = (\{S\}, \{a, b\}, \{S \to aSb, S \to \varepsilon\}, S)$ nicht kontextsensitiv, da sie die Regel $S \to \varepsilon$ enthält, obwohl S auf der rechten Seite der Regel $S \to aSb$ vorkommt.
- Wir werden jedoch sehen, dass sich zu jeder kontextfreien Grammatik eine äquivalente kontextsensitive Grammatik konstruieren lässt.

Entfernen von ε-Regeln

Satz

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

Beweis

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

```
1   E' := {A ∈ V | A → ε}
2   repeat
3     E := E'
4     E' := E ∪ {A ∈ V | ∃ B_1, ..., B_k ∈ E : A → B_1 ... B_k}
5   until E = E'
```

- Nun bilden wir P' wie folgt:

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

□

Entfernen von ε-Regeln

Beispiel

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

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

- Berechnung von E:

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

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

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

Die Chomsky-Hierarchie

Satz

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

Korollar

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

Beweis

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

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

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

Abschlusseigenschaften von CFL

Satz

CFL ist abgeschlossen unter Vereinigung, Produkt und Sternhülle.

Beweis

Seien $G_1 = (V_1, \Sigma, P_1, S_1)$ und $G_2 = (V_2, \Sigma, P_2, S_2)$ kontextfreie Grammatiken mit $V_1 \cap V_2 = \varnothing$ und sei S eine neue Variable.
Dann erzeugen die kontextfreien Grammatiken

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

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

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

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

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

die Sternhülle $L(G_1)^*$. □

Abschlusseigenschaften von CFL

Satz

CFL ist abgeschlossen unter Vereinigung, Produkt und Sternhülle.

Frage

Ist die Klasse CFL auch abgeschlossen unter

- Schnitt und
- Komplement?

Antwort

Nein.

Hierzu müssen wir für bestimmte Sprachen nachweisen, dass sie nicht kontextfrei sind. Dies gelingt mit einem Pumping-Lemma für kontextfreie Sprachen, 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.

Satz

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

Anwendungen der Chomsky-Normalform

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

Das 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 $i \geq 0$.

Das Wortproblem für CFL

Das Wortproblem für kontextfreie Grammatiken

Gegeben: Eine kontextfreie Grammatik G und ein Wort x.

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

Satz

Das Wortproblem für kontextfreie Grammatiken ist effizient entscheidbar.

Chomsky-Normalform

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

Definition

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

Entfernen von Variablenumbenennungen

Satz

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

Beweis

- Zuerst entfernen wir sukzessive alle Zyklen

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

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

Entfernen von Variablenumbenennungen

Beispiel (Fortsetzung)

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

- 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)

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

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

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

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

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

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

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

Entfernen von ε-Regeln und Variablenumbenennungen

Bereits gezeigt:

Korollar

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

Noch zu zeigen:

Satz

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

Umwandlung in Chomsky-Normalform

Satz

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

Beweis

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

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

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

Umwandlung in Chomsky-Normalform

Beispiel (Fortsetzung)

- Betrachte die Regeln

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

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

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

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

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

◁

Links- und Rechtsableitungen

Definition

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

- Eine Ableitung

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

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

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

Leicht zu sehen:

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

Ein- und mehrdeutige Grammatiken

Beispiel

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

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

◁

Ein- und mehrdeutige Grammatiken

Beispiel

- Die Grammatik $G = (\{S\}, \{a, b\}, \{S \to aSbS, \varepsilon\}, S)$ ist eindeutig.
- Dies liegt daran, dass in jeder Satzform $\alpha S \beta$ von G das Suffix β entweder leer ist oder mit einem b beginnt.
- 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, wenn in x auf das Präfix α 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 $v_0, \ldots, v_k$ mit $(v_i, v_{i+1}) \in E$ für $i = 0, \ldots, k-1$. Seine Länge ist k.
- Ein Weg heißt Pfad, falls alle Knoten paarweise verschieden sind.
- Ein u-v-Weg der Länge ≥ 1 mit $u = v$ heißt Zyklus.
- G heißt azyklisch, wenn es in G keinen Zyklus gibt.
- G heißt gerichteter Wald, wenn G azyklisch ist und jeder Knoten $v \in V$ Eingangsgrad $\deg^-(v) \leq 1$ hat.
- Ein Knoten $u \in V$ vom Ausgangsgrad $\deg^+(u) = 0$ heißt Blatt.
- Ein Knoten $w \in V$ heißt Wurzel von G, falls alle Knoten $v \in V$ von w aus erreichbar sind (d.h. es gibt einen w-v-Weg in G).
- Ein gerichteter Wald, der eine Wurzel hat, heißt gerichteter Baum.
- Da die Kantenrichtungen durch die Wahl der Wurzel eindeutig bestimmt sind, kann auf ihre Angabe verzichtet werden. Man spricht dann auch von einem Wurzelbaum.

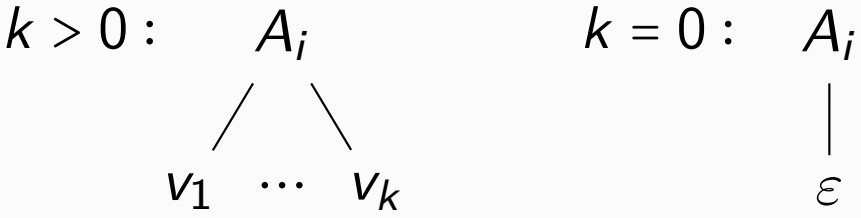
Syntaxbäume

Wir ordnen einer Ableitung

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

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

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



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

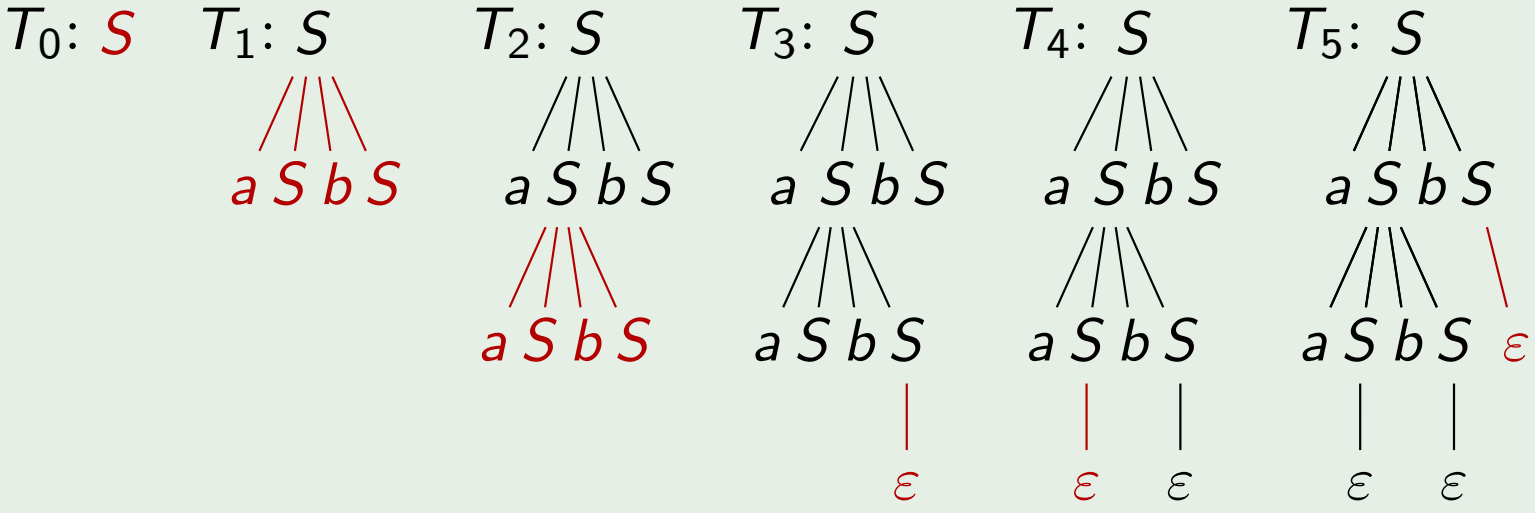
Syntaxbäume

Beispiel

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

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

- Die zugehörigen Syntaxbäume sind dann

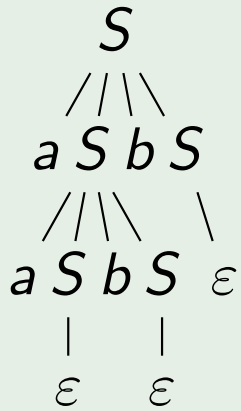


◁

Syntaxbäume

Beispiel

- In $G = (\{S\}, \{a, b\}, \{S \to aSbS, \varepsilon\}, S)$ führen alle 8 Ableitungen des Wortes *aabb* auf denselben Syntaxbaum:



◁

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. □

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). □

Das Pumping-Lemma für kontextfreie Sprachen

Satz (Pumping-Lemma für kontextfreie Sprachen)

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

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

Das Pumping-Lemma für kontextfreie Sprachen

Beispiel

- Betrachte die Sprache $L = \{a^n b^n | n \geq 0\}$.
- Dann lässt sich jedes Wort $z = a^n b^n = a^{n-1} abb^{n-1}$ 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 $uv^i wx^i y = a^{n-1} a^i b^i b^{n-1} \in L$. ◁

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 $z' = uv^0wx^0y = uwy$ 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 $\#_a(z) = \#_a(z')$ und somit gilt

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

  - Also gehört z' nicht zu L. ◁

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

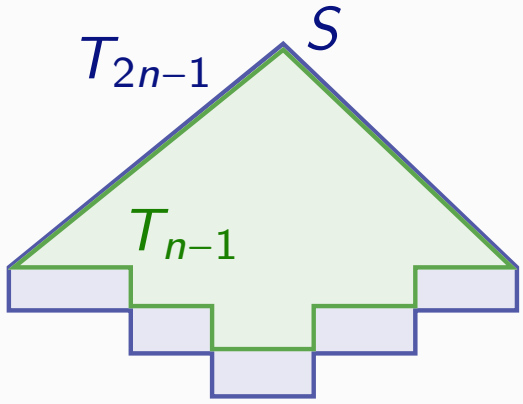
Beweis des Pumping-Lemmas

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

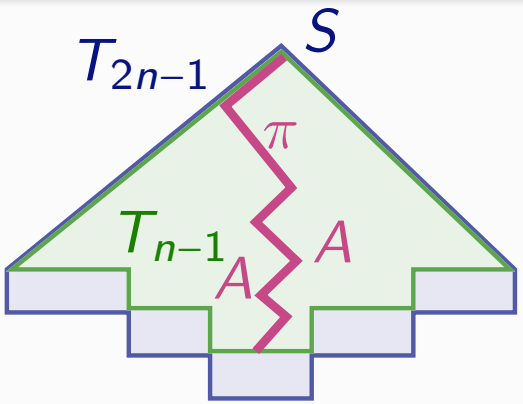
$$S = \alpha_0 \Rightarrow \alpha_1 \cdots \Rightarrow \alpha_m = z.$$

- 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.
- Folglich ist $m = 2n-1$ und z hat den Syntaxbaum T_{2n-1}.
- Wir können annehmen, dass die $n-1$ Regeln der Form $A \to BC$ vor den n Regeln der Form $A \to a$ 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 die Tiefe k, da T_{n-1} mindestens $l = 2^k > 2^{k-1}$ Blätter hat.

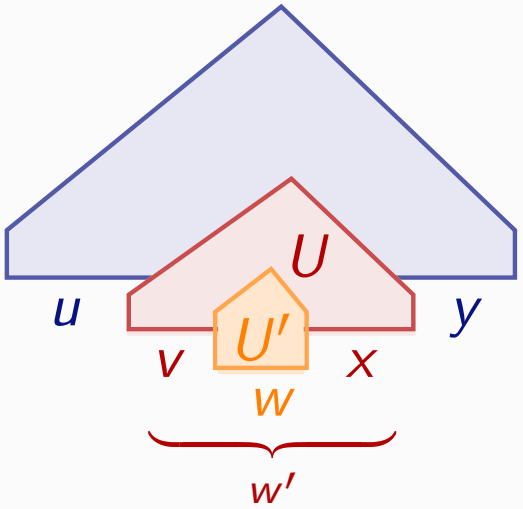


Beweis des Pumping-Lemmas

Beweis (Fortsetzung)



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

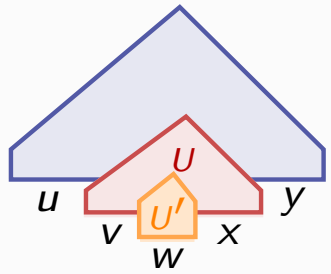


Beweis des Pumping-Lemmas

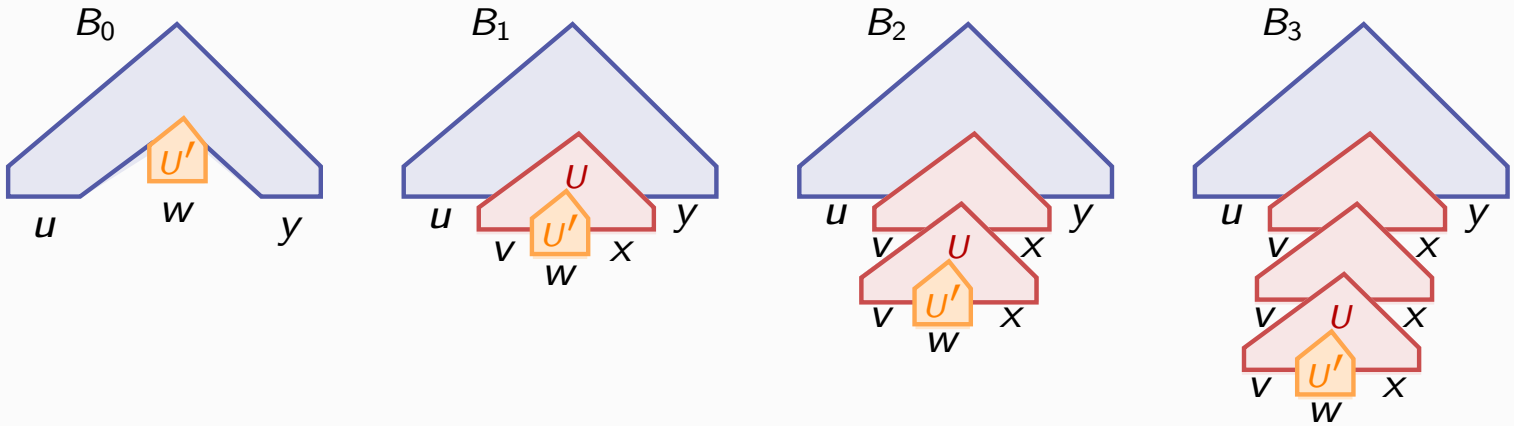
Beweis (Schluss)

- Dann ist $vx \neq \varepsilon$ (Bed. 1), da U mehr Blätter hat als U'.
- Zudem hat U höchstens $2^k = l$ Blätter, da der Baum $U^* = U \cap T_{n-1}$ höchstens die Tiefe k hat (andernfalls wäre π nicht maximal).

  Folglich ist $|vwx| \leq l$ (Bed. 2).



- Schließlich lassen sich Syntaxbäume B_i für die Wörter uv^iwx^iy, $i \geq 0$, wie folgt konstruieren (Bed. 3):
  - B_0 entsteht aus $B_1 = T_{2n-1}$, indem wir U durch U' ersetzen.
  - B_{i+1} entsteht aus B_i, indem wir U' durch U ersetzen:



□

Abschlusseigenschaften von CFL

Wie wir gesehen haben, ist die Klasse CFL abgeschlossen unter

- Vereinigung,
- Produkt und
- Sternhülle.

Satz

CFL ist nicht abgeschlossen unter

- Schnitt und
- Komplement.

Abschlusseigenschaften von CFL

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

- Die beiden Sprachen

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

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

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

Wäre CFL unter Komplement abgeschlossen, so wäre CFL wegen de Morgan auch unter Schnitt abgeschlossen:

$$A, B \in \text{CFL} \Rightarrow \overline{A}, \overline{B} \in \text{CFL} \Rightarrow \overline{A} \cup \overline{B} \in \text{CFL} \Rightarrow \overline{\overline{A} \cup \overline{B}} = A \cap B \in \text{CFL} \; \lightning$$

Das Wortproblem für CFL

Das Wortproblem für kontextfreie Grammatiken

Gegeben: Eine kontextfreie Grammatik G und ein Wort x.

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

Frage

Wie lässt sich das Wortproblem für kontextfreie Grammatiken entscheiden?

Der CYK-Algorithmus

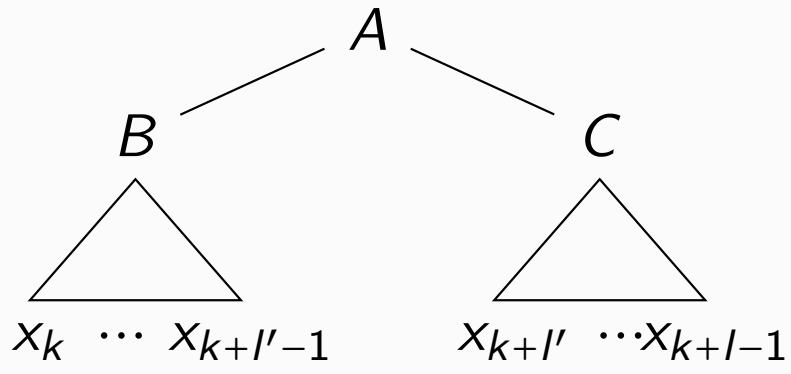
- Sei eine Grammatik $G = (V, \Sigma, P, S)$ und ein Wort $x = x_1 \dots x_n$ gegeben.
- Falls $x = \varepsilon$ ist, können wir effizient prüfen, ob $S \Rightarrow^* \varepsilon$ gilt.
- Hierzu genügt es, die Menge $E = \{A \in V \mid A \Rightarrow^* \varepsilon\}$ aller ε-ableitbaren Variablen zu berechnen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

Berechnung der Mengen $V_{l,k}$

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

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

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



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

Der CYK-Algorithmus

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

```
1   Input: CNF–Grammatik G = (V, Σ, P, S) und Wort x = x_1 ... x_n
2     for k := 1 to n do
3       V_{1,k} := {A ∈ V | A → x_k ∈ P}
4     for l := 2 to n do
5       for k := 1 to n − l + 1 do
6         V_{l,k} := ∅
7         for l' := 1 to l − 1 do
8           for all A → BC ∈ P do
9             if B ∈ V_{l',k} and C ∈ V_{l−l',k+l'} then
10              V_{l,k} := V_{l,k} ∪ {A}
11    if S ∈ 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

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

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

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

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

Der CYK-Algorithmus

Beispiel (Fortsetzung)

- Betrachte die CNF-Grammatik mit den Regeln

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

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

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

Ein Maschinenmodell für die kontextfreien Sprachen

Frage

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

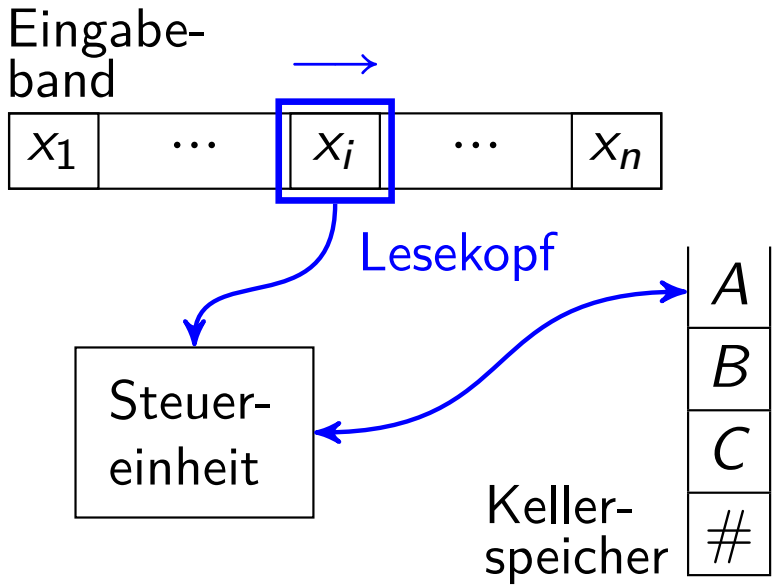
$$L = \{a^n b^n \mid n \geq 0\}$$

und alle anderen kontextfreien Sprachen erkennen zu können?

Antwort

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

Der Kellerautomat



- verfügt zusätzlich über einen Kellerspeicher,
- kann auch ε-Übergänge machen,
- hat Lesezugriff auf das aktuelle Eingabezeichen und auf das oberste Kellersymbol,
- kann das oberste Kellersymbol löschen (durch eine pop-Operation) und
- durch beliebig viele Symbole ersetzen (durch eine push-Operation).

Formale Definition des Kellerautomaten

Notation

Für eine (unendliche) 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 $(q, B_1 \dots B_k) \in \delta(p, u, A)$
  - in den Zustand q wechseln,
  - den Lesekopf auf dem Eingabeband um $|u| \in \{0, 1\}$ Positionen vorrücken und
  - das Zeichen A aus- sowie die Zeichenfolge $B_1 \dots B_k$ einkellern (danach ist B_1 das oberste Kellerzeichen).
- Hierfür sagen wir auch, M führt die Anweisung

  $$puA \to qB_1 \dots B_k$$

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

Formale Definition der Konfiguration eines PDA

- Eine Konfiguration wird durch ein Tripel

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

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

  Diese überführt M in die Folgekonfiguration

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

  Hierfür schreiben wir auch kurz $K \vdash K'$.
- Eine Rechnung von M bei Eingabe x ist eine Folge von Konfigurationen $K_0, K_1, K_2 \dots$ mit $K_0 = (q_0, x, \#)$ und $K_0 \vdash K_1 \vdash K_2 \cdots$.

  K_0 heißt Startkonfiguration von M bei Eingabe x.

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

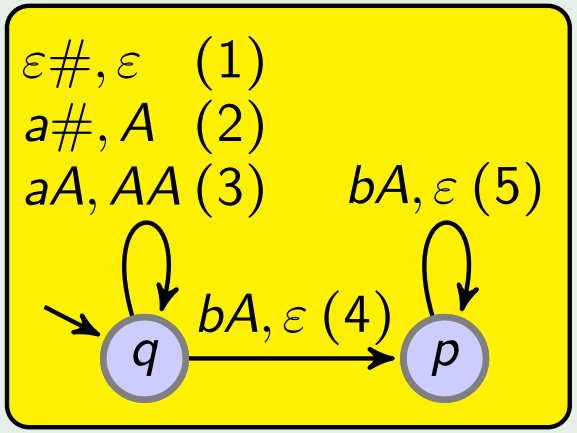
- Ein PDA M akzeptiert also genau dann eine Eingabe x, wenn es eine Rechnung gibt, bei der M
  - das gesamte Eingabewort bis zum Ende liest und
  - den Keller leert.
- Man beachte, dass bei leerem Keller kein weiterer Übergang mehr möglich ist.

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:

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

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

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

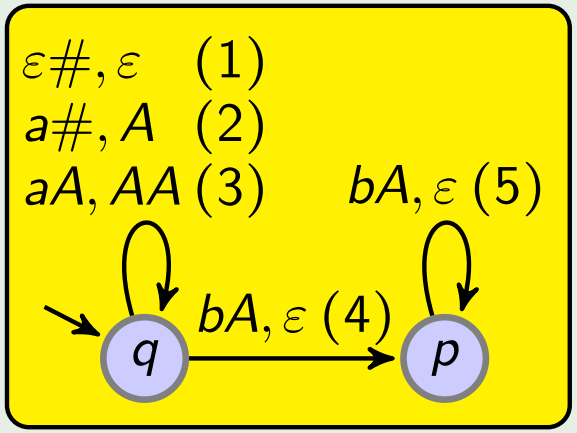
- Dies zeigt, dass M alle Wörter der Form $a^n b^n$, $n \geq 0$, akzeptiert.

Ein Kellerautomat

Beispiel

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

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



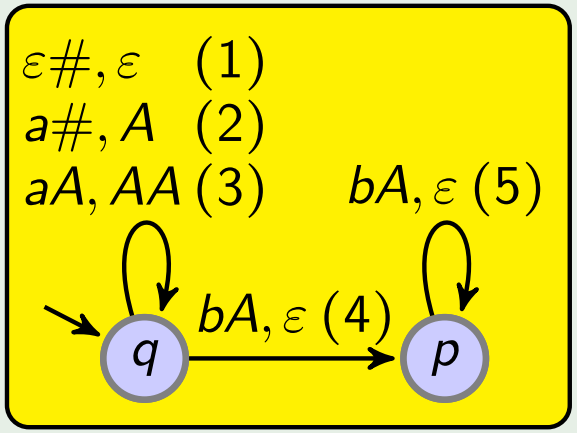
- Als nächstes zeigen wir, dass jede von M akzeptierte Eingabe $x = x_1 \dots x_n \in L(M)$ die Form $x = a^m b^m$ haben muss.
- Ausgehend von der Startkonfiguration $(q, x, \#)$ sind nur die Anweisungen (1) oder (2) ausführbar.
- Führt M zuerst Anweisung (1) aus, so wird der Keller geleert.
- Daher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

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



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

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

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

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

Ein Maschinenmodell für die Klasse CFL

Ziel

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

Satz

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

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

Idee:

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

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

- Hierzu fügen wir folgende Anweisungen zu δ hinzu:

  für jede Regel $A \to_G \alpha$: $q\varepsilon A \to q\alpha$

  für jedes Zeichen $a \in \Sigma$: $qaa \to q\varepsilon$

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

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

□

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

  $$\delta : qaa \to q \quad (0) \qquad qbb \to q \quad (0')$$
  $$q\varepsilon S \to qaSb \quad (1') \qquad q\varepsilon S \to q \quad (2')$$

- Der Linksableitung $S \underset{(1)}{\Rightarrow} aSb \underset{(1)}{\Rightarrow} aaSbb \underset{(2)}{\Rightarrow} aabb$ in G entspricht dann die Rechnung

  $$(q, aabb, S) \underset{(1')}{\vdash} (q, aabb, aSb) \underset{(0)}{\vdash} (q, abb, Sb) \underset{(1')}{\vdash} (q, abb, aSbb)$$
  $$\underset{(0)}{\vdash} (q, bb, Sbb) \underset{(2')}{\vdash} (q, bb, bb) \underset{(0')}{\vdash} (q, b, b) \underset{(0')}{\vdash} (q, \varepsilon, \varepsilon)$$

  von M und umgekehrt. $\triangleleft$

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

Idee:

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

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

- Ein Wort x soll also genau dann in G aus X_{pAq} ableitbar sein, wenn M ausgehend vom Zustand p bei Lesen von x in den Zustand q gelangen kann und dabei das Zeichen A aus dem Keller entfernt.
- Hierzu fügen wir für jede Anweisung $puA \to p_1 A_1 \dots A_k$, $k \geq 0$, die folgenden Regeln zu P hinzu:

  Für jede Zustandsfolge $p_2, \dots, p_{k+1}$: $X_{pAp_{k+1}} \to u X_{p_1 A_1 p_2} \dots X_{p_k A_k p_{k+1}}$

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

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

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

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

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

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

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

Beispiel (Fortsetzung)

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

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

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

Beispiel (Schluss)

- Die Anweisungen

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

von M führen also auf die folgenden Regeln von G:

$$S \to X_{p\#p}, X_{p\#q} \quad (0, 0') \qquad X_{p\#q} \to \varepsilon \quad (1')$$
$$X_{p\#p} \to aX_{pAp} \quad (2') \qquad X_{p\#q} \to aX_{pAq} \quad (2'')$$
$$X_{pAp} \to aX_{pAp}X_{pAp} \quad (3') \qquad X_{pAq} \to aX_{pAp}X_{pAq} \quad (3'')$$
$$X_{pAp} \to aX_{pAq}X_{qAp} \quad (3''') \qquad X_{pAq} \to aX_{pAq}X_{qAq} \quad (3'''')$$
$$X_{pAq} \to b \quad (4') \qquad X_{qAq} \to b \quad (5')$$

- 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

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

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

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

  $$puA \to p_1 A_1 \dots A_k, \; k \geq 0$$

  von M und jede Zustandsfolge $p_2, \dots, p_{k+1}$ die folgende Regel enthält:

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

- Dann lässt sich mit Hilfe der Äquivalenz

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

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

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