3.6 Analyse kontextfreier Sprachen

(vgl. Kapitel 2.6 über die Analyse regulärer Sprachen)
Wir diskutieren Verfahren, die die folgenden Fragestellungen bzw. Probleme für kontextfreie Sprachen entscheiden. Dabei nehmen wir an, daß kontextfreie Sprachen als kontextfreie Grammatiken gegeben sind.

- *Wortproblem:* Gilt $w \in L$ für eine gegebene kontextfreie Sprache L und $w \in \Sigma^*$?
- *Leerheitsproblem:* Gilt $L = \emptyset$ für eine gegebene kontextfreie Sprache L?
- *Endlichkeitsproblem:* Ist eine gegebene kontextfreie Sprache L endlich?

Bemerkung. Wir haben auch Verfahren kennengelernt, die die folgenden Problem lösen:

- *Disjunktheitsproblem:* Gilt $L_1 \cap L_2 = \emptyset$ für gegebene *reguläre* L_1, L_2?
- *Inklusionsproblem:* Gilt $L_1 \subseteq L_2$ für gegebene *reguläre* L_1, L_2?
- *Äquivalenzproblem:* Gilt $L_1 = L_2$ für gegebene *reguläre* L_1, L_2?

Die entsprechenden Problem für kontextfreie Sprachen sind nicht algorithmisch lösbar.

Wortproblem

Eingabe: kontextfreie Grammatik G und $w \in \Sigma^*$
Frage: $w \in L(G)$?

Verfahren:
Wandle G in Chomsky-Normalform um.
Wende CYK-Algorithmus an.

Zeitbedarf: polynomiell in G und w

Leerheitsproblem

Eingabe: kontextfreie Grammatik G
Frage: $L(G) = \emptyset$?

1. Verfahren:
Wandle G in Chomsky-Normalform um (Ergebnis: G' mit Nichtterminalmenge N)
Teste für alle Wörter z mit $|z| \leq 2^{|N|}$, ob sie von G' erzeugt werden
Wird ein solches gefunden, so gib „$L(G) \neq \emptyset$“ aus, sonst „$L(G) = \emptyset$“.

Korrektheit:
Liefert dieses Verfahren „$L(G) \neq \emptyset$“, so wurde $z \in L(G') = L(G)$ gefunden, d.h. $L(G) \neq \emptyset$.

Sei umgekehrt $L(G) \neq \emptyset$. Zu zeigen ist, daß $L(G)$ ein „kurzes“ Wort enthält. Sei z ein kürzestes Wort aus $L(G)$. Ist $|z| \geq 2^{|N|} + 1$, so existieren nach dem Pumpinglemma für kontextfreie Sprachen $u, v, w, x, y \in \Sigma^*$ mit (insbes.) $z = uvwxy$, $|vx| > 0$ und $uwy \in L(G)$. Es existiert also ein Wort $uwy \in L(G)$ mit $|uwy| < |z|$, im Widerspruch zur Wahl von z. Also existiert ein Wort der Länge $\leq 2^{|N|}$ in $L(G)$, d.h. das Verfahren liefert in diesem Fall „$L(G) \neq \emptyset$“.

Zeitbedarf: doppelt exponentiell in G

2. Verfahren:

Wandle G in Chomsky-Normalform um (Ergebnis: $G' = (N, \Sigma, S, P)$)
$W_0 := \{A \in N \mid \exists w \in \Sigma^* \colon (A \to w) \in P\}$
for $i = 0$ **to** $|N|$ **do**
  $W_{i+1} := W_i \cup \{A \in N \mid \exists w \in (\Sigma \cup W_i)^* \colon (A \to w) \in P\}$
endfor
if $S \in W_{|N|+1}$ **then** return „$L(G) \neq \emptyset$“ **else** return „$L(G) = \emptyset$“

Zeitbedarf: polynomiell in G

Beispiel:

Sei G die kontextfreie Grammatik (in Chomsky-Normalform) mit den folgenden Produktionen:

$$S \to AC \qquad A \to BC$$
$$B \to CA \mid b \qquad C \to DA \mid a$$

Wir haben

1. $W_0 = \{B, C\}$,
2. $W_1 = \{A, B, C\}$ und
3. $W_2 = W_3 = W_4 = W_5 = \{S, A, B, C\}$.

Also gibt der Algorithmus „$L(G) \neq \emptyset$“ aus.

dies ist korrekt, denn:

1. $B \Longrightarrow b$ und $C \Longrightarrow a$
2. $A \Longrightarrow BC \Longrightarrow^* ba$
3. $S \Longrightarrow AC \Longrightarrow^* baa$

Endlichkeitsproblem

Eingabe: kontextfreie Grammatik G
Frage: $L(G)$ endlich?

1. Verfahren:
Wandle G in Chomsky-Normalform um (Ergebnis: G' mit Nichtterminalmenge N)
Teste für alle Wörter w mit $2^{|N|} + 1 \leq |w| < 2 \cdot (2^{|N|} + 1)$, ob sie von G' erzeugt werden
Wird ein solches gefunden, so gib „$L(G)$ unendlich“ aus, sonst „$L(G)$ endlich“.

Korrektheit:
Liefert dieses Verfahren „$L(G)$ unendlich“, so wurde ein Wort $z \in L(G')$ gefunden mit $|z| \geq 2^{|N|} + 1$. Nach dem Pumpinglemma existieren Wörter u, v, w, x, y mit (insbes.) $z = uvwxy$, $|vx| > 0$ und $uv^n wx^n y \in L(G)$ f.a. $n \in \mathbb{N}$. Also ist $L(G') = L(G)$ tatsächlich unendlich.

Sei umgekehrt $L(G)$ unendlich. Dann existiert $z \in L(G)$ mit $|z| \geq 2^{|N|} + 1$. Sei z ein kürzestes Wort aus $L(G)$ mit $|z| \geq 2^{|N|} + 1$.
Angenommen, $|z| \geq 2 \cdot (2^{|N|} + 1)$. Nach dem Pumpinglemma für kontextfreie Sprachen existieren dann $u, v, w, x, y \in \Sigma^*$ mit (insbes.) $z = uvwxy$, $0 < |vx| \leq 2^{|N|} + 1$ und

$uwy \in L$. Also

$$2^{|N|} + 1 = 2 \cdot (2^{|N|} + 1) - (2^{|N|} + 1) \leq |z| - |vx| < |z|\,,$$

Wegen $|uwy| = |z| - |vx|$ gilt also

$$uwy \in L(G) \text{ und } 2^{|N|} + 1 \leq |uwy| < |z|\,,$$

im Widerspruch zur Wahl von z. Also gilt $|z| < 2 \cdot (2^{|N|} + 1)$.
Also liefert das Verfahren in diesem Fall tatsächlich „$L(G)$ unendlich“.

Zeitbedarf: doppelt exponentiell in G

2. Verfahren:

1. Wandle G in Chomsky-Normalform um (Ergebnis: $G' = (N, \Sigma, S, P)$)
2. Sei $W = W_{|N|+1} \subseteq N$ die im 2. Verfahren für das Leerheitsproblem berechnete Menge
3. Für $A, B \in W$ setze $(A, B) \in E$ falls es $u, w \in (W \cup \Sigma)^*$ gibt, so daß $A \to uBw$ Regel von G ist.
4. Teste, ob es $A \in W$ gibt, so daß der gerichtete Graph (W, E)
   - Pfad von S nach A enthält und
   - A auf einem Kreis liegt.
5. Wenn dies der Fall ist, so gibt „$L(G)$ ist unendlich“ aus, sonst „$L(G)$ ist endlich“.

Zeitbedarf: polynomiell in G

Beispiel:

G ist die kontextfreie Grammatik in Chomsky-Normalform mit den folgenden Produktionen:

$$\begin{array}{ll} S \to AC & A \to BC \\ B \to CA \mid b & C \to DA \mid a \end{array}$$

Dann gilt $W = \{A, B, C, S\}$ und die Kantenrelation E ist

$$E = \{(S, A), (S, C), (A, B), (A, C), (B, C), (B, A)\}.$$

Damit gilt $S \overset{E}{\to} A \overset{E}{\to} B \overset{E}{\to} A$.

Also gibt das Verfahren „$L(G)$ unendlich“ aus.

dies ist korrekt, denn:

$$S \Longrightarrow AC \Longrightarrow BCC \Longrightarrow CACC \Longrightarrow^* C^n AC^n C \Longrightarrow^* a^n\, ba\, a^n\, a$$

gilt für alle $n \geq 1$.

3.7 Eine Anwendung: Verifikation

Im Kapitel 2.7 haben wir gesehen, wie die formalen Sprachen für die Verifikation eingesetzt werden können.

Die **Idee** war, das zu verifizierende System und die unerwünschte Eigenschaft als Sprachen L_{Sys} bzw. L_{Spec} zu modellieren und dann zu testen, ob $L_{Sys} \cap L_{Spec} = \emptyset$ gilt.

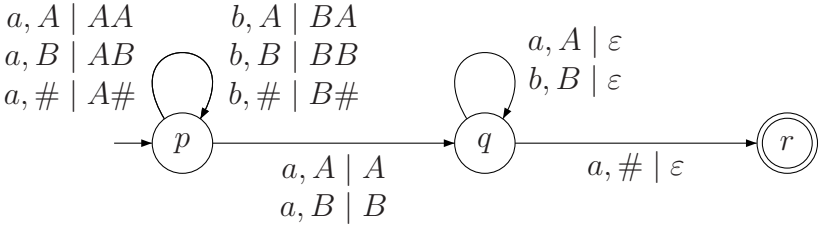
Sind diese Sprachen beide regulär, so konnten wir diesen Test algorithmisch durchführen (Stichworte: $L_{Sys} \cap L_{Spec}$ ist regulär, Leerheitsproblem für reguläre Sprachen kann gelöst werden).
Ist das System aber rekursiv (d.h. gibt es darin Routinen, die sich selber aufrufen), so ist L_{Sys} i.a. nicht regulär, sondern kontextfrei. Das Disjunktheitsproblem für kontextfreie Sprachen ist algorithmisch nicht lösbar.
Aber die Sprache L_{Spec} ist oft weiterhin regulär.

Frage: Können wir das Disjunktheitsproblem wenigstens für eine kontextfreie und eine reguläre Sprache algorithmisch lösen?

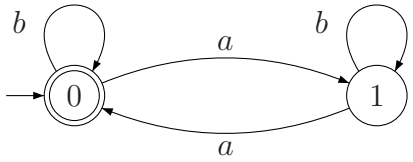
Lemma 3.10. *Aus einem Kellerautomaten M_1 und einem DFA M_2 kann ein Kellerautomat $M_\cap$ berechnet werden mit $L(M_\cap) = L(M_1) \cap L(M_2)$.*

Beweis durch Beispiel:

PDA M_1:

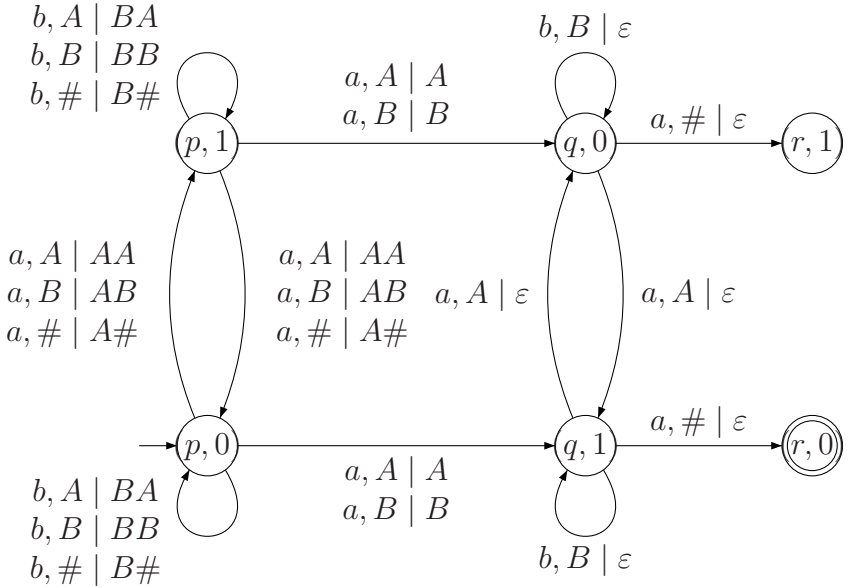


DFA M_2:



Idee: Lasse M_1 und M_2 parallel laufen (vgl. Lemma 2.5, d.h. Kreuzproduktkonstruktion für NFAs)

PDA $M_\cap$:



Allgemein kann $M_\cap = (Q, \Sigma, \Gamma, \iota, \#, \delta, F)$ aus $M_1 = (Q_1, \Sigma, \Gamma, \iota_1, \#, \delta_1, F_1)$ und $M_2 = (Q_2, \Sigma, \iota_2, \delta_2, F_2)$ wie folgt bestimmt werden:

$$\begin{aligned} Q &= Q_1 \times Q_2 \\ \iota &= (\iota_1, \iota_2) \\ \delta((p_1, p_2), a, A) &= \delta(p_1, a, A) \times \{\delta_2(p_2, a)\} \\ \delta((p_1, p_2), \varepsilon, A) &= \delta(p_1, \varepsilon, A) \times \{p_2\} \\ F &= F_1 \times F_2 \end{aligned}$$

für alle $p_1 \in Q_1$, $p_2 \in Q_2$, $a \in \Sigma$ und $A \in \Gamma$. □

Disjunktheitsproblem für PDA und NFA

Eingabe: PDA M_1 und NFA M_2 **Frage:** Gilt $L(M_1) \cap L(M_2) = \emptyset$?

Verfahren:

0. Berechne DFA M_2' mit $L(M_2) = L(M_2')$.
1. Berechne PDA $M_\cap$ mit $L(M_\cap) = L(M_1) \cap L(M_2)$.
2. Berechne kontextfreie Grammatik G mit $L(G) = L(M_\cap)$.
3. Teste, ob $L(G) = \emptyset$.

Bemerkung. 1. Idee von Lemma 3.10 erlaubt es auch, polynomiell großen PDA $M_\cap$ aus PDA M_1 und NFA M_2 zu bestimmen. Damit wird Schritt 0 unnötig und $M_\cap$ ist polynomiell groß

2. kontextfreie Grammatik G kann in polynomieller Zeit bestimmt werden („Tripel-Konstruktion“)

Damit kann Disjunktheitsproblem für PDA und NFA in polynomieller Zeit gelöst werden. Dies erlaubt es, die Verifikation wie im Kapitel 2.7 angedeutet, auf rekursive Systeme zu erweitern.

Zusammenfassung kontextfreie Sprachen

- äquivalente Beschreibungsformen: kontextfreie Grammatiken (in Chomsky-Normalform), PDAs
- Abschlußeigenschaften: Vereinigung, Produkt, Iteration
- Nicht-Abgeschlossenheit unter Schnitt und Komplement (aber Schnitt von kontextfreier und regulärer Sprache ist kontextfrei!)
- Nicht-Kontextfreiheitsbeweise: Pumpinglemma für kontextfreie Sprachen
- Algorithmen für Wort-, Leerheits- und Endlichkeitsproblem (und Disjunktheitsproblem für PDA und NFA)
- Anwendung in Verifikation