3.4 Der CYK-Algorithmus

Ziel: effizienter Algorithmus, mit dessen Hilfe bei Eingabe eines Wortes w entschieden werden kann, ob es von der kontextfreien Grammatik G erzeugt wird.
1. Idee: probiere aus, eine Ableitung zu konstruieren, die mit S beginnt und mit w endet

1. Aber dann ist nicht klar, wann man aufgeben soll. (Aufgrund der leeren rechten Seiten kann es sehr lange Ableitungen mit sehr kurzem Ergebniswort geben).

   Bsp.: $S \to A_1A_1$, $A_i \to A_{i+1}A_{i+1}$ für alle $1 \leq i < n$, $A_n \to \varepsilon$.
   Dann gilt $S \Rightarrow^* \varepsilon$, aber die einzige Ableitung hat die Länge $2^{n+1} - 1$.

2. selbst wenn es keine leeren rechten Seiten gibt, gibt es noch immer exponentiell viele auszuprobierende Ableitungen.

   Bsp.: $S \to A_1 \mid B_1$, $A_i \to A_{i+1} \mid B_{i+1}$ und $B_i \to A_{i+1} \mid B_{i+1}$ für alle $1 \leq i < n$, $A_n \to a$, $B_n \to a$.
   Es gibt 2^n viele Ableitungen, die alle die Länge $n + 1$ haben.

2. Idee: versuche, die Teilwörter systematisch zu erzeugen.

Sei $G = (N, \Sigma, S, P)$ eine kontextfreie Grammatik und $w = a_1 a_2 \cdots a_n \in \Sigma^+$.

Für $1 \leq \ell \leq n$ und $1 \leq i \leq n - \ell + 1$ sei

$$U_{i,\ell} = \{X \in N \mid X \Rightarrow_G^* a_i a_{i+1} \cdots a_{i+\ell-1}\}$$

die Menge der Nichtterminale, aus denen das Teilwort der Länge ℓ, das an der Stelle i beginnt, abgeleitet werden kann.

Beispiel 3.4. $w = aabcbc$ und G hat die folgenden Regeln:

$S \to AD \mid FG$	$A \to a$	$B \to b$	$C \to c$
$D \to SE \mid BC$	$E \to BC$	$F \to AF \mid a$	$G \to BG \mid CG \mid b$

Dann gelten

- $U_{11} = U_{21} = \{A, F\}$, $U_{31} = U_{51} = \{B, G\}$, $U_{41} = U_{61} = \{C\}$
- $U_{12} = \{F\}$, $U_{22} = \{S\}$, $U_{32} = \{D, E\}$, $U_{42} = \{G\}$, $U_{52} = \{D, E\}$
- $\cdots$

Es gilt offensichtlich $w \in L(G) \iff S \in U_{1,n}$, d.h. wir müssen die Menge $U_{1,n}$ berechnen.

Hierzu werden wir zunächst alle Mengen $U_{i,1}$ berechnen, dann alle Menge $U_{i,2}$ usw.

Bei der Berechnung von $U_{i,\ell}$ müssen wir also für jedes $X \in N$ herausfinden, ob $X \in U_{i,\ell}$ gilt. Dazu werden wir *nicht* versuchen, aus X das Wort $a_i a_{i+1} \cdots a_{i+\ell-1}$ abzuleiten. Vielmehr werden wir die Mengen $U_{j,k}$ mit $k < \ell$ verwenden. D.h. wir verwenden das Verfahren der *dynamischen Programmierung*.

Dies funktioniert einfacher, wenn wir eine kontextfreie Grammatik in Chomsky-Normalform verwenden (was nach Satz 3.2 ja immer möglich ist).

Beobachtung. Sei $G = (N, \Sigma, S, P)$ kontextfreie Grammatik in Chomsky-Normalform, $X \in N$ und $w = a_1 a_2 \cdots a_n \in \Sigma^+$

Dann gilt $X \in U_{1,n}$ (d.h. $X \Rightarrow_G^* w$) genau dann, wenn

- $n = 1$ und $X \to w \in P$ oder
- $n > 1$ und es existieren $1 \leq k < n$ und $X \to YZ \in P$ mit $Y \Rightarrow_G^* a_1 a_2 \cdots a_k$ und $Z \Rightarrow_G^* a_{k+1} \cdots a_n$, d.h. $Y \in U_{1,k}$ und $Z \in U_{k+1,n-(k+1)}$

Allgemeiner gilt $X \in U_{i,\ell}$ genau dann, wenn

- $\ell = 1$ und $X \to a_i \in P$ oder
- $\ell > 1$ und es gibt $X \to YZ \in P$ und $1 \leq k < \ell$ mit $Y \Rightarrow_G^* a_i a_{i+1} \cdots a_{i+k}$ und $Y \Rightarrow_G^* a_{i+k+1} a_{i+k+2} \cdots a_{i+\ell}$, d.h. $Y \in U_{i,k}$ und $Z \in U_{i+k+1,\ell-(k+1)}$

Damit kann $U_{i,\ell}$ leicht aus den Mengen $U_{j,k}$ mit $k < \ell$ berechnet werden.

Damit erhalten wir den **CYK-Algorithmus**:

Eingabe: kontextfreie Grammatik $G = (N, \Sigma, S, P)$ in Chomsky-Normalform
nichtleeres Wort $w = a_1 a_2 \cdots a_n \in \Sigma^+$

Ausgabe: „ja“, falls $w \in L(G)$, „nein“ sonst

(1) für alle $1 \leq i \leq n$ setze $U_{i,1} = \{X \in N \mid X \to a_i \in P\}$

(2) für alle $2 \leq \ell \leq n$

(3) für alle $1 \leq i \leq n - \ell + 1$

(4) $U_{i,\ell} = \{X \in N \mid \exists X \to YZ \in P, 1 \leq k < \ell \text{ mit } Y \in U_{i,k} \text{ und } Z \in U_{k+1,\ell-(k+1)}\}$

(5) falls $S \in U_{1,n}$, so gib „ja“ aus, sonst „nein“

Bemerkung. Der CYK-Algorithmus muß $O(n^2)$ viele Mengen $U_{i,\ell}$ berechnen.
Die Berechnung von $U_{i,\ell}$ betrachtet $O(|G|)$ viele Regeln $X \to YZ$ und $O(n)$ viele Zahlen k.

Der Algorithmus läuft also in Zeit $O(|G| \cdot |w|^3)$.

Beispiel 3.4 (Fortsetzung) $w = aabcbc$ und G hat die folgenden Regeln:

$S \to AD \mid FG$ $\quad A \to a$ $\quad B \to b$ $\quad C \to c$

$D \to SE \mid BC$ $\quad E \to BC$ $\quad F \to AF \mid a$ $\quad G \to BG \mid CG \mid b$

	a	a	b	c	b	c
$\ell = 1$	$U_{1,1}$	$U_{2,1}$	$U_{3,1}$	$U_{4,1}$	$U_{6,1}$	$U_{6,1}$
$\ell = 2$	$U_{1,2}$	$U_{2,2}$	$U_{3,2}$	$U_{4,2}$	$U_{5,2}$	
$\ell = 3$	$U_{1,3}$	$U_{2,3}$	$U_{3,3}$	$U_{4,3}$		
$\ell = 4$	$U_{1,4}$	$U_{2,4}$	$U_{3,4}$			
$\ell = 5$	$U_{1,5}$	$U_{2,5}$				
$\ell = 6$	$U_{1,6}$					

a	a	b	c	b	c

a	a	b	c	b	c
A, F	A, F	B, G	C	B, G	C
F	S	D, E	G	D, E	
S	S	G			
	S				
S	D				
S					

Wegen $S \in U_{1,6}$ gilt also $w \in L(G)$.

3.5 Kellerautomaten

1. Frage: Wir haben gesehen, daß kontextfreie Sprachen nicht unbedingt durch NFAs akzeptiert werden können (Bsp.: $\{a^m b^m \mid m \in \mathbb{N}\}$ ist kontextfrei, aber nicht regulär).
Wie müssen wir die NFAs erweitern, um alle kontextfreien Sprachen (aber auch nicht mehr) akzeptieren zu können?

2. Frage: Wir haben gesehen, daß mittels CYK-Algorithmus für jede kontextfreie Grammatik G und jedes Wort w in polynomieller Zeit entschieden werden kann, ob $w \in L(G)$ liegt. Im positiven Fall wird in derselben Zeit auch ein Ableitungsbaum konstruiert (was für Compiler sehr hilfreich ist).
Für eine feste kontextfreie Grammatik in Chomsky-NF benötigt der Algorithmus $O(|w|^3)$ viele Schritte.
Kann man (zumindest für gewissen kontextfreie Sprachen) schnellere Algorithmen angeben?

Definition. 1. Ein *Kellerautomat (mit Endzuständen)* (PDA) ist ein Tupel $M = (Q, \Sigma, \Gamma, \iota, \#, \delta, F)$, wobei

- Q eine endliche Menge von „Zuständen“,
- Σ und Γ Alphabete („Eingabe-“ bzw. „Kelleralphabet“),
- $\iota \in Q$ der „Initialzustand“,
- $\# \in \Gamma$ das „Kellerstartsymbol“,
- $\delta\colon Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ mit $\delta(q, x, A)$ endlich f.a. $q \in Q$, $x \in \Sigma \cup \{\varepsilon\}$ und $A \in \Gamma$ die „Überführungsfunktion“ und
- $F \subseteq Q$ die Menge der „Endzustände“ sind.

2. Eine *Konfiguration von M* ist ein Tupel $(q, w, \gamma) \in Q \times \Sigma^* \times \Gamma^*$, wobei q der „aktuelle Zustand“, w das „noch zu lesende Wort“ und γ der „Kellerinhalt“ ist.

3. Für $q, q' \in Q$, $w, w' \in \Sigma^*$, $A \in \Gamma$ und $\gamma, \gamma' \in \Gamma^*$ setzen wir

$$(q, w, A\gamma) \vdash (q', w', \gamma'\gamma)\,,$$

falls es $a \in \Sigma \cup \{\varepsilon\}$ gibt mit

$$w = aw' \text{ und } (q', \gamma') \in \delta(q, a, A)\,.$$

4. Für Konfigurationen K und K' schreiben wir

$$K \vdash^* K' ,$$

wenn es $n \in \mathbb{N}$ und Konfigurationen $K_0, K_1, \cdots, K_n$ gibt mit

$$K = K_0 \vdash K_1 \vdash \cdots \vdash K_n = K' .$$

5. Der PDA M *akzeptiert* das Wort $w \in \Sigma^*$, wenn es $f \in F$ und $\gamma \in \Gamma^*$ gibt mit

$$(\iota, w, \#) \vdash^* (f, \varepsilon, \gamma) .$$

Sei $L(M) \subseteq \Sigma^*$ die Menge der von M akzeptierten Wörter.

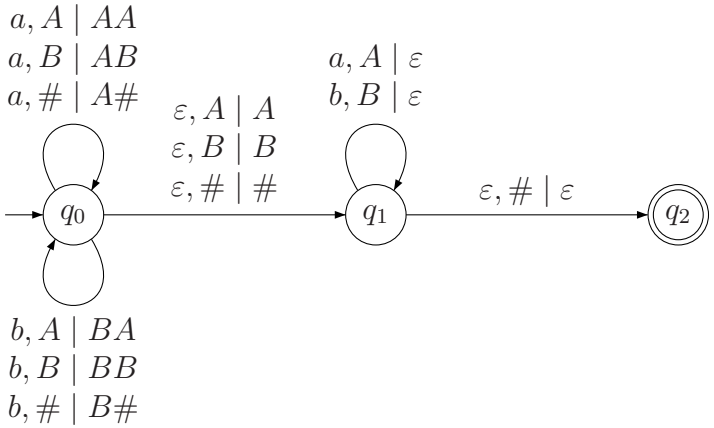
Beispiel. Sei $\Sigma = \{a, b\}$ und $M = (Q, \Sigma, \Gamma, \iota, \#, \delta, F)$ der PDA mit

- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{A, B, \#\}$
- $\iota = q_0$,
- $F = \{q_2\}$

$\delta(q_0, ., .)$	a	b	ε
A	$\{(q_0, AA)\}$	$\{(q_0, BA)\}$	$\{(q_1, A)\}$
B	$\{(q_0, AB)\}$	$\{(q_0, BB)\}$	$\{(q_1, B)\}$
$\#$	$\{(q_0, A\#)\}$	$\{(q_0, B\#)\}$	$\{(q_1, \#)\}$

$\delta(q_1, ., .)$	a	b	ε
A	$\{(q_1, \varepsilon)\}$	$\emptyset$	$\emptyset$
B	$\emptyset$	$\{(q_1, \varepsilon)\}$	$\emptyset$
$\#$	$\emptyset$	$\emptyset$	$\{(q_2, \varepsilon)\}$

$\delta(q_2, x, X) = \emptyset$ für alle $x \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$



Dann gilt

$$(q_0, abba, \#) \vdash (q_0, bba, A\#) \vdash (q_1, bba, A\#)$$
$$(q_0, abba, \#) \vdash (q_0, bba, A\#) \vdash (q_0, ba, BA\#) \vdash (q_1, ba, BA\#)$$
$$\vdash (q_1, a, A\#) \vdash (q_1, \varepsilon, \#) \vdash (q_2, \varepsilon, \varepsilon)$$

Also haben wir $abba \in L(M)$ und allgemeiner $L(M) = \{ww^R \mid w \in \Sigma^*\}$

Lemma 3.5. *Aus einer kontextfreien Grammatik G kann ein Kellerautomat M berechnet werden mit $L(G) = L(M)$.*

Konstruktion. Idee:

- erzeuge (nichtdet.) ein Wort von $L(G)$ auf dem Keller und vergleiche es mit der Eingabe
- da PDAs nur Zugriff auf das oberste Kellersymbol haben, muß Erzeugung und Vergleich im Reißverschlußverfahren erfolgen.

Sei $G = (N, \Sigma, S, P)$.
Konstruiere $M = (Q, \Sigma, \Gamma, \iota, \#, \delta, F)$ mit

- $Q = \{\iota, f\}$,
- $\Gamma = N \cup \Sigma \cup \{\#, \bot\}$,
- $F = \{f\}$,

- $\delta(\iota, \varepsilon, \#) = \{(\iota, S\bot)\}$ (Initialisierung)

  $\delta(\iota, \varepsilon, A) = \{(\iota, \gamma) \mid (A \to \gamma) \in P\}$ (steht $A \in N$ auf dem Keller, so expandiere es gemäß einer Regel)

  $\delta(\iota, a, a) = \{(\iota, \varepsilon)\}$ (steht $a \in \Sigma$ auf dem Keller, so vergleiche es mit nächstem Eingabesymbol)

  $\delta(\iota, \varepsilon, \bot) = \{(f, \varepsilon)\}$ (Abschluß)

  $\delta(\iota, \varepsilon, a) = \delta(\iota, a, b) = \delta(\iota, a, B) = \delta(f, x, X) = \emptyset$ f.a. $a, b \in \Sigma$ mit $a \neq b$, $A \in V$, $x \in \Sigma \cup \{\varepsilon\}$, $X \in \Gamma$

□

Beispiel. Wir betrachten die kontextfreie Grammatik $G = (N, \Sigma, S, P)$ mit $N = \{S\}$ und $\Sigma = \{0, 1\}$, wobei P die folgenden Regeln enthält:

$$S \to 0S1S \qquad S \to 01S$$
$$S \to 0S1 \qquad S \to 01$$

PDA-Berechnung	Ableitung der Grammatik	
$(\iota, 001011, \#)$		
$\vdash (\iota, 001011, S\bot)$	S	
$\vdash (\iota, 001011, 0S1\bot)$	$\Rightarrow_G 0S1$	
$\vdash (\iota, 01011, S1\bot)$		
$\vdash (\iota, 01011, 01S1\bot)$	$\Rightarrow_G 001S1$	
$\vdash (\iota, 1011, 1S1\bot)$		Die Expansionsschritte des
$\vdash (\iota, 011, S1\bot)$		PDA entsprechen also den
$\vdash (\iota, 011, 011\bot)$	$\Rightarrow_G 001011$	Ableitungsschritten der
$\vdash (\iota, 11, 11\bot)$		kontextfreie Grammatik.
$\vdash (\iota, 1, 1\bot)$		
$\vdash (\iota, \varepsilon, \bot)$		
$\vdash (f, \varepsilon, \varepsilon)$		

Bemerkung. Angenommen, alle rechten Seiten von G gehören zu ΣN^*. Dann besteht jede Berechnung des PDA aus abwechselnden Expansions- und Vergleichstransitionen (abgesehen von Initialisierung und Abschluß). Dann können diese zusammengefaßt werden (an Stelle von z.B. $(\iota, aw, A\gamma) \vdash (\iota, aw, aBCD\gamma) \vdash (\iota, w, BCD\gamma)$ mache in einem Schritt $(\iota, aw, A\gamma) \vdash (\iota, w, BCD\gamma)$).
Zusätzlich kommt man dann mit einem Zustand und ohne Initialisierung und Abschluß aus.
die gute Nachricht: Zu jeder kontextfreien Grammatik G gibt es eine kontextfreie Grammatik G' in „Greibach-Normalform“ mit $L(G') = L(G) \setminus \{\varepsilon\}$.

Satz 3.6. *Sei $L \subseteq \Sigma^*$. Dann sind äquivalent*

(1) L ist kontextfrei.

(2) es gibt einen PDA M mit $L = L(M)$.

(3) es gibt einen PDA M mit nur einem Zustand und ohne ε-Transitionen mit $L(M) = L \setminus \{\varepsilon\}$.

Definition. Ein PDA $M = (Q, \Sigma, \Gamma, \iota, \#, \delta, F)$ ist *deterministisch*, wenn für alle $q \in Q$, $a \in \Sigma$ und $A \in \Gamma$ gilt

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

Bemerkung.

- Ist M det. PDA und $w \in \Sigma^*$, so kann in linearer Zeit entschieden werden, ob $w \in L(M)$.
- Aus einem det. PDA M kann ein det. PDA M' berechnet werden mit $L(M') = \Sigma^* \setminus L(M)$.

  Für beliebige PDAs gilt dies nicht, da z.B. das Komplement der kontextfreien Sprache $L = \{a^k b^\ell c^m \mid k = \ell \text{ oder } \ell = m\}$ nicht kontextfrei ist.

  Damit kann aber diese kontextfreie Sprache nicht von einem det. PDA akzeptiert werden (aber natürlich von einem nichtdet.).

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: $L(G) = \emptyset$?

Verfahren:

Wandle G in Chomsky-Normalform um (mgl. nach Satz 3.2, 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?

Verfahren:

1. Wandle G in Chomsky-Normalform um (mgl. nach Satz 3.2, 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.

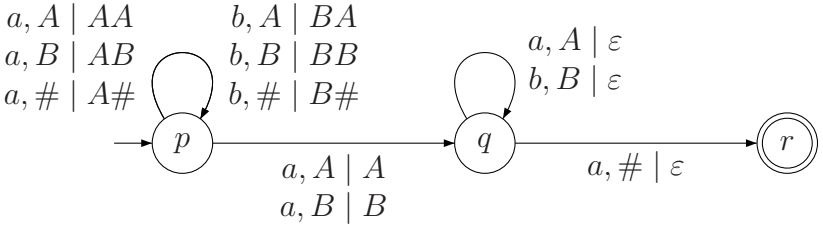
Sind diese Sprachen beide regulär, so konnten wir diesen Test algorithmisch durchführen (Stichworte: $L_{Sys} \cap L_{Spec}$ ist regulär, Leerheitsproblem für reguläre Sprachen kann gelöst werden).
Ist das System aber rekursiv (d.h. gibt es darin Routinen, die sich selber aufrufen), so ist L_{Sys} i.a.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

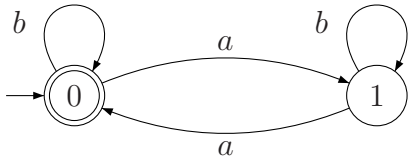
Lemma 3.7. *Aus einem Kellerautomaten M_1 und einem DFA M_2 kann ein Kellerautomat $M_\cap$ berechnet werden mit $L(M_\cap) = L(M_1) \cap L(M_2)$.*

Beweis durch Beispiel:

PDA M_1:

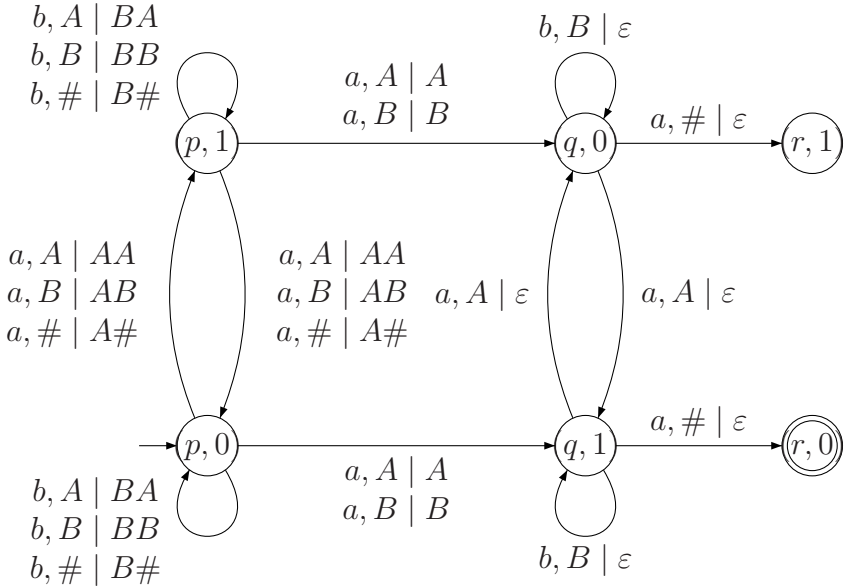


DFA M_2:



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 3.7 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