RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 1
Emmes—Kneis—Langer—Rossmanith 22.04.2009

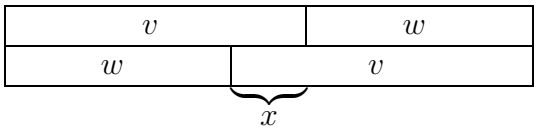
Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

Tutoraufgabe T1

Es seien $v, w \in \Sigma^*$, so daß $vw = wv$.
Beweisen Sie: Es existieren $u \in \Sigma^*$, $i, j \in \mathbf{N}_0$ mit $v = u^i$, $w = u^j$.

Lösungsvorschlag

Beweis durch Induktion über $|vw|$.



- Falls $|vw| = 0$, dann folgt die Aussage mit $u = \varepsilon$: $v = u^0$ und $w = u^0$.
- Falls $|vw| > 0$:
  - Falls $|v| = |w|$, dann gilt $v = w$ und für $u = v = w$ und $i = j = 1$ gilt nun $u^i = u^j = v = w$.
  - Falls $|v| > |w|$ ($|w| > |v|$ analog), dann existiert ein $x \in \Sigma^+$ mit $vx = w$ und $xv = w$. Da $|vx| < |vw|$ existiert durch Induktionsschluß ein $u \in \Sigma^*$, so daß $x = u^i$ und $v = u^j$. Somit gilt $w = u^{i+j}$.

Tutoraufgabe T2

Sei $w \in \Sigma^*$ ein Wort. Wenn wir das Wort w rückwärts schreiben, so nennen wir es w^R. Das ist aber keine anständige Definition.

1. Definieren Sie die Abbildung $\cdot^R$ formal.
2. Ist $w \mapsto w^R$ ein Homomorphismus?

Lösungsvorschlag

1. Bei einem freien Monoid $(\Sigma, \cdot)$ hat jedes Wort w eine eindeutige Darstellung $w = c_1 \cdot \ldots \cdot c_n$ mit $c_i \in \Sigma$. Somit läßt sich die Umkehrung definieren als $w^R := c_n \cdot \ldots \cdot c_1$.
2. Falls $|\Sigma| = 1$, dann gilt o.B.d.A. $\Sigma = \{a\}$. Somit gilt $w^R = (w_1 \ldots w_n)^R = (a \ldots a)^R = a^R \ldots a^R = w$. Allerdings, für $|\Sigma| > 1$, wähle $a, b \in \Sigma$, $a \neq b$. Dann gilt $(ab)^R = ba$, aber $a^R b^R \neq (ab)^R$. Somit ist $w \mapsto w^R$ kein Homomorphismus.

Tutoraufgabe T3
Es sei L regulär und $L^R := \{ w^R \mid w \in L \}$.
Beweisen Sie: L^R ist regulär.

Lösungsvorschlag
Falls L regulär ist, dann existiert ein regulärer Ausdruck ϕ der L erzeugt.
Wir beweisen per Induktion über den Aufbau von ϕ, daß ein regulärer Ausdruck ϕ' existiert, der L^R erzeugt. Somit ist L^R ebenfalls regulär.

- Sei $\phi = \emptyset$, dann gilt $\phi' = \emptyset$.
- Sei $\phi = \varepsilon$, dann gilt $\phi' = \varepsilon$.
- Sei $\phi = c \in \Sigma$, dann gilt $\phi' = c$.
- Sei $\phi = \phi_1 + \phi_2$, so daß ϕ_1 und ϕ_2 die regulären Sprachen L_1 bzw. L_1 erzeugen. Dann existieren nach Induktionsvorraussetzung ϕ'_1 und ϕ'_2 die die regulären Sprachen L_1^R bzw. L_2^R erzeugen. Somit erzeugt $\phi' = \phi'_1 + \phi'_2$ die Sprache L_R.
- Sei $\phi = \phi_1\phi_2$, so daß ϕ_1 und ϕ_2 die regulären Sprachen L_1 bzw. L_1 erzeugen. Dann existieren nach Induktionsvorraussetzung ϕ'_1 und ϕ'_2 die die regulären Sprachen L_1^R bzw. L_2^R erzeugen. Somit erzeugt $\phi' = \phi'_2\phi'_1$ die Sprache L_R.
- Sei $\phi = \phi_1^*$, so daß ϕ_1 die reguläre Sprachen L_1 erzeugt. Dann existiert nach Induktionsvorraussetzung der Ausdruck ϕ'_1, der die reguläre Sprache L_1^R erzeugt. Somit erzeugt $\phi'_1 = \phi'^*_1$ die Sprache L_R.

Wir haben dabei die folgenden Identitäten verwendet, die noch gesondert bewiesen werden müssen:

1. $(L_1 \cup L_2)^R = \{ w^R \mid w \in L_1 \cup L_2 \} = \{ w^R \mid w \in L_1 \} \cup \{ w^R \mid w \in L_2 \} = L_1^R \cup L_2^R$
2. $(L_1 L_2)^R = \{ (uv)^R \mid u \in L_1,\, v \in L_2 \} = \{ v^R u^R \mid u \in L_1,\, v \in L_2 \} = L_2^R L_1^R$
3. $(L_1^*)^R = \bigcup_{n\geq 0} (L_1^n)^R = \bigcup_{n\geq 0} (L_1^R)^n = (L_1^R)^*$

Dabei gilt $(uv)^R = v^R u^R$, weil

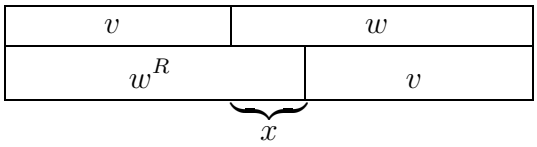
$$(uv)^R = (u_1 \dots u_n v_1 \dots v_m)^R = v_m \dots v_1 u_n \dots u_1.$$

Hausaufgabe H1 (10 Punkte)
Gegeben seien $v, w \in \Sigma^*$ mit

- $vw = w^R v$,
- $|w| \geq |v|$.

Beweisen oder widerlegen Sie, daß dann $(vw)^R = vw$ gilt.

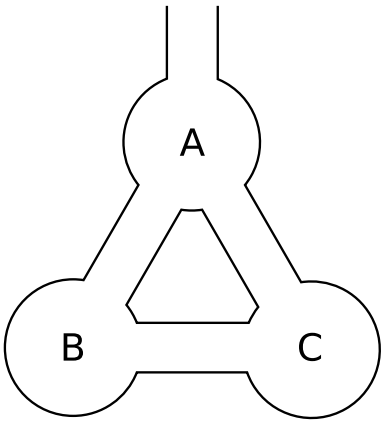
Lösungsvorschlag
Wegen $vw = w^R v$ und $|w| \geq |v|$ existiert ein $x \in \Sigma^*$ mit $w = xv$ und $w^R = vx$.



Dann gilt aber auch $xv = w = (w^R)^R = (vx)^R = x^R v^R$. Also ist insbesondere $v = v^R$. Daraus folgt dann leicht

$$vw = w^R v = w^R v^R = (vw)^R.$$

Hausaufgabe H2 (10 Punkte)
Geben Sie einen regulären Ausdruck an, der einen (nicht leeren) Pfad durch das nebenstehende Museum beschreibt. Ein Pfad startet im Raum A und endet ebenfalls dort. Beispielsweise wäre ABCABABCA ein gültiger Pfad aber ABBA oder ε nicht.



Lösungsvorschlag
Anschaulich können wir einen Weg durch das Mini-Atomium folgendermaßen gliedern: Wir kehren immer wieder in den Raum A zurück und bewegen uns dazwischen zwischen den Räumen B und C. Letzteres läßt sich durch den regulären Ausdruck

$$H = (BC)^+ + (BC)^*B + (CB)^+ + (CB)^*C$$

beschreiben. Die vier Teile entsprechen den Wegen, die

1. in B beginnen und in C enden,
2. in B beginnen und in B enden,
3. in C beginnen und in B enden und
4. in C beginnen und in C enden.

Der gesamte reguläre Ausdruck ist dann

$$(AH)^*A = \Big[A\Big((BC)^+ + (BC)^*B + (CB)^+ + (CB)^*C\Big)\Big]^*A.$$

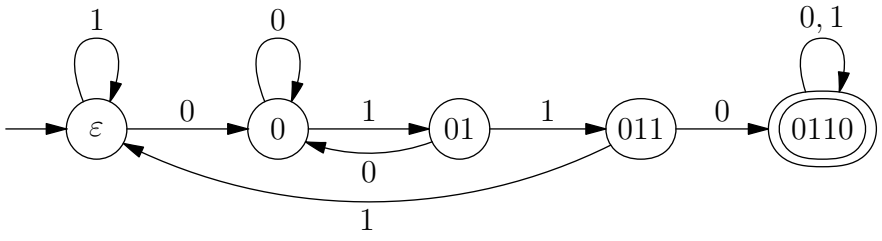
RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 2
Emmes—Kneis—Langer—Rossmanith 29.04.2009

Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

T4

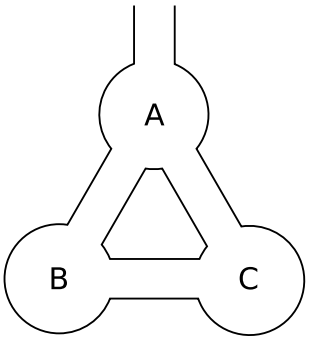
Geben Sie einen deterministischen endlichen Automaten über dem Alphabet $\Sigma = \{0, 1\}$ an, der ein Wort genau dann akzeptiert, wenn es an einer beliebigen Stelle die Zeichenfolge 0110 enthält. Der DFA erkennt also genau die Sprache $L = (0 + 1)^* \, 0110 \, (0 + 1)^*$.

Lösungsvorschlag

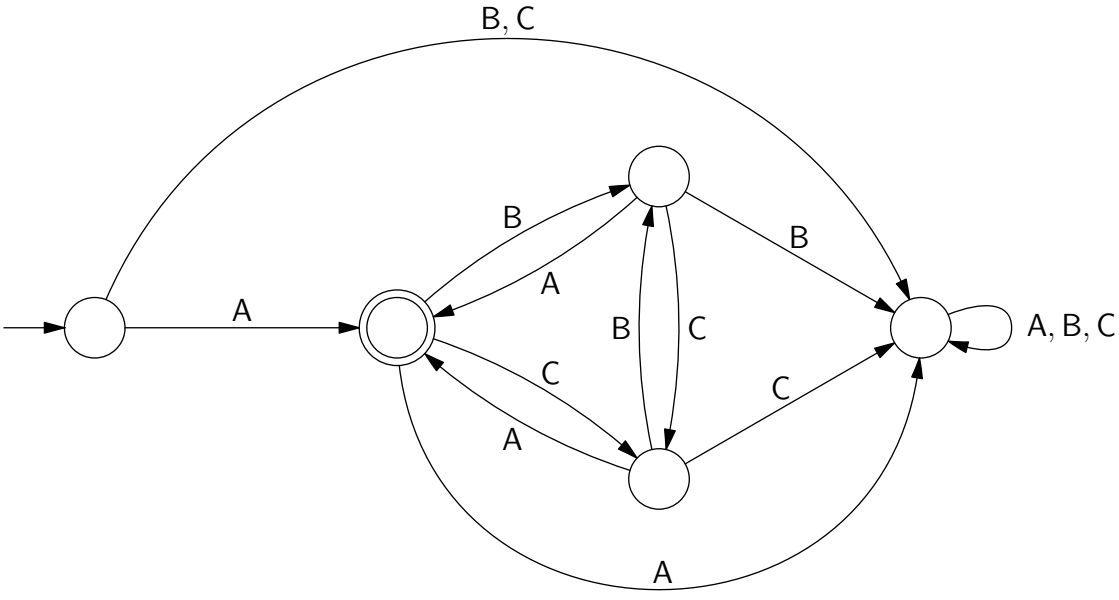


T5

Geben Sie einen deterministischen endlichen Automaten an, der alle (nicht leeren) Pfade durch das nebenstehende Museum (vergleiche Aufgabe H2) beschreibt. Ein Pfad startet im Raum A und endet ebenfalls dort. Beispielsweise akzeptiert der Automat das Wort ABCABABCA, aber die Wörter ABBA und ε nicht.



Lösungsvorschlag



T6

Wir nehmen an, daß die Sprache $L = \{ a^n b^m \mid m \geq n \geq 0 \}$ nicht regulär ist. Zeigen Sie, daß unter dieser Annahme auch die folgenden Sprachen nicht regulär sind, indem Sie Abschlußeigenschaften regulärer Sprachen verwenden.

1. $L_1 = \{ b^n a^m \mid m \geq n \geq 0 \}$
2. $L_2 = \{ a^n b^n \mid n \geq 0 \}$
3. $L_3 = \{ a^n b^n c^n \mid n \geq 0 \}$
4. $L_4 = \{ a^m b^n \mid m \geq n \geq 0 \}$
5. $L_5 = \{ a^{2n} b^n \mid n \geq 0 \}$

Lösungsvorschlag

Im folgenden sei $h\colon \Sigma \to \Sigma^*$ eine Umbenennung von Buchstaben in Wörter. Die Erweiterung von h über Wörtern $h : \Sigma^* \to \Sigma^*$, $a \in \Sigma, w \in \Sigma^*$ mit $h(\varepsilon) := \varepsilon$ und $h(aw) := h(a)h(w)$ ist dann ein Homomorphismus. Über einer Sprache L sei h definiert als $h(L) := \{ h(w) \mid w \in L \}$.

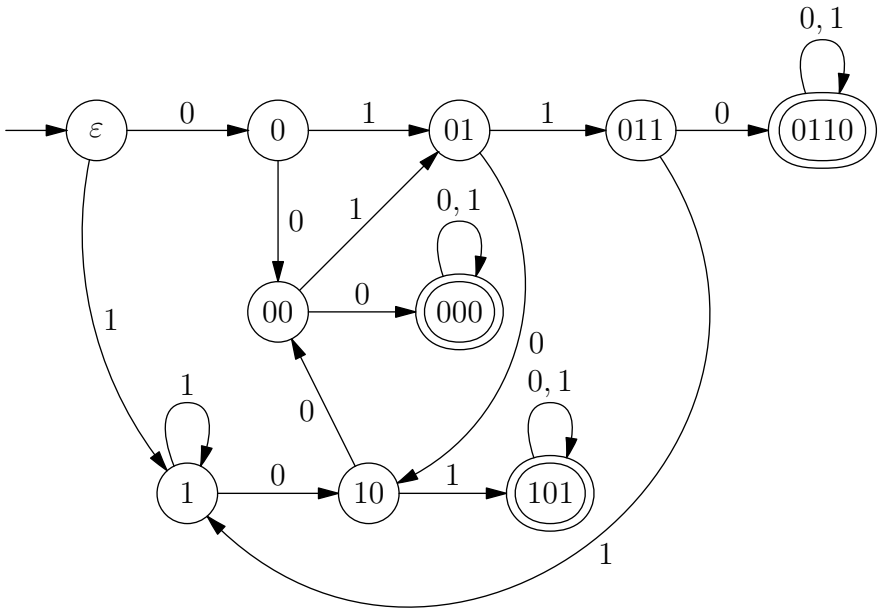
1. Angenommen L_1 sei regulär. Sei h definiert durch $h(a) := b$ und $h(b) := a$. Dann ist $h(L_1) = \{ a^n b^m \mid m \geq n \geq 0 \} = L$. Widerspruch, da reguläre Sprachen unter Homomorphismen abgeschlossen sind.
2. Angenommen L_2 sei regulär. Dann ist auch $L_2 b^* = L$ regulär, da reguläre Sprachen unter Konkatenation abgeschlossen sind. Widerspruch.
3. Sei $h(a) := a$, $h(b) := b$ und $h(c) := \varepsilon$. Dann gilt $h(L_3) = \{ a^n b^n \mid n \geq 0 \} = L_2$. Widerspruch.
4. Wie in Aufgabe **T3** gezeigt, sind reguläre Sprachen unter $\cdot^R$ abgeschlossen. Es gilt $L_4^R = \{ b^n a^m \mid m \geq n \geq 0 \} = L_1$. Widerspruch.
5. Sei $h(a) = a$ und $h(b) = bb$. Dann gilt:
   - $h(L_5) = \{ a^{2n} b^{2n} \mid n \geq 0 \}$
   - $ah(L_5)b = \{ a^{2n+1} b^{2n+1} \mid n \geq 0 \}$

   Somit ist $ah(L_5)b + h(L_5) = L_2$. Widerspruch.

H3 (10 Punkte)

Geben Sie einen deterministischen endlichen Automaten über dem Alphabet $\Sigma = \{0, 1\}$ an, der ein Wort aus Σ^* genau dann akzeptiert, wenn es eine der Zeichenketten 0110, 101 oder 000 enthält. Der DFA soll also genau $(0+1)^*(0110+101+000)(0+1)^*$ akzeptieren.

Lösungsvorschlag



H4 (10 Punkte)

Sei $L \subseteq \Sigma^*$ eine Sprache und $a \in \Sigma$. Der deterministische endliche Automat A erkenne die Sprache La. Beweisen Sie, daß dann ein DFA B existiert, der die Sprache L erkennt.

Lösungsvorschlag

Sei $M := (Q, \Sigma, \delta, q_0, F)$ ein deterministischer endlicher Automat, der die Sprache La erkennt. Sei $F' := \{\, q \in Q \mid \delta(q, a) \in F \,\}$. Dann erkennt der DFA $M' := (Q, \Sigma, \delta, q_0, F')$ die Sprache L.

Beweis: Sei $wa \in L(M) = La$. Dann gilt $\delta(q_0, wa) \in F$. Somit gilt $\delta(\delta(q_0, w), a) \in F$ und $\delta(q_0, w) \in F'$. Somit akzeptiert M' das Wort w.

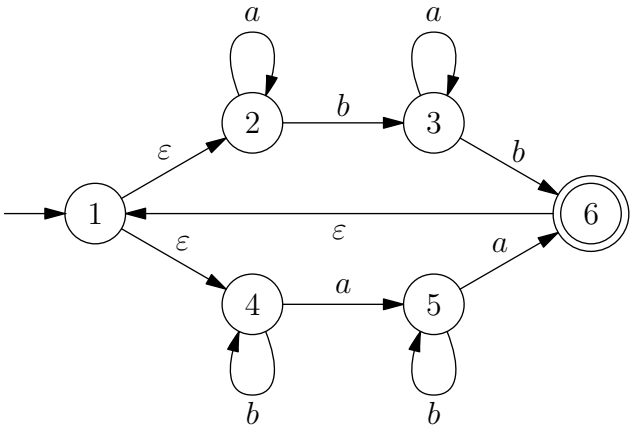
Andererseits, sei $wa \notin L(M) = La$. Dann gilt $\delta(q_0, wa) \notin F$ und analog zu oben $\delta(q_0, w) \notin F'$.

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 3
Emmes—Kneis—Langer—Rossmanith 6.5.2009

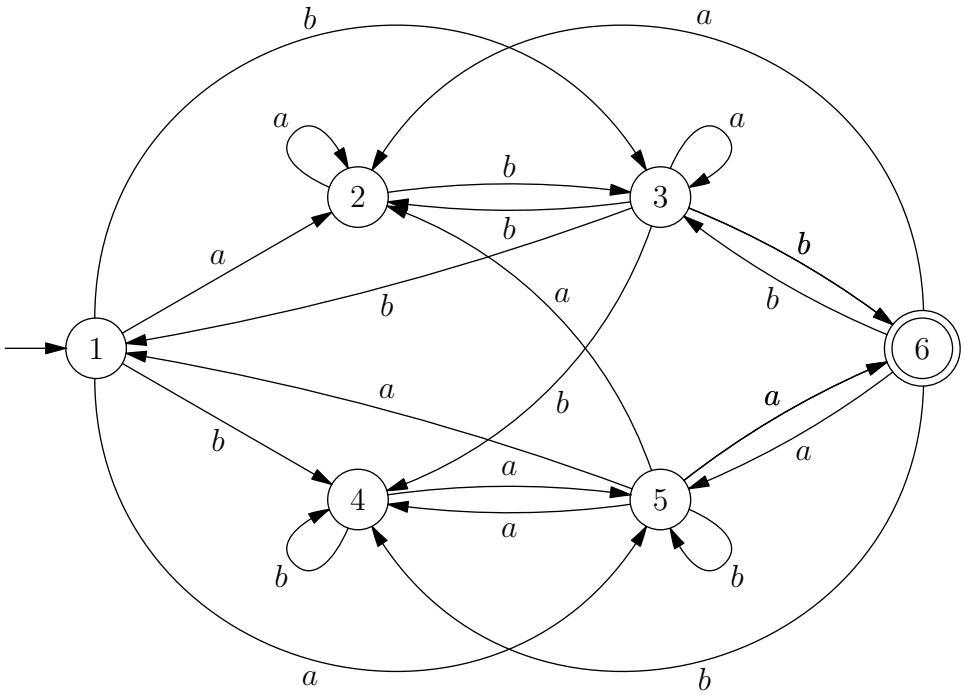
Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

T7

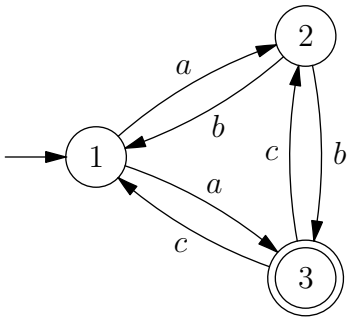
Wandeln Sie den folgenden NFA mit ϵ-Übergängen in einen NFA (ohne ϵ-Übergänge) um.



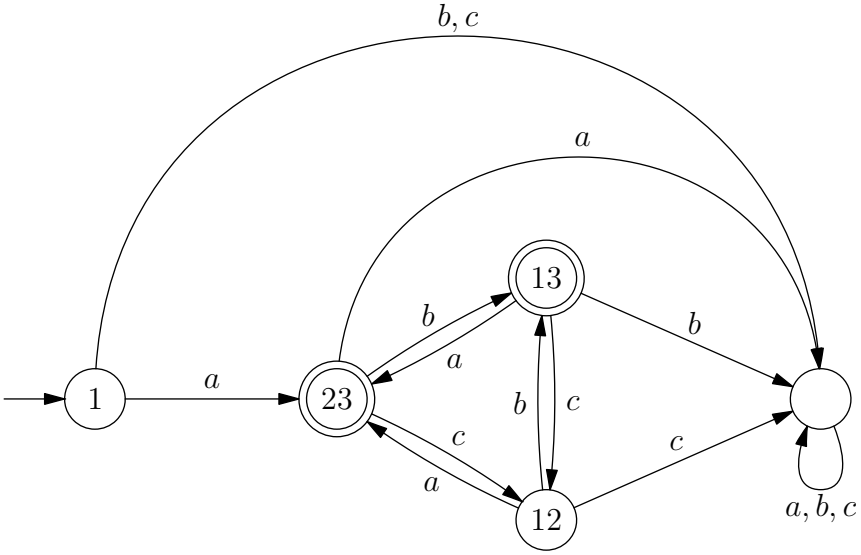
Lösungsvorschlag



T8
Führen Sie die Potenzmengenkonstruktion auf folgendem NFA durch.



Lösungsvorschlag



T9
Benutzen Sie die L_{ij}^{k}-Konstruktion, um einen regulären Ausdruck zu dem NFA aus Aufgabe T8 zu berechnen.

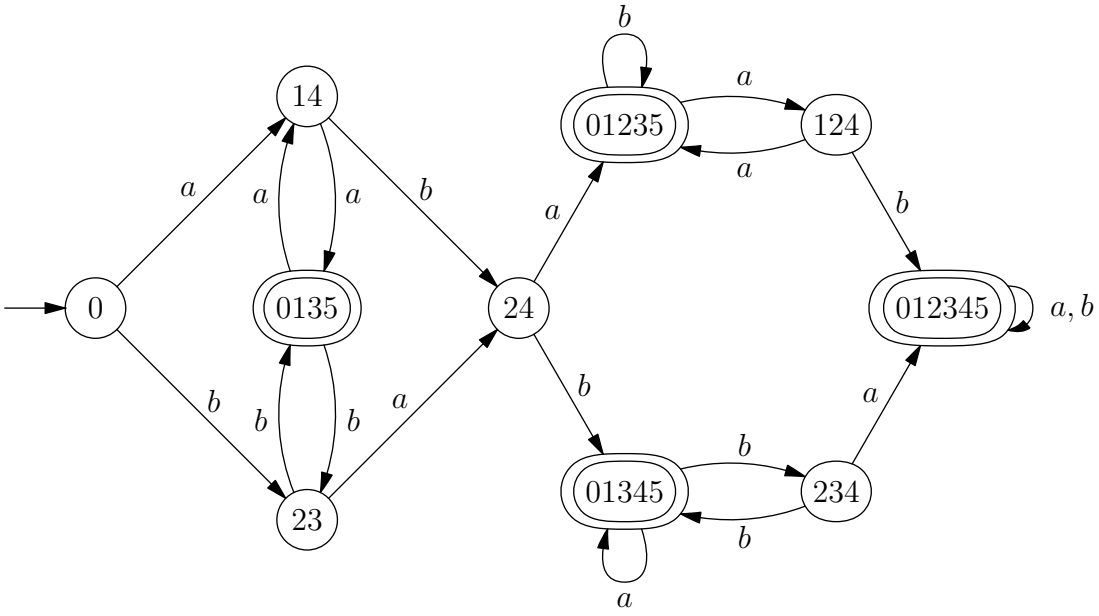
Lösungsvorschlag

$$
\begin{aligned}
L(A) &= R^3_{13} \\
\\
R^3_{13} &= R^2_{13} + R^2_{13}(R^2_{33})^* R^2_{33} \\
R^2_{13} &= R^1_{13} + R^1_{12}(R^1_{22})^* R^1_{23} \\
R^2_{33} &= R^1_{33} + R^1_{32}(R^1_{22})^* R^1_{23} \\
R^1_{12} &= a \\
R^1_{13} &= a \\
R^1_{22} &= \varepsilon + ba \\
R^1_{23} &= b + ba \\
R^1_{32} &= c + ca \\
R^1_{33} &= \varepsilon + ca \\
\\
R^2_{13} &= a + a(\varepsilon + ba)^*(b + ba) \\
R^2_{33} &= \varepsilon + ca + (c + ca)(\varepsilon + ba)^*(b + ba) \\
R^3_{13} &= a + a(\varepsilon + ba)^*(b + ba) + (a + a(\varepsilon + ba)^*(b + ba)) \\
&\quad (\varepsilon + ca + (c + ca)(\varepsilon + ba)^*(b + ba))^*(\varepsilon + ca + (c + ca)(\varepsilon + ba)^*(b + ba))
\end{aligned}
$$

H5 (10 Punkte)

Führen Sie die Potenzmengenkonstruktion auf dem NFA durch, der durch die Entfernung der ε-Übergänge des Automaten in Aufgabe T7 entstanden ist.

Lösungsvorschlag



RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 4
Emmes—Kneis—Langer—Rossmanith 13.5.2009

Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

T10

1. Betrachten Sie $L = \{ a^n b^n \mid n \geq 0 \}$. Gelten $a \equiv_L aa$, $b \equiv_L bb$ oder $ab \equiv_L ba$?
2. Sei nun $L = (ab)^*$. Gelten $a \equiv_L b$, $aa \equiv_L a$ oder $\varepsilon \equiv_L ab$? Wie lauten in diesem Fall die Äquivalenzklassen von $\equiv_L$?

Lösungsvorschlag

1.
   - $a \not\equiv_L aa$, da $ab \in L$, aber $aab \notin L$.
   - $b \equiv_L bb$, da für alle $u \in \Sigma^*$ gilt: $bu \notin L$ gdw. $bbu \notin L$.
   - $ab \not\equiv_L ba$, da $ab \in L$ aber $ba \notin L$.
2.
   - $a \not\equiv_L b$, da $ab \in L$, aber $bb \notin L$.
   - $aa \not\equiv_L a$, da $aab \notin L$, aber $ab \in L$.
   - $\varepsilon \equiv_L ab$, da für alle $u \in \Sigma^*$ gilt: $u \in L$ gdw. $abu \in L$.

   Die Äquivalenzklassen von $\equiv_L$ sind:

   - $[\varepsilon]_{\equiv_L} = L$, da $\epsilon \in L$ und für alle $w \in L$ und alle $w' \in \{a, b\}^*$ gilt: $\varepsilon w' = w' \in L$ gdw. $ww' \in L = LL$.
   - $[a]_{\equiv_L} = La$, da $a \in La$ und für alle $wa \in La$ und alle $w' \in \{a, b\}^*$ gilt: $waw' \in L$ gdw. $w' \in b(ab)^*$ gdw. $aw' \in L$.
   - $[b]_{\equiv_L} = \Sigma^* \setminus (L \cup La) = (ab)^*(b + aa)(a + b)^*$, da $bw \notin L$ für alle $w \in \{a, b\}^*$ gilt, und ebenso alle Wörter, die zwei aufeinanderfolgende gleiche Buchstaben enthalten, ebenfalls nie (Präfixe von Wörtern) in L sind.

T11

Zeigen Sie mit dem Satz von Myhill-Nerode, daß folgende Sprachen nicht regulär sind:

1. $L = \{ ww \mid w \in \{a, b\}^* \}$
2. $L = \{ a^n b^m \mid |n - m| < 5 \}$

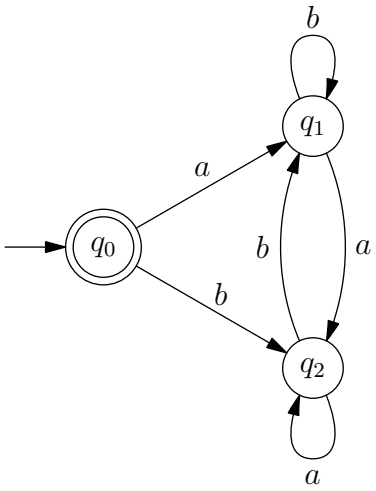
Lösungsvorschlag
Die hier verwendete Beweismethode benutzt jeweils eine unendliche Menge N und zeigt dann, daß alle Elemente von N in verschiedenen Äquivalenzklassen von $\equiv_L$ liegen. Daraus resultiert dann, daß $\equiv_L$ einen unendlichen Index hat, und L somit nach dem Satz von Myhill-Nerode nicht regulär sein kann.

1. Seien $N = a^*b$ und $u, v \in N$ mit $u = a^ib$ und $v = a^jb$ für irgendwelche $i \neq j$. Somit gilt $uu \in L$, aber $vu = a^jba^ib \notin L$. Dies bedeutet, daß u und v in verschiedenen Äquivalenzklassen von $\equiv_L$ liegen.

2. Seien $N = a^*$ und $u, v \in N$ mit $u = a^i$ und $v = a^j$ für irgendwelche $i > j$. Somit ist $a^ib^{i+4} \in L$, aber $a^jb^{i+4} \notin L$. Analog zu oben liegen a^i und a^j somit in verschiedenen Äquivalenzklassen.

T12
Sei $u \sim v$ gdw. $\hat{\delta}(q_0, u) = \hat{\delta}(q_0, v)$. Wie lauten die Äquivalenzklassen von $\sim$ für nebenstehenden Automaten M?

Sei nun $L = L(M) = [\varepsilon]_\sim$. Wie lauten die Äquivalenzklassen von $\equiv_L$?



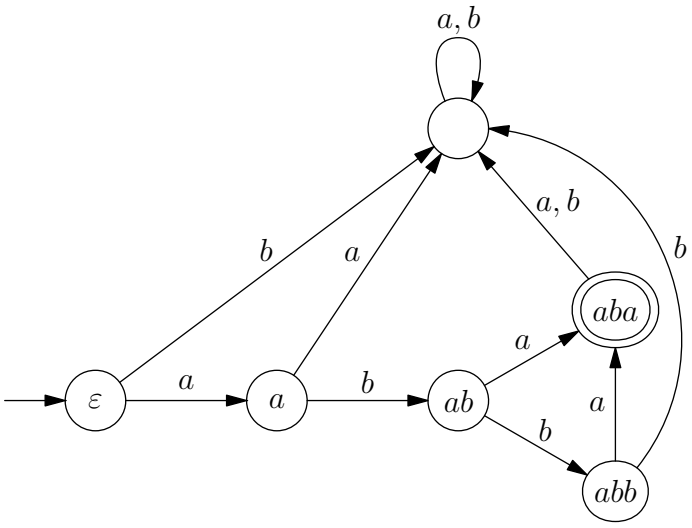
Lösungsvorschlag
Die Äquivalenzklassen von $\sim$ sind

- $[\varepsilon]_\sim = \{\varepsilon\}$,
- $[a]_\sim = a + (a+b)(a+b)^*b$, also das Wort a oder die Menge aller Wörter der Länge mindestens zwei, die mit b enden, sowie
- $[b]_\sim = b + (a+b)(a+b)^*a$, das Wort b oder die Menge aller Wörter der Länge mindestens zwei, die mit a enden.

Die Äquivalenzklassen von $\equiv_L$ sind $\{\varepsilon\}$ und $[a]_\sim \cup [b]_\sim = \Sigma^* \setminus \{\varepsilon\}$, wie leicht einzusehen ist.

H6 (10 Punkte)
Sei $L = \{aba, abba\}$. Ist $ab \equiv_L abb$? Wie lauten die Äquivalenzklassen von $\equiv_L$?

Lösungsvorschlag
Die Äquivalenzklassen von L sind $\{\varepsilon\}$, $\{a\}$, $\{ab\}$, $\{abb\}$, $\{aba, abba\}$ und $\hat{L} := \Sigma^* \setminus \{\varepsilon, a, ab, aba, abb, abba\}$.
Seien u und v Vertreter obiger Äquivalenzklassen, so daß $u \notin \hat{L}$ ist. Dann läßt sich jeweils leicht ein Wort $w \in \Sigma^*$ finden, so daß $uw \in L$ und $vw \notin L$ gelten (vgl. untenstehenden Automaten). Für jedes $u \in \hat{L}$ und alle $w \in \{a, b\}^*$ gilt hingegen, daß $uw \notin L$ ist, woraus folgt, daß es keine weiteren Äquivalenzklassen gibt.



H7 (10 Punkte)
Sei $L = \{ a^{n^2} \mid n \geq 0 \}$. Zeigen Sie mit Hilfe des Satzes von Myhill-Nerode, daß L nicht regulär ist.

Lösungsvorschlag
Sei $N = \{ a^{n^2-n} \mid n \geq 10 \}$. Seien $u, v \in N$ mit $u = a^{i^2-i}$, $v = a^{j^2-j}$ und $i < j$. Wegen $(j-1)^2 = j^2 - 2j + 1 < j^2 - j < j^2 - j + i < j^2$ gilt $a^{i^2-i}a^i = a^{i^2} \in L$ aber $a^{j^2-j}a^i = a^{j^2-j+i} \notin L$. Analog zu oben kann L somit nicht regulär sein.

H8 (10 Punkte)
Die Sprache der korrekt geklammerten Ausdrücke L sei induktiv wie folgt definiert:

- $\varepsilon \in L$.
- Für alle $u, v \in L$ ist auch $(u)v \in L$.

Beispielsweise ist (()(())) korrekt geklammert, aber ()) oder)(nicht.
Zeigen Sie mit dem Satz von Myhill-Nerode, daß L nicht regulär ist.

Lösungsvorschlag
Sei $N = \{ (^n \mid n \geq 0 \}$. Für beliebige $u = (^i$, $v = (^j$ mit $i \neq j$ gilt $u)^i \in L$ aber $v)^i \notin L$. Somit liegen alle Elemente von N in verschiedenen Äquivalenzklassen von $\equiv_L$, der Index von $\equiv_L$ ist also unendlich.

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 5
Emmes—Kneis—Langer—Rossmanith 20.5.2009

Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

T13

In der Vorlesung wurde nur eine Richtung des folgenden Lemmas bewiesen:

> Es sei $M = (Q, \Sigma, \delta, q_0, F)$ ein DFA.
>
> Die Zustände $q_1, q_2 \in Q$ sind unterscheidbar genau dann wenn:
>
> Es gibt $w \in \Sigma^*$ mit
>
> 1. $\hat{\delta}(q_1, w) \in F$, $\hat{\delta}(q_2, w) \notin F$ oder
> 2. $\hat{\delta}(q_2, w) \in F$, $\hat{\delta}(q_1, w) \notin F$.

Beweisen Sie die fehlende Richtung.

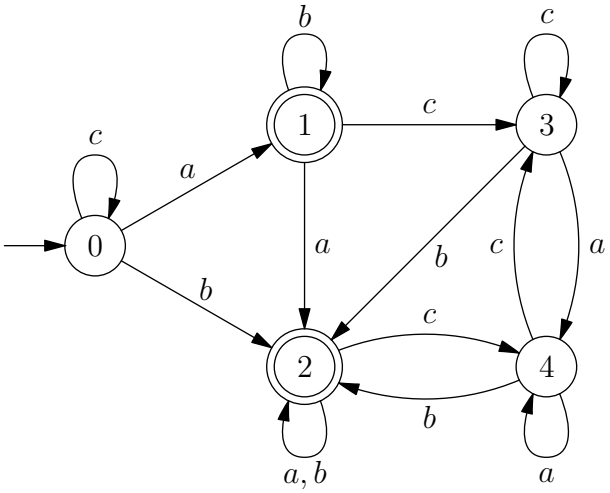
Lösungsvorschlag

Wir verwenden die (induktive) Definition von Unterscheidbarkeit für einen Induktionsbeweis:

- Induktionsanfang: Seien q_1 und q_2 unterscheidbar, weil $q_1 \in F$, aber $q_2 \notin F$ ist. Dann gilt natürlich $\hat{\delta}(q_1, \varepsilon) \in F$ und $\hat{\delta}(q_1, \varepsilon) \notin F$. Analog verhält es sich, wenn $q_1 \notin F$ und $q_2 \in F$ sind.
- Induktionsschritt: Seien q_1 und q_2 unterscheidbar, weil ein $a \in \Sigma$ existiert, so daß $q_1' := \delta(q_1, a)$ und $q_2' := \delta(q_2, a)$ unterscheidbar sind. Nach Induktionsvoraussetzung existiert daher ein $w \in \Sigma^*$ für das Zustandspaar q_1' und q_2', so daß $\hat{\delta}(q_1', w) \in F$ und $\hat{\delta}(q_2', w) \notin F$ oder $\hat{\delta}(q_1', w) \notin F$ und $\hat{\delta}(q_2', w) \in F$. Wegen $\hat{\delta}(q_1, aw) = \hat{\delta}(\delta(q_1, a), w) = \hat{\delta}(q_1', w)$ und $\hat{\delta}(q_2, aw) = \hat{\delta}(\delta(q_2, a), w) = \hat{\delta}(q_2', w)$ folgt sofort die Aussage.

T14

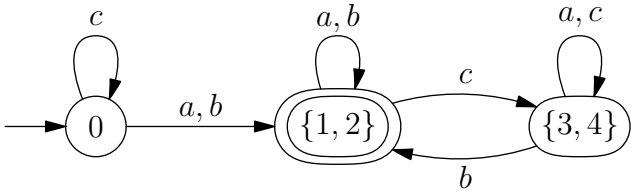
Minimieren Sie folgenden DFA und finden Sie anschließend einen regulären Ausdruck, der seine Sprache erzeugt.



Lösungsvorschlag

Der Markierungsalgorithmus zur Minimierung des Automaten ergibt folgende Tabelle.

	0	1	2	3	4
0		X	X	X	X
1	X			X	X
2	X			X	X
3	X	X	X		
4	X	X	X		



Wir können nun wahlweise den Automat weiter reduzieren oder direkt den regulären Ausdruck ablesen. Wir erhalten den Ausdruck

$$c^*(a+b)((a+b)+c(a+c)^*b)^*$$

T15

Richtig oder falsch?

a) Es gibt eine reguläre Sprache, die von einem NFA mit 10 Zuständen akzeptiert wird, aber von keinem DFA mit 100 Zuständen.

b) Es gibt eine reguläre Sprache, die von einem NFA mit 10 Zuständen akzeptiert wird, aber von keinem DFA mit 1217 Zuständen.

c) Zwei minimale DFAs, welche jeweils komplementäre Sprachen akzeptieren, haben gleich viele Zustände.

d) Zwei minimale NFAs, welche jeweils komplementäre Sprachen akzeptieren, haben gleich viele Zustände.

Lösungsvorschlag

a) Richtig: Wie in der Vorlesung gezeigt wurde, gibt es einen NFA mit $n+2$ Zuständen, der die Sprache $(a+b)^*a(a+b)^n$ akzeptiert, gleichzeitig aber jeder DFA wenigstens 2^n Zustände benötigt.

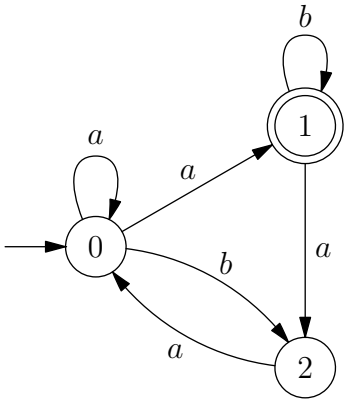
b) Falsch: Wenn ein NFA mit 10 Zuständen eine Sprache akzeptiert, dann hat der zugehörige Potenzmengenautomat höchstens $2^{10} = 1024 < 1217$ Zustände.

c) Richtig: Wenn M ein minimaler Automat für $L = L(M)$ ist, dann erhält man einen Automaten $\overline{M}$ mit $\overline{L} = L(\overline{M})$, indem man Nichtendzustände und Endzustände vertauscht (siehe Vorlesung), wobei die Größe des Automaten sich nicht ändert. Wenn $\overline{M}$ nun nicht minimal wäre, dann existiert $\overline{M}'$ mit $\overline{L} = L(\overline{M}) = L(\overline{M}')$ und $|\overline{M}'| < |\overline{M}|$. Der Komplementärautomat von $\overline{M}'$ wiederum, nennen wir ihn M', hat nun weniger Zustände als M, erkennt aber dennoch dieselbe Sprache. Ein Widerspruch zur Minimalität von M.

d) Falsch: Betrachte $L = \{\epsilon\}$ und $\overline{L} = \Sigma^* - \{\epsilon\}$. Dann enthält $\overline{L}$ alle Wörter, die wenigstens ein Zeichen enthalten. L kann man mit einem NFA erkennen, der nur einen einzigen Zustand, einen Endzustand, aber keine Transitionen enthält. Für $\overline{L}$ benötigt man hingegen auf jeden Fall mindestens zwei Zustände, da ϵ nicht akzeptiert werden darf.

H9 (15 Punkte)

Eine technische Anlage besitzt zwei Ventile, welche durch jeweils durch die Befehle a und b für eine Sekunde geöffnet werden können. Der Hersteller übernimmt keine Garantie, wenn nicht eine der Öffnungssequenzen verwendet wird, welche im Handbuch durch folgenden NFA spezifiert werden:



Genauer gesagt: Nachdem die Maschine eingeschaltet wurde, dürfen wir die Ventile laut einer erlaubten Öffnungssequenz öffnen und die Maschine anschließend wieder abschalten. Es ist also beispielsweise nicht erlaubt, die Maschine ein- und sofort wieder auszuschalten, da offensichtlich ϵ keine sichere Öffnungssequenze laut Handbuch ist.

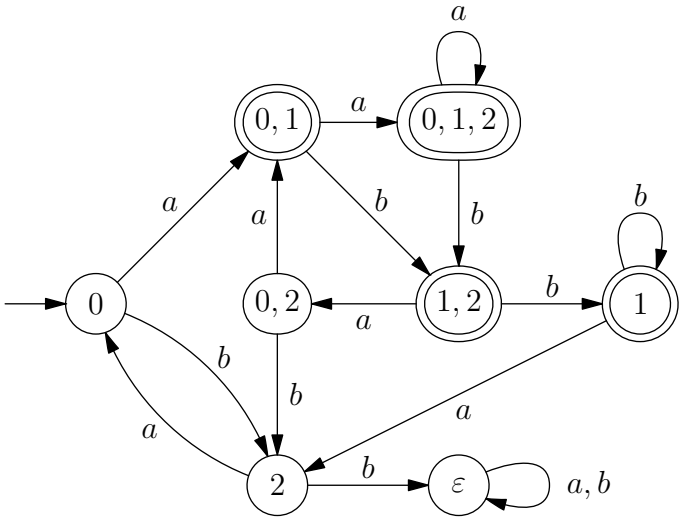
Uns interessieren jetzt allerdings die möglicherweise *unsicheren* Öffnungssequenzen, also gerade die, die durch den obigen NFA nicht akzeptiert werden.

Ist die Öffnungssequenz *aaabaabaa* sicher oder unsicher?

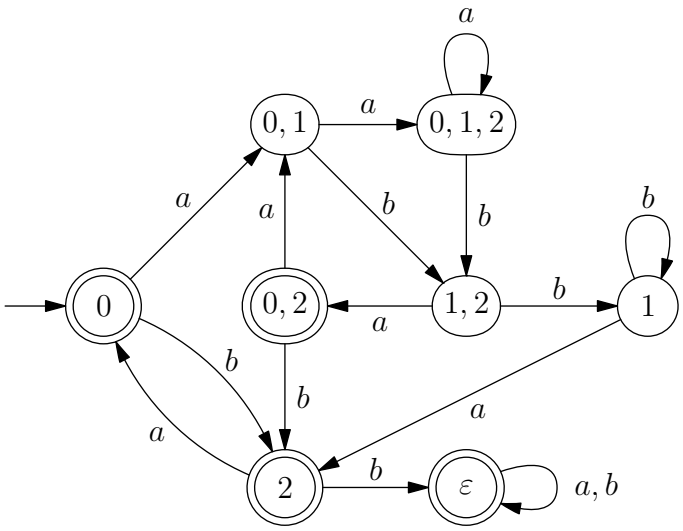
Konstruieren Sie sowohl einen DFA, als auch einen regulären Ausdruck für jene unsicheren Öffnungssequenzen. Erklären Sie, welche Konstruktionen Sie dabei verwenden.

Lösungsvorschlag

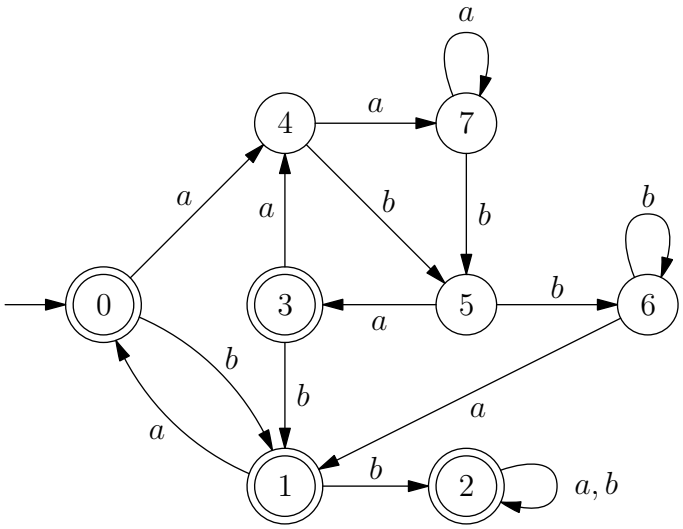
Durch die Potenzmengenkonstruktion kann der zu dem NFA äquivalente DFA gewonnen werden:



Nun kann die Negation der Sprache durch Vertauschen der Endzustände erreicht werden. Der resultierende Automat akzeptiert nun genau die unsicheren Öffnungssequenzen.



Zur Übersichtlichkeit benennen wir die Zustände um:



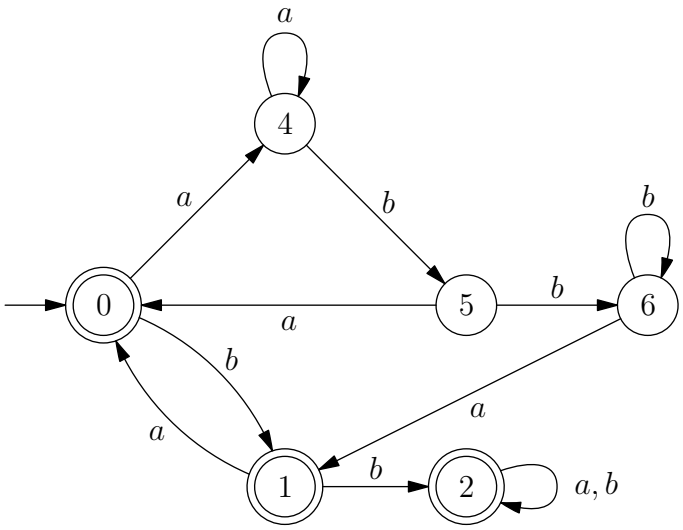
Um den regulären Ausdruck nicht zu groß werden zu lassen, führen wir die Minimierung durch den Markierungsalgorithmus aus der Vorlesung durch:

	7	6	5	4	3	2	1
0	X	X	X	X		X	X
1	X	X	X	X	X	X	
2	X	X	X	X	X		
3	X	X	X	X			
4		X	X				
5	X	X					
6	X						

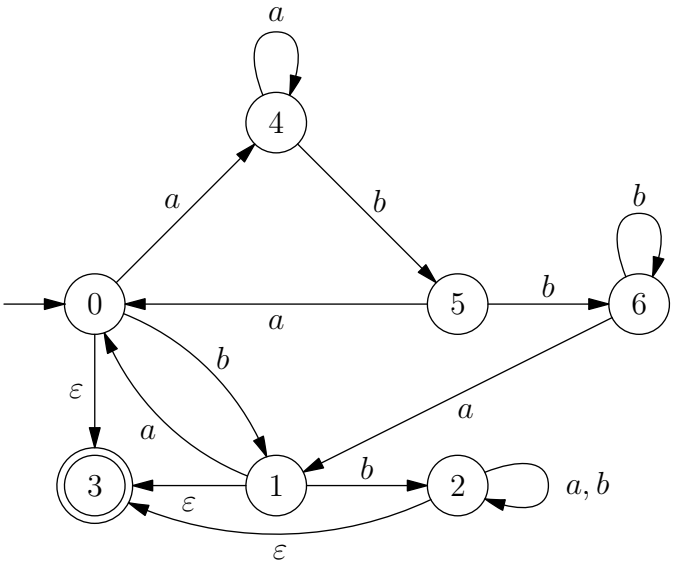
$0,3 \xrightarrow{a} 4,4$
$0,3 \xrightarrow{b} 1,1$
$0,2 \xrightarrow{a} 2,4$
$0,1 \xrightarrow{a} 0,4$
$1,3 \xrightarrow{a} 2,4$
$1,3 \xrightarrow{b} 1,2$
$1,2 \xrightarrow{a} 0,2$
$2,3 \xrightarrow{a} 2,4$

$4,7 \xrightarrow{a} 7,7$
$4,7 \xrightarrow{b} 5,5$
$4,6 \xrightarrow{a} 1,7$
$4,3 \xrightarrow{a} 3,7$
$5,7 \xrightarrow{a} 3,7$
$5,7 \xrightarrow{a} 1,3$
$6,7 \xrightarrow{a} 1,7$

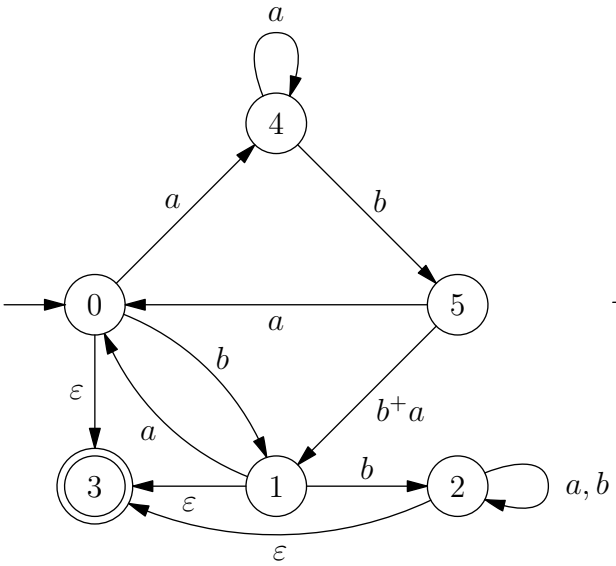
Somit können die Zustände 4 und 7 sowie 0 und 3 vereinigt werden:



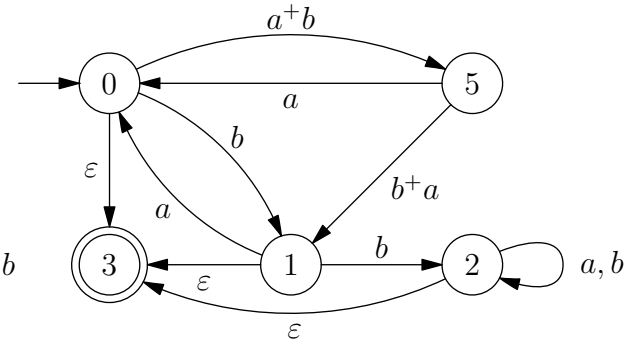
Um den regulären Ausdruck konstruieren zu können, führen wir einen eindeutigen Endzustand ein, der von den ursprünglichen Endzuständen über ε-Kanten erreicht wird:



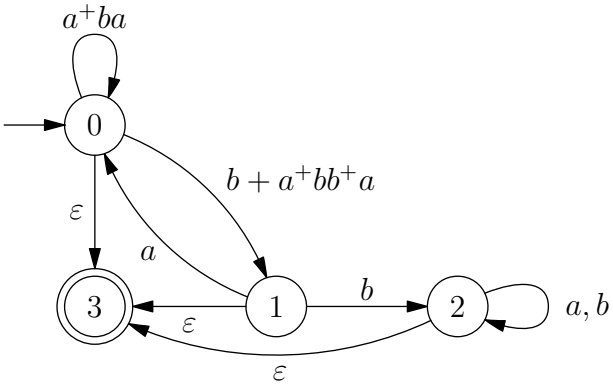
Nun können wir schrittweise Knoten aus dem Graphen entfernen und dabei die betroffenen Kanten mit den entsprechenden regulären Ausdrücken markieren.



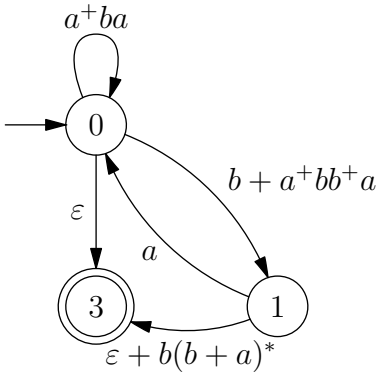
Zustand 6 entfernt.



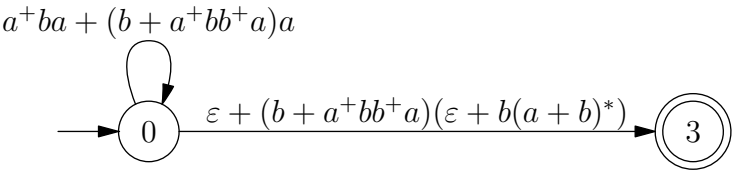
Zustand 4 entfernt.



Zustand 5 entfernt.



Zustand 2 entfernt.



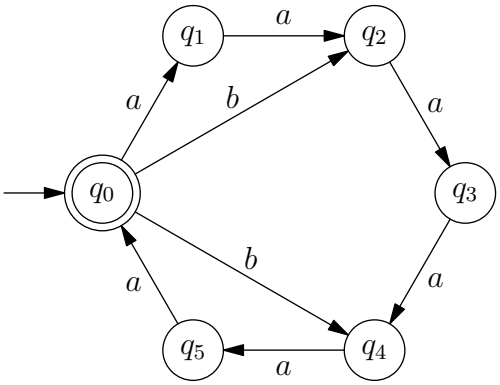
Zustand 1 entfernt.

$$\boxed{(a^+ba + (b + a^+bb^+a)a)^*(\varepsilon + (b + a^+bb^+a)(\varepsilon + b(a + b)^*))}$$

Ein regulärer Ausdruck, der die unsicheren Öffnungssequenzen beschreibt.

H10 (5 Punkte)

Konstruieren Sie einen kurzen regulären Ausdruck für die Sprache, die folgender NFA erkennt:

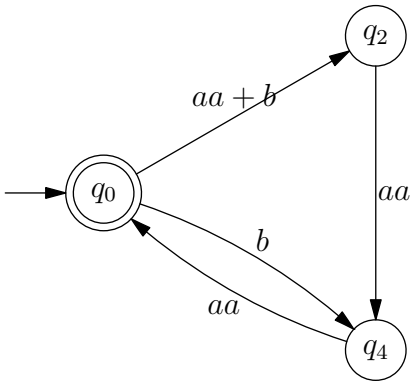


Hinweis: Die R_{ij}^k-Konstruktion ist nicht ratsam. Sie könnte zu einer Lösung wie dieser führen, selbst wenn man zwischendurch vereinfacht.

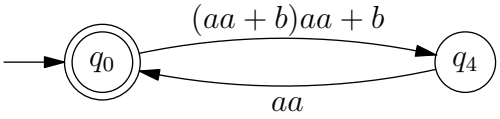
$\epsilon + ((b + ((b + aa)a + ((b + aa)a))a + (b + ((b + aa)a + ((b + aa)a))a))a)(\epsilon + (ab + ((ab + aaa)a + ((ab + aaa)a))a + (ab + ((ab + aaa)a + ((ab + aaa)a))a))a)^*a$

Lösungsvorschlag

Wir entfernen zuerst die Zustände q_1, q_3 und q_5 und erhalten so leicht den folgenden Automaten.



Anschließend entfernen wir den Knoten q_2, und können aus dem resultierenden Automaten



den regulären Ausdruck $(((aa + b)aa + b)aa)^*$ ablesen.

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 6
Emmes—Kneis—Langer—Rossmanith 27.5.2009

Übung zur Vorlesung Formale Sprachen, Automaten und Prozesse

T16

Der *Quotient* zweier Sprachen $L_1 \subseteq \Sigma^*$ und $L_2 \subseteq \Sigma^*$, in Zeichen L_1/L_2, ist wieder eine Sprache, die folgendermaßen definiert ist:

$$L_1/L_2 = \{\, u \in \Sigma^* \mid uv \in L_1 \text{ für ein } v \in L_2\}$$

a) Was ist a^*b^*/b^*?

b) Was ist $a^*b^*/\{\, a^nb^n \mid n > 0\,\}$?

c) Beweisen oder widerlegen Sie: Falls L_1 und L_2 regulär sind, dann ist auch L_1/L_2 regulär.

d) Beweisen oder widerlegen Sie: Falls L_1 regulär ist und L_2 nicht regulär ist, dann ist L_1/L_2 regulär.

e) Was bedeutet das für Aufgabe H4?

Lösungsvorschlag

a) Offensichtlich ist $a^*b^*/b^* \supseteq a^*b^*$, da $\epsilon \in b^*$ und somit jedes $w \in a^*b^*$ auch in a^*b^*/b^* enthalten ist.

Andererseits gilt auch $a^*b^*/b^* \subseteq a^*b^*$, da für jedes jedes Wort $u \in a^*b^*/b^*$ ein $v \in b^*$ existiert, mit $uv \in a^*b^*$. Damit gilt aber sofort $u \in a^*b^*$.

b) Wir zeigen $L_b := a^*b^*/\{\, a^nb^n \mid n > 0\,\} = a^*$. Es gilt $a^* \subseteq L_b$, da $a^ia^nb^n \in a^*b^*$ für alle $i \in \mathbf{N}_0$ und alle $n \in \mathbf{N}$.

Sei $w \in L_b$. Dann gibt es ein n und ein $u \in a^nb^n$ mit $wu \in a^*b^*$. Falls $|w|_b \geq 1$, dann folgt in wu nach einem b irgendwann ein a und somit wäre $wu \notin a^*b^*$, also auch $w \notin L_b$. Also folgt $|w|_b = 0$ und damit $L_b \subseteq a^*$.

c) Die Aussage ist wahr. Sei $\mathcal{M} = (\Sigma, Q, \delta, q_0, F)$ ein DFA der L_1 akzeptiert. Wir konstruieren einen DFA $\mathcal{M}' = (\Sigma, Q, \delta, q_0, F')$, der L_1/L_2 akzeptiert. Dazu setzen wir $F' = \{q \in Q \mid \text{ es existiert ein } u \in L_2 \text{ mit } \hat{\delta}(q, u) \in F\}$.

Sei nun w ein Wort in $L(\mathcal{M}')$. Dann existiert nach Konstruktion ein $v \in L_2$ mit $wv \in L_1$, da andernfalls $\hat{\delta}(q_0, w) \notin F'$ gelten würde. Also ist $w \in L_1/L_2$ und es gilt $L(\mathcal{M}') \subseteq L_1/L_2$.

Für die Rückrichtung, sei $w \in L_1/L_2$. Nach Definition existiert ein v mit $wv \in L_1$ und $v \in L_2$. Also gilt $\hat{\delta}(\hat{\delta}(q_0, w), v) \in F$ und damit nach Konstruktion $\hat{\delta}(q_0, w) \in F'$. Also gilt auch $L(\mathcal{M}') \supseteq L_1/L_2$.

Somit akzeptiert der DFA $\mathcal{M}'$ genau L_1/L_2.

d) Auch diese Aussage ist wahr. Der Beweis für c) kann direkt übernommen werden, allerdings kann M' nun nicht mehr leicht konstruiert werden. Existieren muss M' aber.

e) Da die Sprache L aus H4 als $La/\{a\}$ geschrieben werden kann und La nach Voraussetzung regulär war, folgt die Regularität von L sofort aus c).

T17

Verwenden Sie das Pumping-Lemma, um zu zeigen, daß folgende Sprachen nicht regulär sind:

a) $w \in \{\text{if}, \text{then}, \text{else}, \text{fi}, A\}^*$, wobei w einen korrekten bedingten Ausdruck darstellt. Hierbei soll A einen primitive Ausdruck verkörpern.

Beispielsweise ist

$$\text{if } A \text{ then } A \text{ else if } A \text{ then } A \text{ else } A \text{ fi fi}$$

korrekt, aber

$$\text{A then if else A fi fi}$$

ist aus vielerlei Gründen nicht korrekt.

b) Die korrekten arithmetischen Ausdrücke mit natürlichen Zahlen und den vier Grundrechenarten, z.B. $(4 + 23) \times (3 - 18)$.

c) Die korrekten arithmetischen Ausdrücke mit natürlichen Zahlen und Additon sowie Subtraktion, aber ohne Klammern, deren Ergebnis positiv ist.

Beispiel: $23 - 40 + 23 - 1 - 1 - 1 + 5$

Lösungsvorschlag

a) Beweis durch Widerspruch. Sei die Sprache der korrekten bedingten Ausdrücke regulär. Dann gibt es nach dem Pumping-Lemma eine Zahl n mit den beschriebenen Eigenschaften. Sei $w = \text{if}^n\text{A}(\text{then A fi})^n$ und $xyz = w$ eine beliebige Zerlegung von w mit $|xy| \leq n$ und $|y| > 0$. Somit ist xy ein Präfix von if^n und xy^2z kann kein korrekter bedingter Ausdruck sein, da er unterschiedliche Anzahlen von if und fi enthält.

b) Analog zu oben, sei $w = ({}^n[+1)]^n$. Somit gilt für jede Zerlegung $xyz = w$ mit $|xy| \leq n$ und $|y| > 0$, daß xy ein Präfix von $({}^n$ ist. Somit ist xy^2z kein korrekt geklammerter Ausdruck, da sich die Zahl der öffnenden und schließenden Klammern unterscheiden.

c) Angenommen, die Sprache L_c dieser Aufgabe sei regulär. Nach Pumpinglemma existiert dann ein $n \in \mathbf{N}$, sodaß für alle $w \in L_c$ mit $|w| \geq n$ eine Zerlegung $w = xyz$ existiert, mit $|xy| \leq n$, $|y| \geq 1$ und $xy^kz \in L_c$ für alle $k \in \mathbf{N}_0$.

Sei $w = 1^n - 1^n + 1$. Wertet man diesen Ausdruck aus, so ergibt sich offensichtlich 1, also ist w in der Sprache enthalten. Außerdem ist offensichtlich $|w| > n$.

Sei $xyz = w$ eine beliebige Zerlegung von w mit $|xy| \leq n$ und $|y| \geq 1$. Dann ist $xy^0z = xz = 1^{n-|y|} - 1^n + 1$. Offensichtlich ist aber xz nicht in L_c, da der zugehörige Wert kleiner als 1 ist. Dies ist ein Widerspruch zur Aussage des Pumpinglemmas, also kann L_c nicht regulär sein.

H11 (3 Punkte)
Es seien $L_1, L_2 \subseteq \Sigma^*$ für ein Alphabet Σ.
Beweisen oder widerlegen Sie folgende Aussage:

$$(L_1/L_2)L_2 = L_1$$

Falls Sie „nein“ antworten, welche der beiden Inklusionen gilt nicht?

Lösungsvorschlag
$\underline{(L_1/L_2)L_2 \not\subseteq L_1}$:
Sei $L_1 = \{ab\}$ und $L_2 = \{b, c\}$. Dann ist $a \in L_1/L_2$, da $ab \in L_1$ und $b \in L_2$, aber $ac \notin L_1$.
$\underline{L_1 \not\subseteq (L_1/L_2)L_2}$:
Sei $L_2 = \emptyset$, dann ist $(L_1/L_2)L_2 = \emptyset$. Sei $L_1 \neq \emptyset$. Widerspruch.

H12 (10 Punkte)
Beweisen Sie mithilfe des Satzes von Myhill–Nerode, daß die drei in Aufgabe T17 erwähnten Sprachen nicht regulär sind.

Lösungsvorschlag

a) Sei $N = \{\text{if}^n \mid i \in \mathbf{N}\}$. Dann sind je zwei verschiedene Wörter if^i, if^j aus N nicht aquivalent bezüglich $\equiv_{L_a}$, da $if^i A(\text{then}A\text{fi})^i$ ein gültiger Audruck ist, $if^j A(\text{then}A\text{fi})^i$ allerdings nicht. Somit hat $\equiv_{L_a}$ unendlich viele Klassen, also ist L_a nicht regulär.

b) Sei $N = \{(^i \mid i \in \mathbf{N}\}$. Dann ist $[(^i]_{\equiv_{L_b}} \neq [(^j]_{\equiv_{L_c}}$ für $i \neq j$, da $(^i 1 + 1)^i$ ein gültiger Ausdruck ist, $(^j 1 + 1)^i$ allerdings nicht. Also hat $\equiv_{L_b}$ unendliche viele Aquivalenzklassen, woraus folgt, daß L_b nicht regulär ist.

c) Sei $N = \{(1+)^i 0 \mid i \in \mathbf{N}\}$. Dann ist $[u]_{\equiv_{L_c}} \neq [v]_{\equiv_{L_c}}$ für alle $u \neq w \in N$: Sei obdA $u = (1+)^i 0$ und $v = (1+)^j 0$ mit $i > j$. Dann ist $u - i \in L_c$ aber $v - i \notin L$. Somit hat $\equiv_{L_c}$ unendlich viele Klassen, also ist L_c nicht regulär.

H13 (10 Punkte)
Eine Zeichenmaschine wird durch ein Wort über dem Alphabet $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$ gesteuert.
Jedes Symbol entspricht einem Befehl, eine 1 cm lange Linie zu zeichnen, wobei es für jede der vier Himmelsrichtungen ein Symbol gibt.
Es ist ein Fehler, wenn ein ein Zentimeter langes Segment auf dem Papier mehr als zehnmal überschrieben wird (egal, ob vorwärts oder rückwärts). Solche Zeichenbefehle können nämlich dazu führen, daß das Papier an einer solchen Stelle durch zu viel Tinte aufweicht und reißt.
Verwenden Sie das Pumping-Lemma, um zu beweisen, daß die Sprache der sicheren Zeichenbefehle leider nicht regulär ist und daher die Maschine nicht durch einen endlichen Automaten überwacht werden kann.

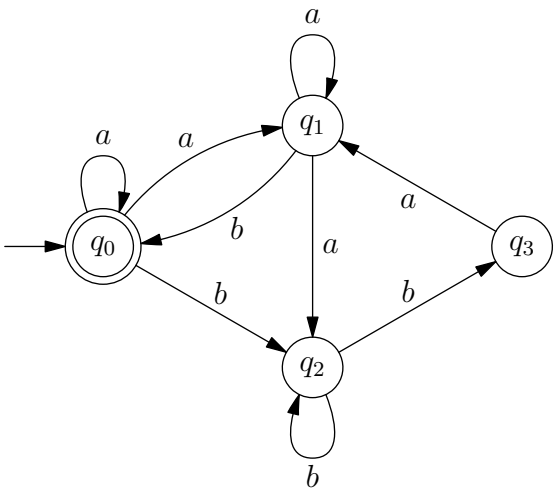
Lösungsvorschlag

Angenommen, die Sprache L dieser Aufgabe sei regulär. Nach Pumpinglemma existiert dann ein $n \in \mathbf{N}$, sodaß für alle $w \in L$ mit $|w| \geq n$ eine Zerlegung $w = xyz$ existiert, mit $|xy| \leq n$, $|y| \geq 1$ und $xy^k z \in L$ für alle $k \in \mathbf{N}_0$.

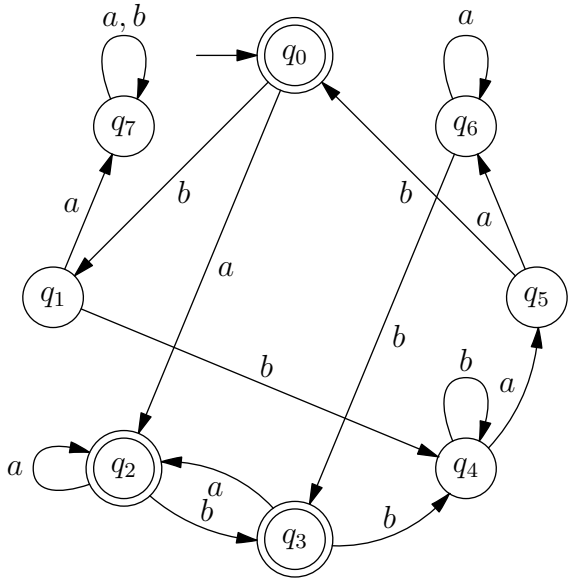
Sei $w = \uparrow^n \downarrow^n (\downarrow\uparrow)^{10}$. Offensichtlich ist w ein gültiger Zeichenbefehl. Für jede entsprechende Zerlegung $w = xyz$ ist aber $y = \uparrow^l$ für ein $1 \leq l \leq n$. Dann ist aber $xy^0z = xz = \uparrow^n \downarrow^{n-l} (\downarrow\uparrow)^{10}$ kein gültiger Zeichenbefehl, da auf Höhe l elf mal über die gleiche Stelle gezeichnet wird.

H14 (10 Punkte)

Wie sieht der minimale DFA zu folgendem NFA aus?



Lösungsvorschlag



H15 (10 Punkte)

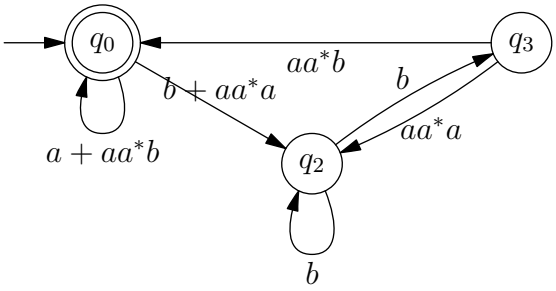
Konstruieren Sie einen äquivalenten regulären Ausdruck zum Automaten aus Aufgabe H14.

Wird das Wort *aaabbbaaaba* akzeptiert?

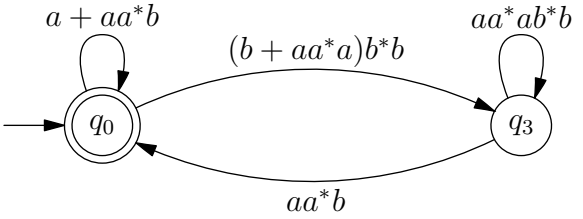
Wird das Wort *abbbbabbbaa* akzeptiert?

Lösungsvorschlag

Wir eliminieren zunächst den Zustand q_1 und erhalten:



Nach der Eliminierung von q_2 erhalten wir:



Nun können wir den regulären Ausdruck ganz einfach ablesen:

$$\Big(a + aa^*b + (b + aa^*a)b^*b(aa^*ab^*b)^*aa^*b\Big)^*$$

Am deterministischen Automaten aus H14 erkennen wir sofort, daß das erste Wort erkannt wird, das zweite aber nicht.

H16 (3 Punkte)

Geben Sie einen regulären Ausdruck für die Menge aller Wörter über dem Alphabet $\{a, b\}$ an, die genau dann eine gerade Anzahl von as enthalten, wenn sie auch eine gerade Anzahl von bs enthalten.

Hinweis: Denken Sie erst einmal genau nach, bevor Sie die geeigneten Konstruktionen der Vorlesung durchführen, welche natürlich auch zum Ziel führen.

Lösungsvorschlag

Wir beginnen mit einem sehr einfachen DFA mit vier Zuständen, der die Anzahl der bisher gelesenen a und b modulo 2 zählt. Wenn die entsprechenden Zustände z.B. die Namen $q_{a0,b0}$, $q_{a0,b1}$, $q_{a1,b0}$ und $q_{a1,b1}$ tragen, dann sind die Zustände $q_{a0,b0}$ (beide eine gerade Anzahl) und $q_{a1,b1}$ (ungerade Anzahl) die Endzustände. Für die bekannten Konstruktionen benötigen wir allerdings einen Automaten mit nur einem Endzustand, welches den Automaten vergrößert und die Konstruktion unnötig kompliziert werden lässt.

Wir behelfen uns mit der Beobachtung, daß die vorgegebene Sprache die Vereinigung der Sprache L_1 der Wörter mit gerader Anzahl a und b vereinigt mit der Sprache L_2 der Wörter mit ungerader Anzahl a und b ist. Die jeweiligen Automaten enthalten dann jeweils nur einen Endzustand, und wir kommen durch die Zustandselimination leicht zu folgenden Ergebnissen:

$$L_1 = \Big(aa + bb + (ab + ba)(aa + bb)^*(ab + ba)\Big)^*$$

$$L_2 = (aa + bb)^*(ab + ba)\Big(aa + bb + (ab + ba)(aa + bb)^*(ab + ba)\Big)^*$$

Wegen $L_2 = (aa + bb)^*(ab + ba) \cdot L_1$ ist ein schöner regulärer Ausdruck für die Sprache dann

$$\Big(\epsilon + (aa + bb)^*(ab + ba)\Big)\Big(aa + bb + (ab + ba)(aa + bb)^*(ab + ba)\Big)^*.$$

Die Lösungen zu diesem Aufgabenblatt können in der Tutorübung am 17. Juni abgegeben werden. Nächste Woche finden aufgrund der Exkursionswoche keine Tutorübungen statt und in der übernächsten Woche auf Grund des Dies Academicus.

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 7
Emmes—Kneis—Langer—Rossmanith 17.6.2009

Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

T18

Gegeben sei die einfache kontextfreie Grammatik

$$S \to SS \mid SaSbS \mid SbSaS \mid \epsilon.$$

a) Welche Sprache erzeugt diese Grammatik?

b) Beweisen Sie Ihre Vermutung formal und vollständig.

c) Ist die Grammatik eindeutig?

Lösungsvorschlag

a) Die Grammatik erzeugt die Sprache in der in jedem Wort gleich viele a und b vorkommen.

b) Sei G die gegebene Grammatik.

Das Symbol S läßt sich nur zu Wörtern mit gleich vielen a und b ableiten:
Die Anwendung einer beliebigen Regel führt gleich viele a und b Symbole ein. Da im Startsymbol S die Zahl der a und b Symbole gleich ist, muß in jedem abgeleiteten Wort ebenfalls die gleiche Zahl von a und b Symbolen vorliegen.

Jedes Wort w mit gleich vielen a und b liegt in $L(G)$:
Beweis per Induktion über die Wortlänge. Für $|w| = 0$ gilt die Annahme, da das Wort $w = \varepsilon$ offensichtlich in $L(G)$ liegt. Sei $|w| > 0$ und $w = aw'$. Dann gilt $|w'|_a + 1 = |w'|_b$. Es gibt einen kleinsten Präfix x von w', in dem $|w'|_a + 1 = |w'|_b$ gilt, da in w' mehr b als a vorkommen. Dieser kann nicht auf a enden, da ansonsten ein kleinerer Präfix mit den gleichen Bedingungen existiert. Somit existiert eine Zerlegung $w' = xby$ mit $|x|_a = |x|_b$ und $|y|_a = |y|_b$. Da $|x| < |w|$ und $|y| < |w|$, folgt aus der Induktionshypothese daß $x, y \in L(G)$. Die Regel $S \to SaSbS$ impliziert, daß für alle $w_1, w_2, w_3 \in L(G)$ auch $w_1 a w_2 b w_3 \in L(G)$ gilt. Da $x, y, \varepsilon \in L(G)$, gilt folglich auch $w = \varepsilon axby \in L(G)$. Für $|w| > 0$ und $w = bw'$ gilt die Behauptung analog, da auch die Regel $S \to SbSaS$ in der Grammatik enthalten ist.

c) Nein. Beispielsweise läßt sich das Wort ab auf verschiedene Weisen ableiten:

- $S \Rightarrow SS \Rightarrow SaSbSS \overset{*}{\Rightarrow} ab$
- $S \Rightarrow SS \Rightarrow SSaSbS \overset{*}{\Rightarrow} ab$

T19

Es sei eine weitere einfache Grammatik gegeben:

$$S \to iStSeSf \mid iStSf \mid a$$

Konstruieren Sie NFAs, welche folgende Sprachen akzeptieren:

a) $pre^*(\{iataeiataff\})$

b) $pre^*(\{iiataeiaataeiatafff\})$

c) $pre^*(\{a^*\})$

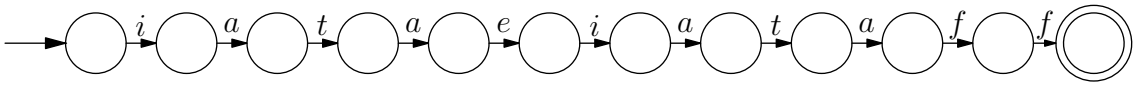
d) $pre^*(\{i,t,e,f,a\}^*)$

e) $pre^*(\emptyset)$

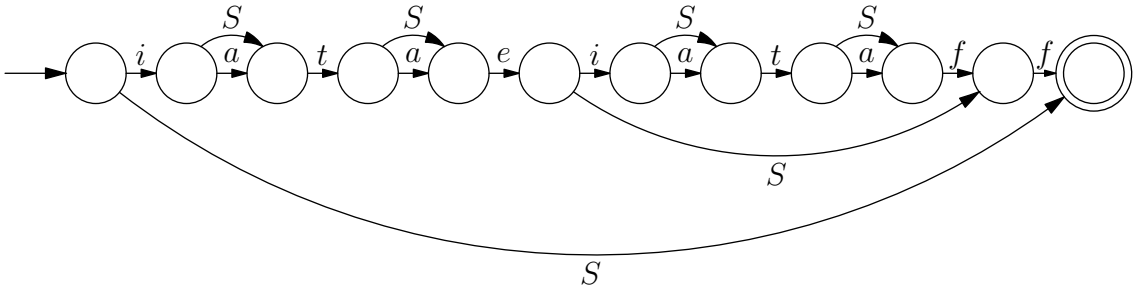
f) $pre^*(\{i,t,e,f\}^*)$

Lösungsvorschlag

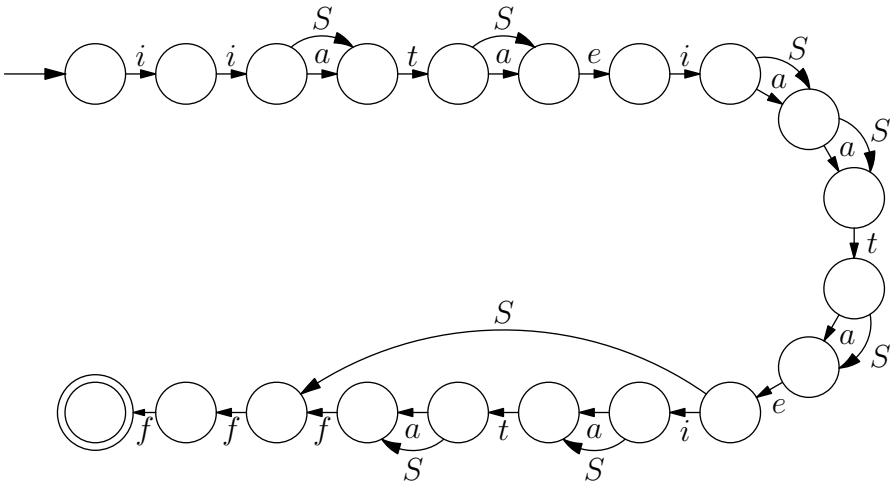
a) Zwischenschritt:



Vollständig:



b)



c) $pre^*(\{a^*\}) = \{a^*, S^*\}$

d) $pre^*(\{i,t,e,f,a\}^*) = \{i,t,e,f,a,S\}^*$

e) $pre^*(\emptyset) = \emptyset$

f) $pre^*(\{i,t,e,f\}^*) = \{i,t,e,f\}^*$

H17 (12 Punkte)

Geben Sie eine eindeutige kontextfreie Grammatik an, welche die Sprache aus Aufgabe T18 erzeugt. Beweisen Sie sowohl die Korrektheit, als auch die Eindeutigkeit Ihrer Grammatik.

Lösungsvorschlag

Eine eindeutige Grammatik, die die Sprache erzeugt, ist die folgende:

$$\begin{aligned} S &\rightarrow aAbS \mid bBaS \mid \varepsilon \\ A &\rightarrow aAbA \mid \varepsilon \\ B &\rightarrow bBaB \mid \varepsilon \end{aligned}$$

Wir treffen folgende Annahmen: Das Nichtterminal S erzeugt alle Wörtern mit gleich vielen a und b. Das Nichtterminal A leitet ebenfalls Wörter mit gleich vielen a und b ab, allerdings nur genau die, bei denen *jeder* Präfix gleich viele oder mehr a- als b-Symbole enthält. Im Gegensatz dazu leitet B alle Wörter mit gleich vielen a- und b-Symbolen ab, bei denen jeder Präfix gleich viele oder mehr b- als a-Symbole enthält.

Die Grammatik kann nur Wörter mit gleich vielen a und b erzeugen, da jede rechte Regelseite gleich viele a- und b-Symbole enthält. Es bleibt zu zeigen, daß jedes Wort mit gleich vielen a- und b-Symbolen auch von der Grammatik mit einer eindeutigen Ableitung erzeugt wird.

Wir führen den etwas allgemeineren Beweis, daß S, A und B die oben beschriebenen Eigenschaften erfüllen und eindeutige Ableitungen haben, per Induktion über die Wortlänge eines beliebigen Wortes w.

Sei $w = \varepsilon$, so kann das Wort durch jedes der Nichtterminale S, A und B erzeugt werden. Da w keinen echten Präfix besitzt, sind die oben beschriebenen Bedingungen erfüllt. Die jeweiligen Ableitungen sind eindeutig, da alle anderen Regeln Terminalsymbole einführen und somit kein ε ableiten können.

Nun sei $w \neq \varepsilon$ für ein Wort w mit $|w|_a = |w|_b$. Sei zunächst w so, daß es mit einem a beginnt. Wir betrachten die Präfixe von w, die gleich viele a- und b-Symbole haben. Den kleinsten solchen Präfix nennen wir u, wobei es natürlich sein kann, daß $u = w$ gilt. Nach Konstruktion hat u also keinen *echten* Präfix, mit gleich vielen a und b Symbolen (denn u war ja der kleinste Präfix von w, der diese Eigenschaft hat, und jeder Präfix von u ist natürlich auch Präfix von w).

Wenn an ein Wort ein weiterer Buchstabe aus $\{a, b\}$ angehängt wird, so verändert sich die Zahl seiner a- gegenüber seiner b-Symbole um genau eins. Somit folgt, daß der letzte Buchstabe in u ein b sein muß; wäre es ein a, hätte u einen echten Präfix mit gleich vielen a- und b-Symbolen. Somit läßt sich u zerlegen in $au'b$, wobei u' keinen Präfix enthält, der mehr b- als a-Symbole enthält. Da außerdem u' kürzer als w ist und gleich viele a und b Symbole enthält, kann es nach Induktionsvoraussetzung eindeutig durch A erzeugt werden. Wir haben w also zerlegt in $w = au'bv$. Über v wissen wir ebenfalls, daß es gleich viele a und b Symbole enthält und kürzer ist als w. Somit läßt es sich nach Induktionsvoraussetzung aus S eindeutig erzeugen. Daraus folgt, daß w sich mit der Regel $S \rightarrow aAbS$ erzeugen läßt. Dies ist die einzige S Regel, die ein mit a beginnendes Wort ableitet. Somit wäre eine uneindeutige Ableitung nur möglich, falls wir die Zerteilung von w anders wählen. Dies funktioniert allerdings nicht: Benutzen wir nicht den ersten Präfix u, der gleich viele a- und b-Symbole hat, hätte u' aufgrund der Zerlegung $u = au'b$ einen Präfix mit mehr b- als a-Symbolen und ließe sich somit nicht aus A herleiten.

Hat w keinen Präfix mit mehr b- als a-Symbolen, so gilt dies auch für v. Nach Induktionsvoraussetzung ist v somit eindeutig durch A herleitbar. In diesem Fall läßt sich w durch A mit der Regel $A \rightarrow aAbA$ erzeugen. Diese Ableitung ist, analog zu oben, ebenfalls eindeutig. Für den Fall, daß w mit einem b beginne ist der Beweis analog.

H18 (8 Punkte)
Gegeben ist die CFG $G = (N, T, P, S)$ mit $N = \{A, B, S\}$, $T = \{a, b\}$ und den Produktionen

$$S \to aBS \mid bAS \mid \epsilon$$
$$A \to bAA \mid a$$
$$B \to aBB \mid b.$$

Konstruieren Sie nichtdeterministische endliche Automaten, welche folgende Sprachen akzeptieren:

a) $pre^*(\{aabaa\})$

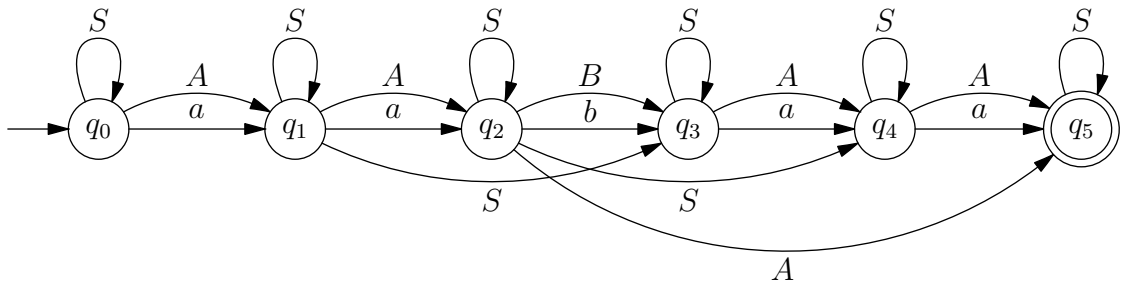
b) $pre^*(\{abba\})$

c) $pre^*(a^*)$

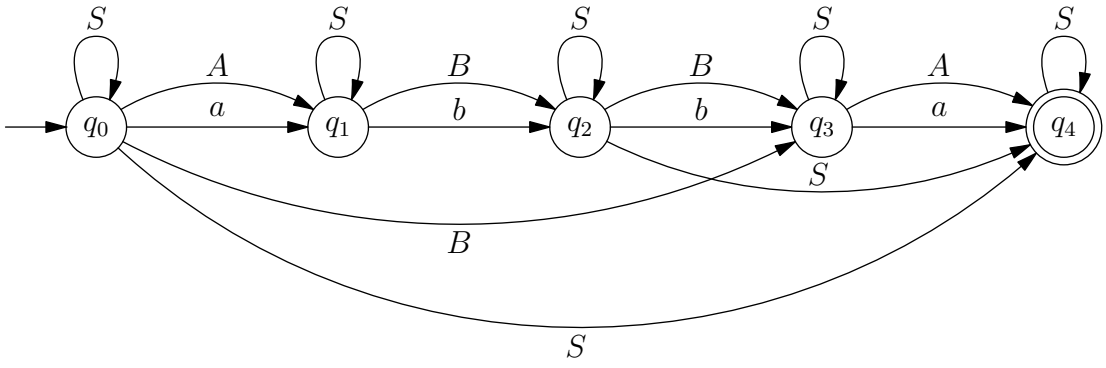
Gibt es ein $\alpha \in (aSb)^+$ mit $S \overset{*}{\Rightarrow} \alpha$?

Lösungsvorschlag

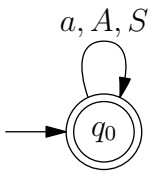
a)



b)



c)



Es gibt kein $\alpha \in (aSb)^+$ mit $S \overset{*}{\Rightarrow} \alpha$, da $S \notin pre^*((aSb)^+)$.

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 8
Emmes—Kneis—Langer—Rossmanith 24.6.2009

Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

Sei G die folgende kontextfreie Grammatik:

$$\begin{array}{lcl} S & \rightarrow & ABC \mid BCD \mid BD \\ A & \rightarrow & \varepsilon \mid Sa \mid aB \\ B & \rightarrow & AS \mid aD \\ C & \rightarrow & SC \mid bC \\ D & \rightarrow & bB \mid cC \mid dD \mid b \mid c \mid d \end{array}$$

H19 (10 Punkte)

a) Berechnen Sie die unerreichbaren Symbole von G. Die unerreichbaren Symbole sind definiert als $\{ A \in N \mid \text{es gibt kein } \alpha A \beta \in (N \cup T)^* \text{ mit } S \stackrel{*}{\Rightarrow} \alpha A \beta \}$.

b) Die unproduktiven Symbole einer Grammatik sind definiert als:

$$\{ A \in N \mid \text{es gibt kein } w \in T^* \text{ mit } A \stackrel{*}{\Rightarrow} w \}$$

Wie lauten die unproduktiven Symbole von G?

c) Gilt $L(G) = \emptyset$?

d) Gilt $\varepsilon \in L(G)$?

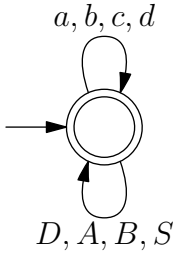
e) Die nullierbaren Symbole einer Sprache sind alle Symbole die sich nach ε ableiten lassen. Wie lauten die nullierbaren Symbole von G?

f) Geben sie eine kontextfreie Grammatik G' an, so daß $L(G') = L(G)$. Hierbei darf G' keine unproduktiven, unerreichbaren oder nullierbaren Symbole enthalten.

Lösungsvorschlag

a) Durch die beiden Regeln $S \Rightarrow ABC$ und $S \Rightarrow BCD$ sind schon alle Nichtterminale erreichbar.

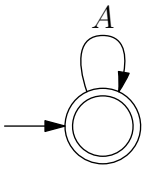
b) Die unproduktiven Zustände sind $N \setminus pre^*(T^*)$:



Somit ist C unproduktiv.

c) Da S produktiv ist, ist die von G erzeugte Sprache nicht leer.

d) Genau wenn S in $pre^*(\{\varepsilon\})$ enthalten ist, enthält $L(G)$ das leere Wort.



Aber $pre^*(\{\varepsilon\}) = \{A\}$.

e) Nur A läßt sich nach ε ableiten (siehe Teil d).

f) Regeln mit unerreichbaren oder unproduktiven Symbolen dürfen direkt gelöscht werden, da sie, in einer Ableitung vom Startsymbol aus, entweder nicht verwendet werden können (unerreichbar) oder bei Verwendung eine Ableitung eines Wortes aus Terminalsymbolen unmöglich machen (unproduktiv). Nullierbare Symbole können wie folgt behandelt werden:

1) Sei $A \to \varepsilon \in G$ und $A \neq S$.
2) Lösche $A \to \varepsilon$ aus G.
3) Für jede Regel $B \to \alpha A \beta$ füge die Regel $B \to \alpha\beta$ in G ein.
4) Falls keine Regel der Form $A \to \varepsilon$ mehr existiert fertig, ansonsten gehe zu 1.

Somit hat die resultierende Grammatik keine Regeln der Form $A \to \varepsilon$ für $A \neq S$. Die Regel $S \to \varepsilon$ gehört genau dann zur Grammatik, wenn $\varepsilon \in L(G)$ ist.

Für die Grammatik G ergibt sich also folgende Grammatik G':

$$\begin{array}{lcl} S & \to & BD \\ A & \to & Sa \mid aB \\ B & \to & AS \mid S \mid aD \\ D & \to & bB \mid dD \mid b \mid c \mid d \end{array}$$

H20 (10 Punkte)

a) Ist $L(G)$ endlich?

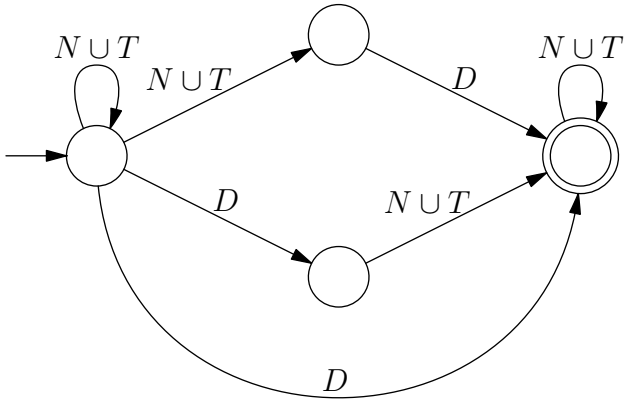
b) Ist $L(G)$ universell, d.h. $L(G) = \{a, b, c, d\}^*$?

c) Gilt $(abcd)^* \cap L(G) = \emptyset$?

d) Gilt $(ad + da)^* ca (db)^* \cap L(G) = \emptyset$?

Lösungsvorschlag

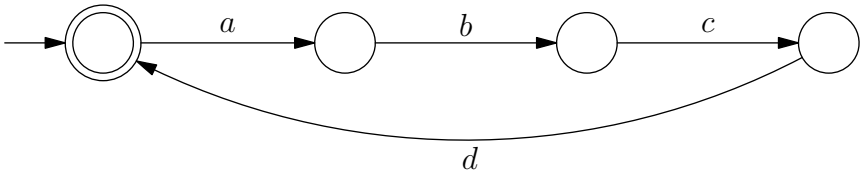
a) Nach einem Satz aus der Vorlesung ist $L(G)$ genau dann endlich, falls es kein $A \in N$ gibt, so daß $A \in pre^*\left((N \cup T)^+ A (N \cup T)^* \cup (N \cup T)^* A (N \cup T)^+\right)$. Wähle $A = D$.



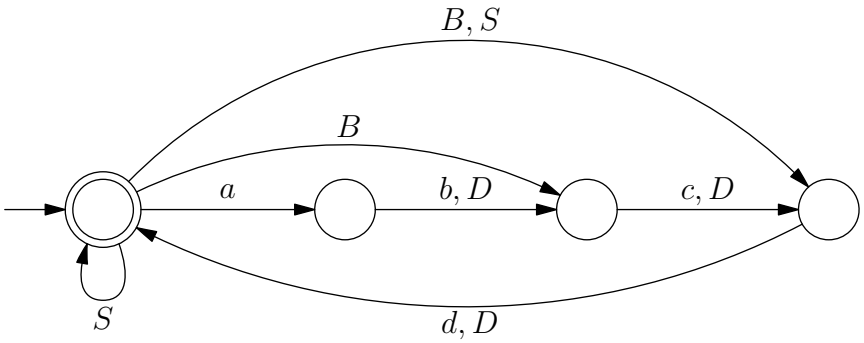
Da $D \to dD \in G$, gilt $D \in pre^*\left((N \cup T)^+ D (N \cup T)^* \cup (N \cup T)^* D (N \cup T)^+\right)$.

b) Nein, denn $\varepsilon \notin L(G)$.

c) Wir testen ob $S \in pre^*((abcd)^*)$, indem wir einen DFA für die Sprache $(abcd)^*$ bauen und schrittweise um Kanten ergänzen. Können wir eine mit S markierte Kante vom Anfangszustand des Automaten zu einem Endzustand hinzufügen, existiert ein Wort in der Schnittmenge der Sprachen. Ein DFA für $(abcd)^*$ ist beispielsweise der Folgende:

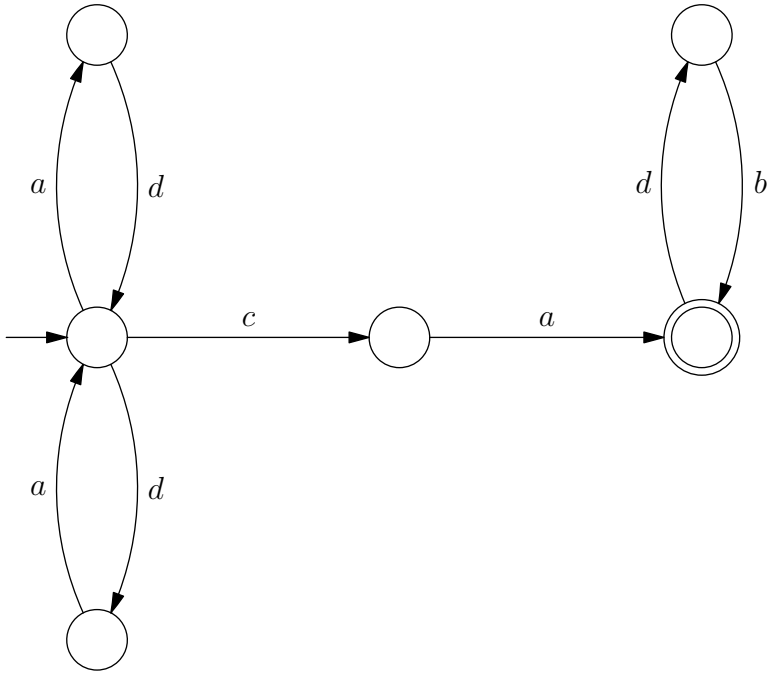


Durch Hinzufügen von Kanten (wir verwenden hierfür die vereinfachte Grammatik G' aus Aufgabe H19) ergibt sich folgender DFA:

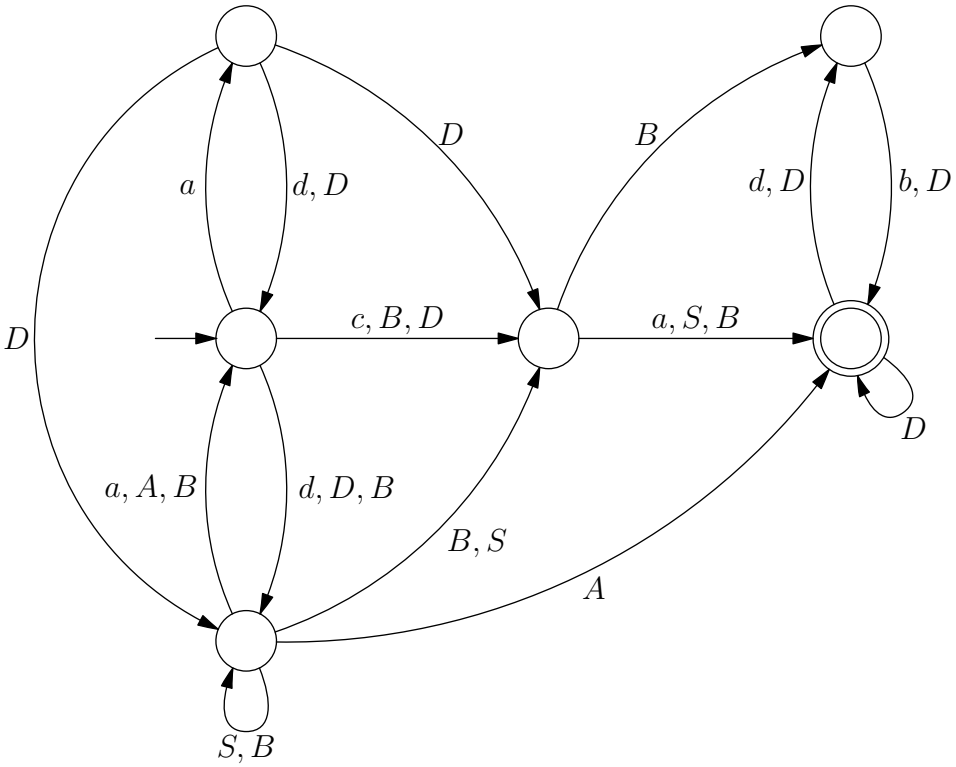


Somit ist die Schnittmenge der Sprachen nicht leer.

d) Analog zu c) starten wir mit einem DFA der die Sprache $(ad+da)^*ca(db)^*$ erkennt.



Hinzufügen von Kanten ergibt folgenden DFA:



Es lassen sich keine weiteren Kanten hinzufügen. Somit gilt $S \notin pre^*((ad+da)^*ca(db)^*)$ und die Schnittmenge der Sprachen ist leer.

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 9
Emmes—Kneis—Langer—Rossmanith 1.7.2009

Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

Sei G die folgende Grammatik. Hierbei sind Terminalsymbole unterstrichen und Nichtterminale kursiv dargestellt.

Ausdruck	→	(*Ausdruck*) | *Ausdruck* Operator *Ausdruck*
		| Zahl | Bezeichner | Bezeichner (*Ausdruck*)
Anweisung	→	if *Ausdruck* then *Anweisung*
Anweisung	→	if *Ausdruck* then *Anweisung* else *Anweisung*
Anweisung	→	begin *Anweisungsfolge* end
Anweisung	→	Bezeichner := *Ausdruck*
Anweisung	→	var Bezeichner
Anweisung	→	return *Ausdruck*
Anweisungsfolge	→	*Anweisung* | *Anweisungsfolge* ; *Anweisung*
Funktionsdefinition	→	function Bezeichner (Bezeichner)
		begin *Anweisungsfolge* end
Funktionsdefinitionsfolge	→	*Funktionsdefinition*
		| *Funktionsdefinitionsfolge Funktionsdefinition*
Programm	→	program Bezeichner *Funktionsdefinitionsfolge*
		begin *Anweisungsfolge* end .

T20

Überführen Sie die Grammatik G in die Greibach-Normalform. Achtung: Das Verfahren aus dem Beweis der Vorlesung führt zu einer sehr großen Grammatik. Wie kann man geschickter vorgehen?

Lösungsvorschlag

Als ersten Schritt entfernen wir die Linksrekursion in den *Funktionsdefinitionsfolge*- und *Anweisungsfolge*-Regeln durch Umordnen der rechten Seiten:

Anweisungsfolge	→	*Anweisung* | *Anweisung* ; *Anweisungsfolge*
Funktionsdefinitionsfolge	→	*Funktionsdefinition*
		| *Funktionsdefinition Funktionsdefinitionsfolge*

Die Linksrekursion in der *Ausdruck*-Regel beseitigen wir durch einmaliges Anwenden der Ausdruck Regel (bis auf den rekursiven Fall), was hier zu einer äquivalenten Grammatik führt.

$$
\begin{aligned}
\textit{Ausdruck} \rightarrow\ & \texttt{(}\ \textit{Ausdruck}\ R_{\texttt{)}}\ R_{\texttt{Operator}}\ \textit{Ausdruck} \mid \texttt{Zahl}\ R_{\texttt{Operator}}\ \textit{Ausdruck} \\
& \mid \texttt{Bezeichner}\ R_{\texttt{Operator}}\ \textit{Ausdruck} \mid \texttt{(}\ \textit{Ausdruck}\ R_{\texttt{)}} \\
& \mid \texttt{Bezeichner}\ R_{\texttt{(}}\ \textit{Ausdruck}\ R_{\texttt{)}}\ R_{\texttt{Operator}}\ \textit{Ausdruck} \\
& \mid \texttt{Zahl} \mid \texttt{Bezeichner} \mid \texttt{Bezeichner}\ R_{\texttt{(}}\ \textit{Ausdruck}\ R_{\texttt{)}}
\end{aligned}
$$

Die meisten Regeln haben jetzt bereits ein Terminalsymbol an der ersten Stelle der rechten Seite und sind daher einfach in Greibach-Normalform zu überführen, indem im Rest der Regel Terminale durch neue Nichtterminale ersetzt werden. Danach sind nur die Regeln für *Anweisungsfolge*n und *Funktionsdefinitionsfolge*n nicht in Greibach-Normalform. Diese läßt sich jedoch durch einmaliges Einsetzen der benutzten Regeln erreichen.

$$
\begin{array}{lcl}
\mathit{Ausdruck} & \to & \underline{\texttt{(}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{)}}}\ R_{\underline{\texttt{Operator}}}\ \mathit{Ausdruck} \mid \underline{\texttt{Zahl}}\ R_{\underline{\texttt{Operator}}}\ \mathit{Ausdruck} \\
 & & \mid \underline{\texttt{Bezeichner}}\ R_{\underline{\texttt{Operator}}}\ \mathit{Ausdruck} \mid \underline{\texttt{(}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{)}}} \\
 & & \mid \underline{\texttt{Bezeichner}}\ R_{\underline{\texttt{(}}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{)}}}\ R_{\underline{\texttt{Operator}}}\ \mathit{Ausdruck} \\
 & & \mid \underline{\texttt{Zahl}} \mid \underline{\texttt{Bezeichner}} \mid \underline{\texttt{Bezeichner}}\ R_{\underline{\texttt{(}}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{)}}} \\
\mathit{Anweisung} & \to & \underline{\texttt{if}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{then}}}\ \mathit{Anweisung} \\
\mathit{Anweisung} & \to & \underline{\texttt{if}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{then}}}\ \mathit{Anweisung}\ R_{\underline{\texttt{else}}}\ \mathit{Anweisung} \\
\mathit{Anweisung} & \to & \underline{\texttt{begin}}\ \mathit{Anweisungsfolge}\ \underline{\texttt{end}} \\
\mathit{Anweisung} & \to & \underline{\texttt{Bezeichner}}\ R_{\underline{\texttt{:=}}}\ \mathit{Ausdruck} \\
\mathit{Anweisung} & \to & \underline{\texttt{var}}\ R_{\underline{\texttt{Bezeichner}}} \\
\mathit{Anweisung} & \to & \underline{\texttt{return}}\ \mathit{Ausdruck} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{if}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{then}}}\ \mathit{Anweisung} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{if}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{then}}}\ \mathit{Anweisung}\ R_{\underline{\texttt{else}}}\ \mathit{Anweisung} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{begin}}\ \mathit{Anweisungsfolge}\ \underline{\texttt{end}} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{Bezeichner}}\ R_{\underline{\texttt{:=}}}\ \mathit{Ausdruck} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{var}}\ R_{\underline{\texttt{Bezeichner}}} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{return}}\ \mathit{Ausdruck} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{if}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{then}}}\ \mathit{Anweisung}\ R_{\underline{\texttt{;}}}\ \mathit{Anweisungsfolge} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{if}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{then}}}\ \mathit{Anweisung}\ R_{\underline{\texttt{else}}}\ \mathit{Anweisung}\ R_{\underline{\texttt{;}}}\ \mathit{Anweisungsfolge} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{begin}}\ \mathit{Anweisungsfolge}\ \underline{\texttt{end}}\ R_{\underline{\texttt{;}}}\ \mathit{Anweisungsfolge} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{Bezeichner}}\ R_{\underline{\texttt{:=}}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{;}}}\ \mathit{Anweisungsfolge} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{var}}\ R_{\underline{\texttt{Bezeichner}}}\ R_{\underline{\texttt{;}}}\ \mathit{Anweisungsfolge} \\
\mathit{Anweisungsfolge} & \to & \underline{\texttt{return}}\ \mathit{Ausdruck}\ R_{\underline{\texttt{;}}}\ \mathit{Anweisungsfolge} \\
\mathit{Funktionsdefinition} & \to & \underline{\texttt{function}}\ R_{\underline{\texttt{Bezeichner}}}\ R_{\underline{\texttt{(}}}\ R_{\underline{\texttt{Bezeichner}}}\ R_{\underline{\texttt{)}}} \\
 & & R_{\underline{\texttt{begin}}}\ \mathit{Anweisungsfolge}\ R_{\underline{\texttt{end}}} \\
\mathit{Funktionsdefinitionsfolge} & \to & \underline{\texttt{function}}\ R_{\underline{\texttt{Bezeichner}}}\ R_{\underline{\texttt{(}}}\ R_{\underline{\texttt{Bezeichner}}}\ R_{\underline{\texttt{)}}} \\
 & & R_{\underline{\texttt{begin}}}\ \mathit{Anweisungsfolge}\ R_{\underline{\texttt{end}}} \\
\mathit{Funktionsdefinitionsfolge} & \to & \underline{\texttt{function}}\ R_{\underline{\texttt{Bezeichner}}}\ R_{\underline{\texttt{(}}}\ R_{\underline{\texttt{Bezeichner}}}\ R_{\underline{\texttt{)}}} \\
 & & R_{\underline{\texttt{begin}}}\ \mathit{Anweisungsfolge}\ R_{\underline{\texttt{end}}}\ \mathit{Funktionsdefinitionsfolge} \\
\mathit{Programm} & \to & \underline{\texttt{program}}\ R_{\underline{\texttt{Bezeichner}}}\ \mathit{Funktionsdefinitionsfolge} \\
 & & R_{\underline{\texttt{begin}}}\ \mathit{Anweisungsfolge}\ R_{\underline{\texttt{end}}}\ R_{\underline{\texttt{.}}} \\
R_{\underline{\texttt{)}}} & \to & \underline{\texttt{)}} \\
R_{\underline{\texttt{Operator}}} & \to & \underline{\texttt{Operator}} \\
R_{\underline{\texttt{(}}} & \to & \underline{\texttt{(}} \\
R_{\underline{\texttt{then}}} & \to & \underline{\texttt{then}} \\
R_{\underline{\texttt{else}}} & \to & \underline{\texttt{else}} \\
R_{\underline{\texttt{end}}} & \to & \underline{\texttt{end}} \\
R_{\underline{\texttt{:=}}} & \to & \underline{\texttt{:=}} \\
R_{\underline{\texttt{Bezeichner}}} & \to & \underline{\texttt{Bezeichner}} \\
R_{\underline{\texttt{;}}} & \to & \underline{\texttt{;}} \\
R_{\underline{\texttt{begin}}} & \to & \underline{\texttt{begin}} \\
R_{\underline{\texttt{end}}} & \to & \underline{\texttt{end}} \\
R_{\underline{\texttt{.}}} & \to & \underline{\texttt{.}}
\end{array}
$$

Abbildung 1: Grammatik G in Greibach-Normalform

T21

Seien „Übung“, „quadrat“, „x“ und „print“ Bezeichner, „·“ ein Operator und „5“ eine Zahl. Ist das Programm P ein Wort aus der Sprache $L(G)$?

```
program Übung
function quadrat(x)
begin
    return x · x;
end
begin
    print (quadrat (5) )
end .
```

Abbildung 2: Das Programm P

Lösungsvorschlag

Nein, die letzte Anweisung in einer *Anweisungsfolge* wird nicht durch ein Semikolon beendet. Daher ist das Semikolon hinter der return Anweisung nicht erlaubt.

H21 (10 Punkte)

Überführen Sie G in Chomsky-Normalform.

Lösungsvorschlag

$$
\begin{array}{lcl}
\mathit{Ausdruck} & \rightarrow & X_{(}\ Y_{\mathit{Ausdruck},)} \mid Y_{\mathit{Ausdruck},\mathit{Operator}}\ \mathit{Ausdruck} \mid Y_{\mathit{Bezeichner},(}\ Y_{\mathit{Ausdruck},)} \\
Y_{\mathit{Ausdruck},)} & \rightarrow & \mathit{Ausdruck}\ X_{)} \\
Y_{\mathit{Ausdruck},\mathit{Operator}} & \rightarrow & \mathit{Ausdruck}\ X_{\mathit{Operator}} \\
Y_{\mathit{Bezeichner},(} & \rightarrow & \mathit{Bezeichner}\ X_{(} \\
Y_{\mathit{Bezeichner},)} & \rightarrow & \mathit{Bezeichner}\ X_{)}
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{Anweisung} & \rightarrow & Y_{\mathit{If},\mathit{Ausdruck}}\ Y_{\mathit{Then},\mathit{Anweisung}} \\
\mathit{Anweisungsfolge} & \rightarrow & Y_{\mathit{If},\mathit{Ausdruck}}\ Y_{\mathit{Then},\mathit{Anweisung}} \\
\mathit{Anweisung} & \rightarrow & Y_{\mathit{If},\mathit{Then}}\ Y_{\mathit{Else},\mathit{Anweisung}} \\
\mathit{Anweisungsfolge} & \rightarrow & Y_{\mathit{If},\mathit{Then}}\ Y_{\mathit{Else},\mathit{Anweisung}} \\
Y_{\mathit{If},\mathit{Ausdruck}} & \rightarrow & X_{If}\ \mathit{Ausdruck} \\
Y_{\mathit{Then},\mathit{Anweisung}} & \rightarrow & X_{\mathit{Then}}\ \mathit{Anweisung} \\
Y_{\mathit{Else},\mathit{Anweisung}} & \rightarrow & X_{\mathit{Else}}\ \mathit{Anweisung} \\
Y_{\mathit{If},\mathit{Then}} & \rightarrow & Y_{\mathit{If},\mathit{Ausdruck}}\ Y_{\mathit{Then},\mathit{Anweisung}}
\end{array}
$$

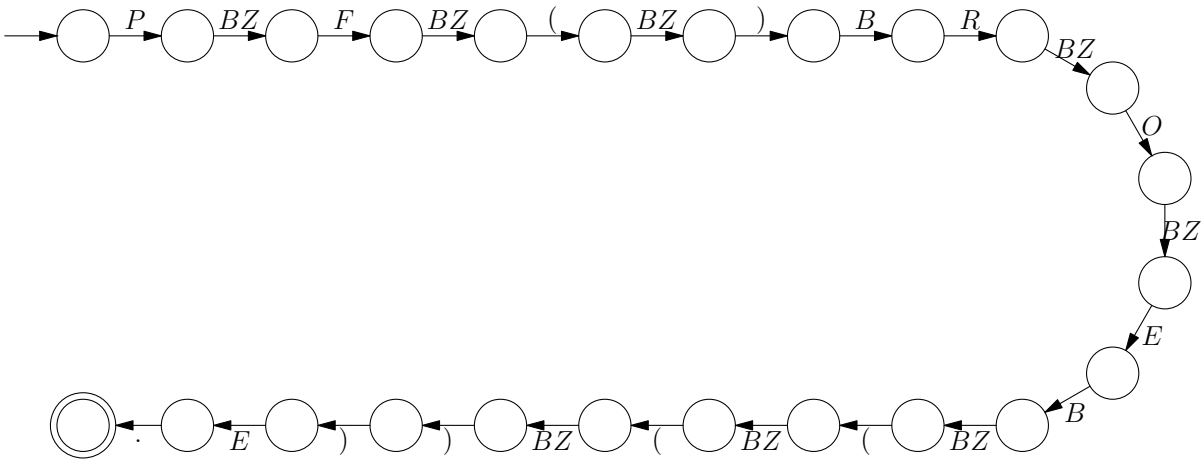
$$
\begin{array}{lcl}
\mathit{Anweisung} & \rightarrow & Y_{\mathit{Begin},\mathit{Anweisungsfolge}}\ X_{\mathit{End}} \\
\mathit{Anweisungsfolge} & \rightarrow & Y_{\mathit{Begin},\mathit{Anweisungsfolge}}\ X_{\mathit{End}} \\
Y_{\mathit{Begin},\mathit{Anweisungsfolge}} & \rightarrow & X_{\mathit{Begin}}\ \mathit{Anweisungsfolge}
\end{array}
$$

$$
\begin{array}{lcl}
\mathit{Anweisung} & \rightarrow & Y_{\mathit{Bezeichner},\mathit{Assignment}}\ \mathit{Ausdruck} \\
\mathit{Anweisungsfolge} & \rightarrow & Y_{\mathit{Bezeichner},\mathit{Assignment}}\ \mathit{Ausdruck} \\
Y_{\mathit{Bezeichner},\mathit{Assignment}} & \rightarrow & X_{\mathit{Bezeichner}}\ X_{\mathit{Assignment}}
\end{array}
$$

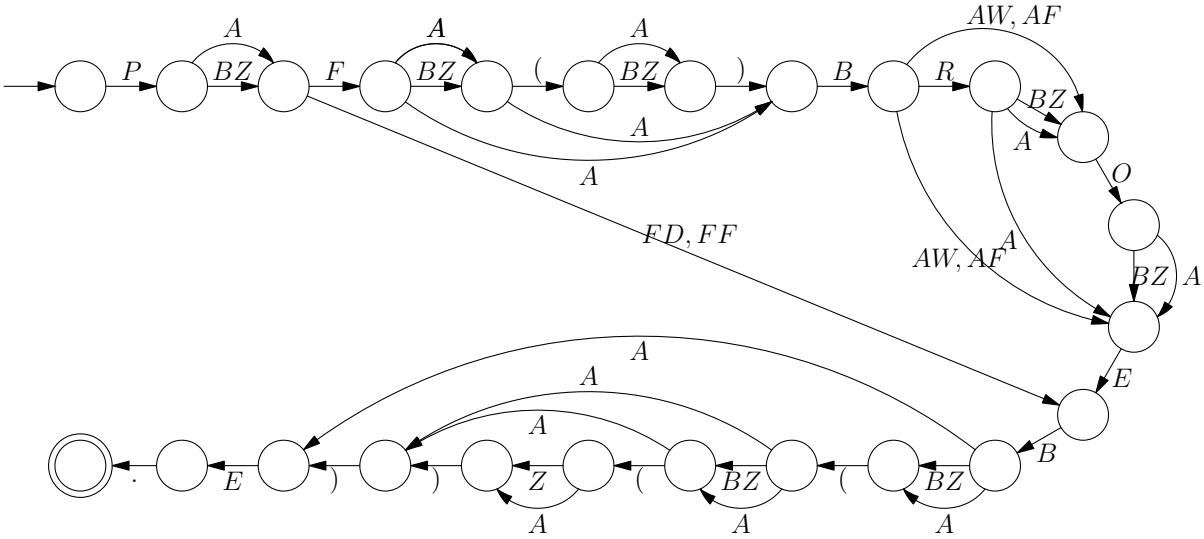
$$
\begin{array}{lcl}
\mathit{Anweisung} & \to & X_{Variable}\ X_{Bezeichner} \\
\mathit{Anweisungsfolge} & \to & X_{Variable}\ X_{Bezeichner} \\
\mathit{Anweisung} & \to & X_{Return}\ \mathit{Ausdruck} \\
\mathit{Anweisungsfolge} & \to & X_{Return}\ \mathit{Ausdruck} \\
\mathit{Anweisungsfolge} & \to & Y_{Anweisungsfolge,;}\ \mathit{Anweisung} \\
Y_{Anweisungsfolge,;} & \to & \mathit{Anweisungsfolge}\ X_{;} \\
 & & \\
\mathit{Funktionsdefinition} & \to & YY_{Function,Bezeichner}\ X_{End} \\
\mathit{Funktionsdefinitionsfolge} & \to & YY_{Function,Bezeichner}\ X_{End} \\
YY_{Function,Bezeichner} & \to & Y_{Function,Bezeichner}\ Y_{Bezeichner,Begin} \\
Y_{Function,Bezeichner} & \to & X_{Function}\ Y_{Bezeichner,(} \\
Y_{Bezeichner,Begin} & \to & Y_{Bezeichner,)}\ Y_{Begin,Anweisungsfolge} \\
Y_{Begin,Anweisungsfolge} & \to & X_{Begin}\ \mathit{Anweisungsfolge} \\
 & & \\
\mathit{Funktionsdefinitionsfolge} & \to & \mathit{Funktionsdefinitionsfolge}\ \mathit{Funktionsdefinition} \\
\mathit{Programm} & \to & YY_{Program,Bezeichner}\ YY_{Begin,End} \\
YY_{Program,Bezeichner} & \to & X_{Program}\ Y_{Bezeichner,Funktionsdefinitionsfolge} \\
YY_{Begin,End} & \to & Y_{Begin,Anweisungsfolge}\ Y_{End,.} \\
Y_{Bezeichner,Funktionsdefinitionsfolge} & \to & X_{Bezeichner}\ \mathit{Funktionsdefinitionsfolge} \\
Y_{End,.} & \to & X_{End}\ X_{.} \\
X_{Operator} & \to & \underline{\texttt{Operator}} \\
X_{Bezeichner} & \to & \underline{\texttt{Bezeichner}} \\
\mathit{Ausdruck} & \to & \underline{\texttt{Bezeichner}} \\
X_{Zahl} & \to & \underline{\texttt{Zahl}} \\
\mathit{Ausdruck} & \to & \underline{\texttt{Zahl}} \\
X_{)} & \to & \underline{\texttt{)}} \\
X_{(} & \to & \underline{\texttt{(}} \\
X_{;} & \to & \underline{\texttt{;}} \\
X_{.} & \to & \underline{\texttt{.}} \\
X_{Function} & \to & \underline{\texttt{function}} \\
X_{If} & \to & \underline{\texttt{if}} \\
X_{Then} & \to & \underline{\texttt{then}} \\
X_{Else} & \to & \underline{\texttt{else}} \\
X_{Begin} & \to & \underline{\texttt{begin}} \\
X_{End} & \to & \underline{\texttt{end}} \\
X_{Return} & \to & \underline{\texttt{return}} \\
X_{Variable} & \to & \underline{\texttt{var}} \\
X_{Assignment} & \to & \underline{\texttt{:=}} \\
X_{Program} & \to & \underline{\texttt{program}}
\end{array}
$$

H22 (10 Punkte)
Sei P' gleich P, allerdings ohne das Semikolon. Bestimmen Sie $pre^*_G(P')$.

Lösungsvorschlag
Im folgenden verwenden wir die Abkürzungen P für program, BZ für Bezeichner, A für Ausdruck, B für begin, E für end, O für Operator, Z für Zahl, R für return, AW für Anweisung, AF für Anweisungsfolge, FD für Funktionsdefinition und FF für Funktionsfolge. Wir beginnen mit dem Automaten



und erhalten nach Ergänzen aller Übergänge das Resultat



Also ist $pre^*_G(P') = \emptyset$, da wir den letzten Begin-End Block nicht in eine Anweisung überführen können.

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 10
Emmes—Kneis—Langer—Rossmanith 8.7.2009

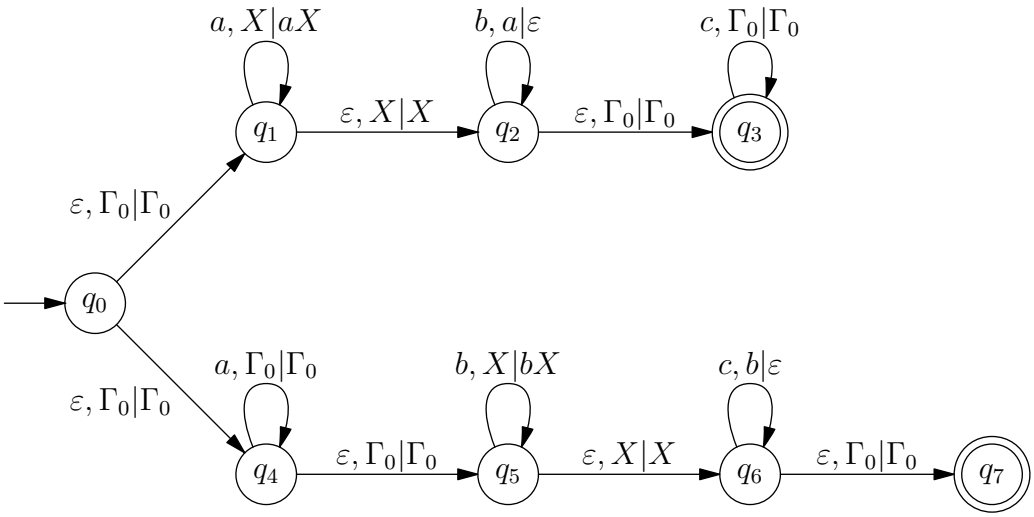
Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

T22

Es sei $L = \{ a^i b^j c^k \mid i = j \text{ oder } j = k \}$. Geben Sie einen nichtdeterministischen Kellerautomaten an, der L akzeptiert.

Lösungsvorschlag

Der Kellerautomat rät erst nichtdeterministisch, ob $i = j$ oder $j = k$ gilt. Danach wird der Stack zum Zählen den entsprechenden Symbole genutzt.



T23

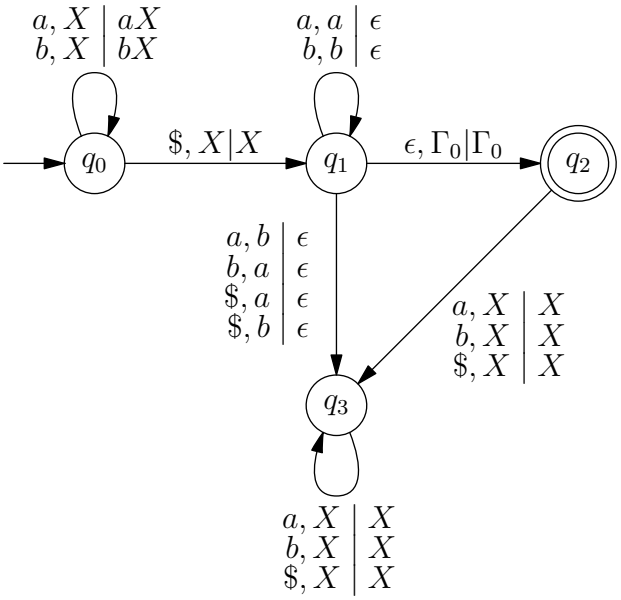
Erstellen Sie einen deterministischen Kellerautomaten für folgende Sprache:

$$\{\$, a, b\}^* \setminus \{ w\$w^R \mid w \in \{a, b\}^* \}$$

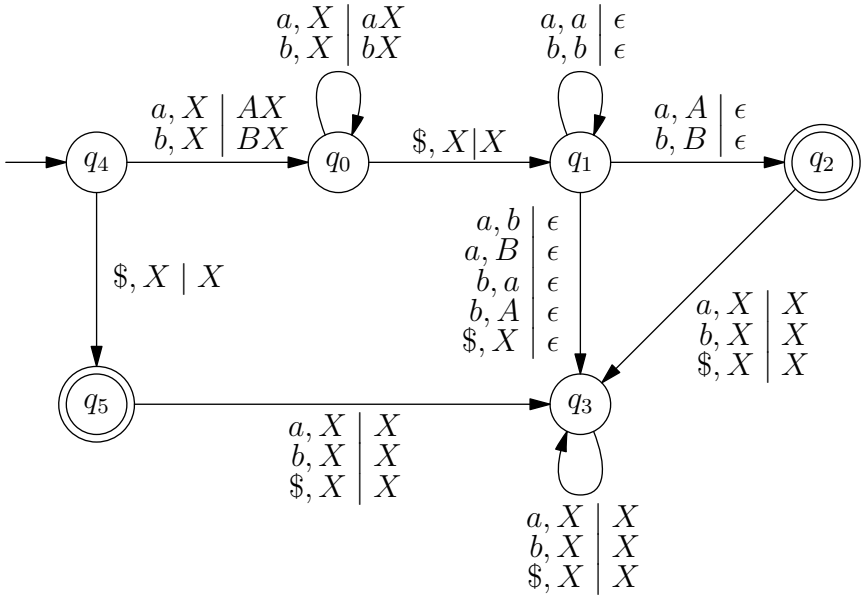
Lösungsvorschlag

Der folgende Automat erkennt die Sprache

$$\{ w\$w^R \mid w \in \{a, b\}^* \}.$$



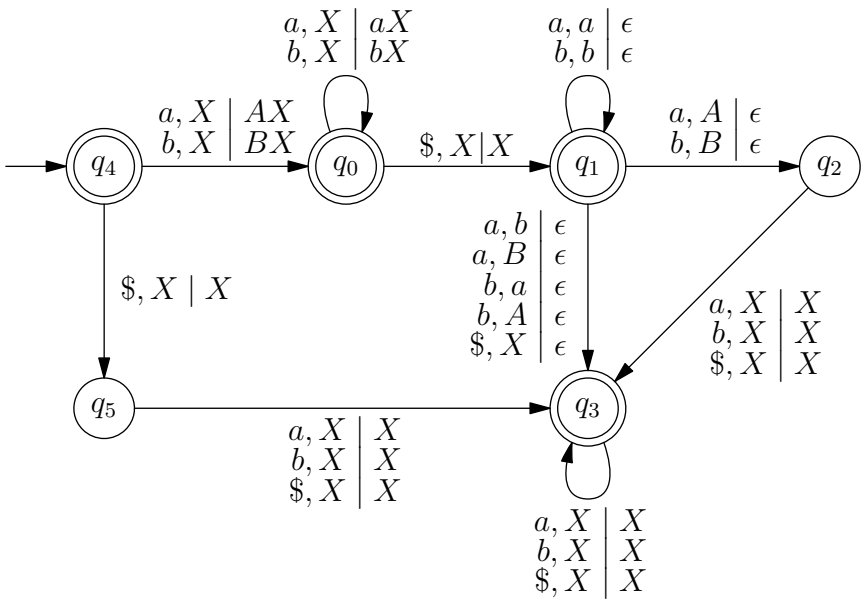
Der Kellerautomat ist zwar deterministisch, enthält aber eine ϵ-Transitionen. Einfaches Tauschen der akzeptierenden Zustände führt daher leider nicht zu einem korrekten Ergebnis. Wir entfernen daher erst die ϵ-Transition, indem wir das erste Symbol auf dem Stack anders notieren als die kommenden Symbole (und daß Wort \$ gesondert behandeln).



Dieser Automat ist deterministisch, enthält keine ϵ-Transitionen und blockiert auch nicht. Letzteres folgt aus der Tatsache, daß alle Zustände außer q_1 vollständig sind und q_1 nur blockieren könnte, falls das Kellerbodensymbol auf dem Stack liegen würde. Dies ist aber in Zustand q_1 nicht möglich, da vorher auf jeden Fall ein A oder ein B auf den Stack geschrieben wurde. Einfaches Austauschen der akzeptierenden Zustände liefert somit einen Automat, der

$$\{\$, a, b\}^* \setminus \{\, w\$w^R \mid w \in \{a, b\}^* \,\}$$

akzeptiert.



H23 (10 Punkte)
Geben Sie eine kontextfreie Grammatik für die Sprache $\{a, b\}^* \setminus \{ ww^R \mid w \in \{a, b\}^* \}$ an.

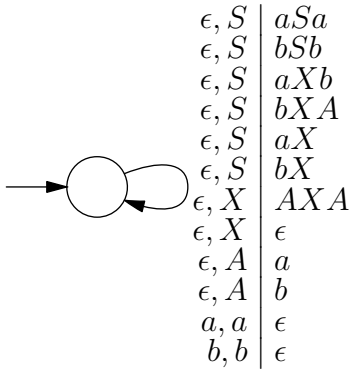
Lösungsvorschlag
Offensichtlich ist ein Wort u nicht in ww^R, wenn es entweder ungerade Länge hat oder es gilt entweder $u = xazz'bx$ oder $u = xbzz'ax^R$ für $x, z, z' \in \{a, b\}^*$ mit $|z| = |z'|$.
Wir benutzen das Hilfssymbol X, das alle Wörte gerade Länge produziert. Wörter ungerader Länge erzeugen wir aus S mittels aX bzw bX. Für Wörter gerader Länge erzeugt unsere Gramatik zuerst xSx^R und anschließend entweder $xaXbx$ oder $xbXax$. Mittels X kann dann jedes Wort in $\{a, b\}^* \setminus \{ ww^R \mid w \in \{a, b\}^* \}$ erzeugt werden.

$$\begin{array}{lcl} S & \rightarrow & aSa \mid bSb \mid aXb \mid bXa \mid aX \mid bX \\ X & \rightarrow & AXA \mid \varepsilon \\ A & \rightarrow & a \mid b \end{array}$$

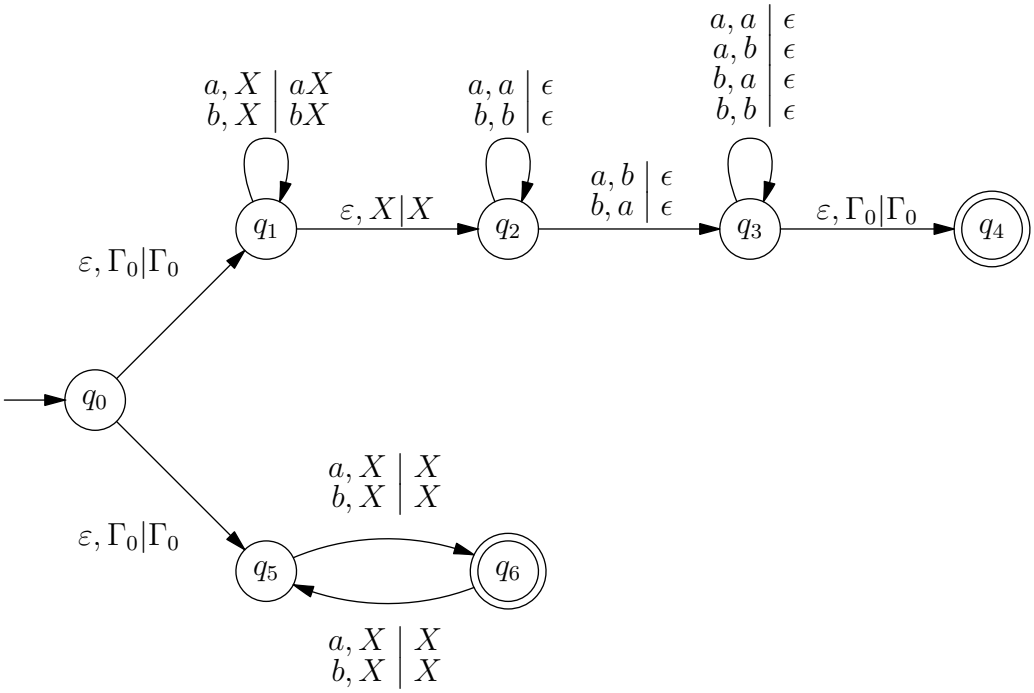
H24 (10 Punkte)
Wie sieht ein nichtdeterministischer Kellerautomat aus, der die Sprache aus Aufgabe H23 erkennt?

Lösungsvorschlag
Da wir die Gramatik der Sprache schon in H24 konstruiert haben, können wir den zu-

gehörigen Automaten leicht mit dem Verfahren aus der Vorlesung konstruieren:



Alternativ kann auch leicht ein Automat direkt angegeben werden. Wir raten zuerst, ob $u \in \{a, b\}^* \setminus \{ ww^R \mid w \in \{a, b\}^* \}$ gerade oder ungerade Länge hat. Im ersten Fall lesen wir danach nichtdeterministisch die erste Hälfte des Wortes und garantieren anschließend daß sie ungleich der zweiten Hälfte ist. Im zweiten Fall garantieren wir einfach daß u ungerade Länge hat.



H25 (10 Punkte)
Es sei $L = \{ w\$w\$w \mid w \in \{a, b\}^* \}$. Benutzen Sie das Pumping-Lemma um zu zeigen, daß L nicht durch eine kontextfreie Grammatik erzeugt werden kann.

Lösungsvorschlag
Angenommen L sei regulär. Dann existiert ein n sodaß jedes Wort $w \in L$ mit $|w| \geq n$ in $w = uvwxy$ zerlegt werden kann mit

1. $|vwx| \leq n$,

2. $|vx| \geq 1$ und

3. $uv^i wx^i y \in L$ für alle $i \in \mathbf{N}_0$.

Sei $w = a^n\$a^n\$a^n$. Offensichtlich gilt $w \in L$. Falls vx mindestens ein \$ enthält, so ist uwy offensichtlich nicht in L enthalten, da dieses Wort noch höchstens ein \$ enthält.

Wir können also davon ausgehen, daß vx kein \$ enthält. Für jede solche Zerlegung die (1) und (2) erfüllt, gilt aber daß vwx höchstens zwei der getrennten Blöcke von as überlappt, da $|vwx| \leq n$. Damit ist aber $uv^2wx^2y \notin L$, da mindestens einer der drei Blöcke von as genau die Länge n hat, während mindestens einer der beiden anderen Blöcke mindestens die Länge $a + 1$ hat.

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 11
Emmes—Kneis—Langer—Rossmanith 15.7.2009

Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

T24

Eine Grammatik G ist monoton, falls keine ihrer Regeln auf der linken Seite mehr Symbole als auf der rechten Seite hat. Um außerdem Sprachen, die das leere Wort enthalten, erzeugen zu können, sei die Regel $S \to \varepsilon$ erlaubt, falls S sonst auf keiner rechten Seite vorkommt. Beispielsweise ist die folgende Grammatik monoton aber nicht kontextsensitiv:

$$\begin{array}{lcl} S & \to & \varepsilon \mid aAb \\ aAb & \to & aAbc \mid ddd \end{array}$$

Zeigen Sie: Die Menge der durch monotone Grammatiken erzeugbaren Sprachen entspricht genau der Menge der durch kontextsensitive Grammatiken erzeugbaren Sprachen.

Lösungsvorschlag

Eine Regel ist kontextsensitiv, falls sie die Form $\alpha A\beta \to \alpha\gamma\beta$ mit $A \in N$, $\alpha, \beta, \gamma \in (N \cup T)^*$ und $\gamma \neq \varepsilon$ hat. Jede kontextsensitive Regel ist ebenfalls monoton. Somit bleibt nur die andere Richtung zu zeigen.

Wir transformieren jede monotone Regel in eine Menge von kontextsensitiven Regeln. Hierbei achten wir darauf, daß keine zusätzlichen Terminalwörter abgeleitet werden können. Beispielsweise läßt sich die monotone Regel $A_1A_2 \to B_1B_2$ in folgende kontextsensitive Regeln übersetzen:

$$\begin{array}{lcl} A_1A_2 & \to & B_1'A_2 \\ B_1'A_2 & \to & B_1'B_2' \\ B_1'B_2' & \to & B_1B_2' \\ B_1B_2' & \to & B_1B_2 \end{array}$$

Allgemein sei $A_1 \dots A_n \to B_1 \dots B_m$ mit $n \leq m$ eine kontextsensitive Regel. Wir führen für diese Regel frische Nichtterminalsymbole $B_1', \dots, B_m'$ ein um die monotone Regel in mehreren kontextsensitiven Schritten auszuwerten:

$$\begin{array}{lcl} A_1A_2 \dots A_n & \to & B_1'A_2 \dots A_n \\ B_1'A_2A_3 \dots A_n & \to & B_1'B_2'A_3 \dots A_n \\ & \vdots & \\ B_1' \dots B_{n-1}'A_n & \to & B_1' \dots B_n' \dots B_m' \end{array}$$

Um die B'-Symbole zurück in B-Symbole zu übersetzen, verfahren wir auf die selbe Weise.

$$\begin{array}{lcl} B_1' \dots B_m' & \to & B_1B_2' \dots B_m' \\ B_1B_2' \dots B_m' & \to & B_1B_2B_3' \dots B_m' \\ & \vdots & \\ B_1 \dots B_{m-1}B_m' & \to & B_1 \dots B_m \end{array}$$

Nun zeigen wir, daß durch die neuen Regeln keine zusätzlichen Terminalwörter ableitbar sind. Durch die Benutzung von frischen Nichtterminalsymbolen sind die neuen Regeln nur dann anwendbar, falls die Regel $A_1A_2 \dots A_n \to B'_1A_2 \dots A_n$ verwendet wird. Wir betrachten also ein Wort der Form $\alpha A_1 \dots A_n\beta$. In diesem Wort werden durch die neuen Regeln von links nach rechts A_i-Symbole durch B'_i-Symbole ersetzt. Da die B'_i-Symbole nicht außerhalb der neuen Regeln vorkommen, kann eine kritischen Ersetzung nicht im B'-Teil eines Wortes stattfinden. Tritt sie im α- oder β-Teil des Wortes auf, wäre sie ohne die neuen Regeln ebenfalls möglich gewesen. Es bleibt der Fall zu betrachten, daß eine Ersetzung im noch nicht ersetzten $A_i \dots A_n\beta$-Teil auftritt. Auch eine solche Ersetzung wäre im ursprünglichen System möglich gewesen, da $A_i \dots A_n\beta$ ein Postfix des ursprünglichen Wortes $\alpha A_1 \dots A_n\beta$ ist.
Somit sind bis zur Ableitung des Wortes $\alpha B'_1 \dots B'_m\beta$ keine unerwünschten Ableitungen möglich. Nun werden schrittweise die B'_i-Symbole durch B_i-Symbole ersetzt. Hierbei sind ebenfalls keine kritischen Ableitungen möglich: Falls eine Ableitung möglich ist, tritt sie aufgrund der frischen Symbole in einem Teil des Wortes ohne B'_i-Symbole auf. Dieser Teil existiert dann allerdings genau so nach der Anwendung der ursprünglichen $A_1 \dots A_n \to B_1 \dots B_m$ Regel und so wäre die Ableitung in der ursprünglichen Grammatik ebenfalls möglich.

T25

In der Vorlesung wurde gezeigt, daß Chomsky-0-Sprachen nicht entscheidbar sind. Hierzu wurde das Post'sche Korrespondenzproblem auf das Wortproblem einer unbeschränkten Grammatik zurückgeführt.

$$I := \left\{ \boxed{\begin{matrix} u_1 \\ \\ v_1 \end{matrix}}, \boxed{\begin{matrix} u_2 \\ \\ v_2 \end{matrix}}, \dots, \boxed{\begin{matrix} u_n \\ \\ v_n \end{matrix}} \right\}$$

$$\begin{array}{lcl} S & \to & u_1Mv_1^R \mid \dots \mid u_nMv_n^R \\ M & \to & u_1Mv_1^R \mid \dots \mid u_nMv_n^R \mid D \\ & \vdots & \end{array}$$

Ergänzen Sie obige Grammatik aus dem Beweis, daß das Leerheitsproblem schon für kontextsensitive Sprachen unentscheidbar ist, welcher in der Vorlesung nicht zu Ende geführt wurde.

Lösungsvorschlag

Die S- und M-Regeln sind monoton. Wenn durch die S- und M-Regeln also ein Wort der Form $\alpha D\alpha^R$ erzeugbar ist (mit $\alpha \in N^*$), folgt daraus, daß das ursprüngliche Post'sche Korrespondenzproblem eine Lösung hat. Wir erzeugen nun eine monotone Grammatik, die überprüft ob ein Wort diese Form hat und dann akzeptiert. Dies stellt einen Widerspruch zur Unentscheidbarkeit des PKPs dar.
Die Idee der Grammatik ist Wörter der Form $\alpha D\alpha^R$ in Terminalwörter zu überführen. Hierzu wird, ausgehend vom D-Symbol in der Mitte, jeweils das nächste linke und rechte Nichtterminalsymbol (falls diese gleich sind) durch ein Terminal ersetzt. O.b.d.A. gehen wir von $\Sigma = \{0, 1\}$ für das PKP aus. Diese Symbole definieren wir in unserer Grammatik als Nichtterminale.

Wir starten mit der Suche nach dem nächsten Nichtterminalsymbol links vom D-Symbol. Hierfür verwenden wir den Marker L.

$$\begin{array}{lcl} D & \to & Lx \\ xL & \to & Lx \end{array}$$

Diese Regeln lassen sich so lange anwenden, bis das L-Symbol direkt nach einem Nichtterminal steht. Nun merken wir uns dieses Symbol, löschen es und suchen nach einem passenden Nichtterminal auf der rechten Seite:

$$\begin{array}{lcl} 0L & \to & xR_0 \\ 1L & \to & xR_1 \\ R_0x & \to & xR_0 \\ R_1x & \to & xR_1 \end{array}$$

Wenn wir dort angekommen sind überprüfen wir, ob das Symbol stimmt und fahren mit dem nächsten Symbol auf der linken Seite fort:

$$\begin{array}{lcl} R_00 & \to & Lx \\ R_11 & \to & Lx \end{array}$$

So werden schrittweise alle passenden Symbole durch das Terminal x ersetzt. Letztendlich müssen wir zulassen, daß auch L durch ein Terminalsymbol ersetzt wird, damit wir im letzten Schritt ein Terminalwort erzeugen:

$$L \to x$$

T26
Linkslineare Grammatiken erzeugen genau die regulären Sprachen. Zeigen Sie, daß dies auch für die rechtslinearen Grammatiken gilt.

Lösungsvorschlag
Transformiere jede rechtslineare Regel der Form $A \to bC$ in die linkslineare Form $A \to Cb$. Die Wörter der von dieser Grammatik erzeugten Sprache sind die umgedrehten Wörter der ursprünglichen Sprache. Da reguläre Sprachen unter Umdrehen der Wörter abgeschlossen sind, gilt die Aussage.

H26 (10 Punkte)
Betrachten Sie Grammatiken, die nur rechtslineare und linkslineare Regeln enthalten. Beweisen oder widerlegen Sie: Diese Grammatiken erzeugen genau die regulären Sprachen.

Lösungsvorschlag
Grammatiken dieser Art erzeugen mehr als die regulären Sprachen. Beispielsweise läßt sich die nichtreguläre Sprache $\{ a^n c b^n \mid n \in \mathbf{N} \}$ mit folgender Grammatik aus rechts- und linkslinearen Regeln erzeugen.

$$\begin{array}{lcl} S & \to & c \\ S & \to & aA \\ A & \to & Sb \end{array}$$

H27 (10 Punkte)

Gegeben sei die Chomsky-0-Grammatik mit den Produktionen $\{S \to bbbbb, bb \to aaa, bab \to a, baa \to bab\}$. Ist das Wort aaa durch die Grammatik erzeugbar?

Lösungsvorschlag

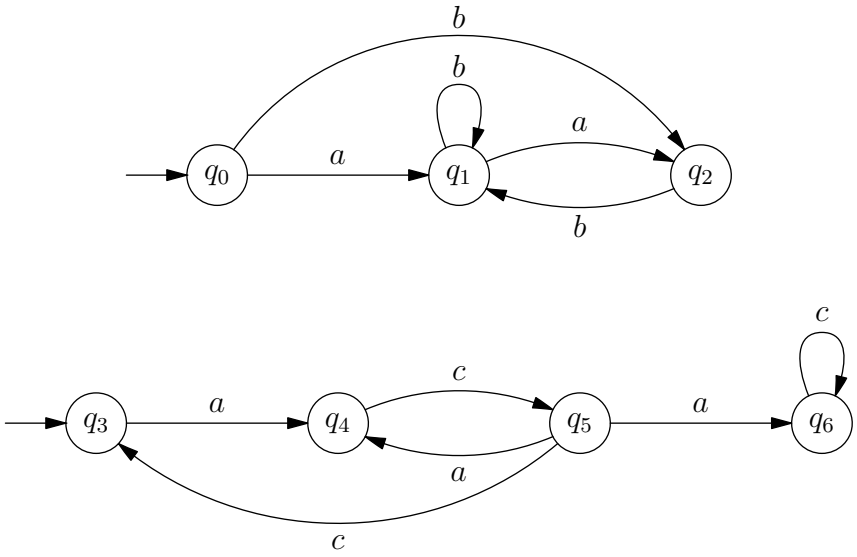
$$
\begin{aligned}
\underline{S} &\Rightarrow b\underline{bbb}b \\
&\Rightarrow baaa\underline{bb} \\
&\Rightarrow \underline{baa}aaaa \\
&\Rightarrow ba\underline{baa}aa \\
&\Rightarrow baba\underline{baa} \\
&\Rightarrow \underline{bab}abab \\
&\Rightarrow aa\underline{bab} \\
&\Rightarrow aaa
\end{aligned}
$$

RWTH Aachen **SS 2009**
Lehrgebiet Theoretische Informatik Übungsblatt 12
Emmes—Kneis—Langer—Rossmanith 22.7.2009

Übung zur Vorlesung Formale Systeme, Automaten und Prozesse

T27

Gegeben seien die NFAs M_1 und M_2:



Bestimmen Sie:

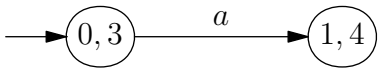
a) $M_1 \times M_2$

b) $M_1 \circ M_2$

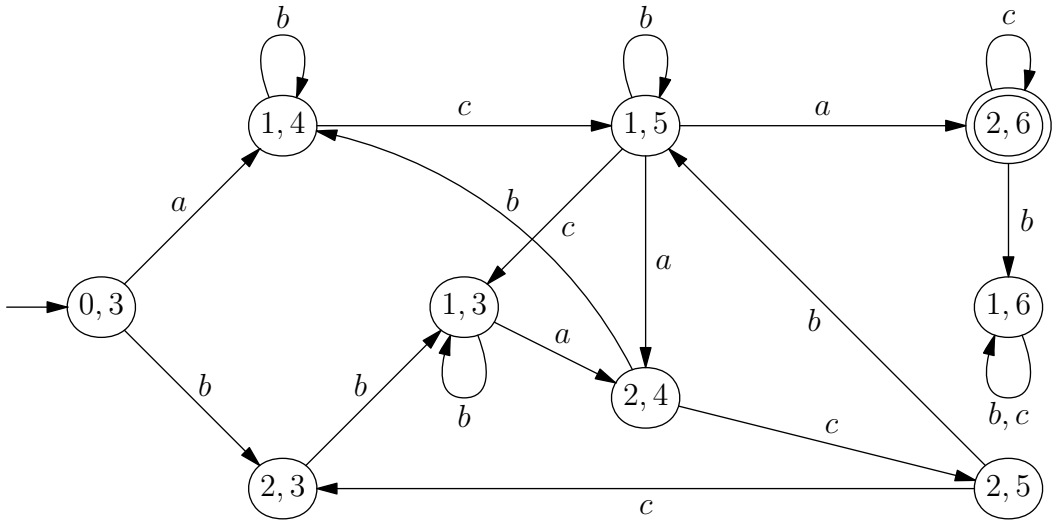
c) $M_1 \sqcup M_2$

Lösungsvorschlag

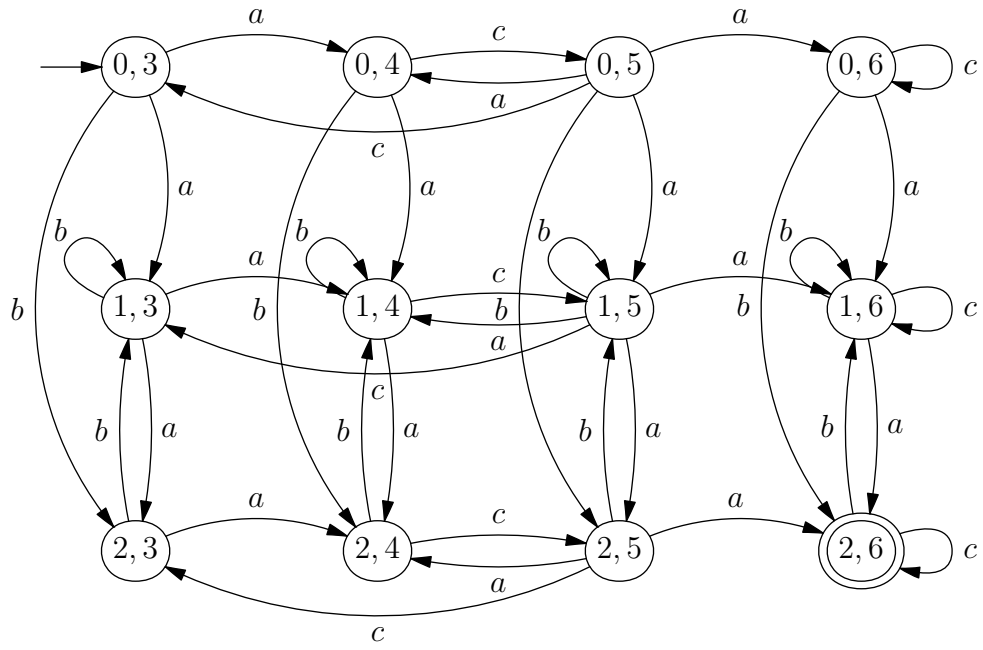
a)



b)



c)



T28

Gegeben sind die folgenden Programme:

Programm P_1:

```
while (x = 0)
    x := 1;
print;
```

Programm P_2:

```
while (x = 1)
    x := 0;
print;
```

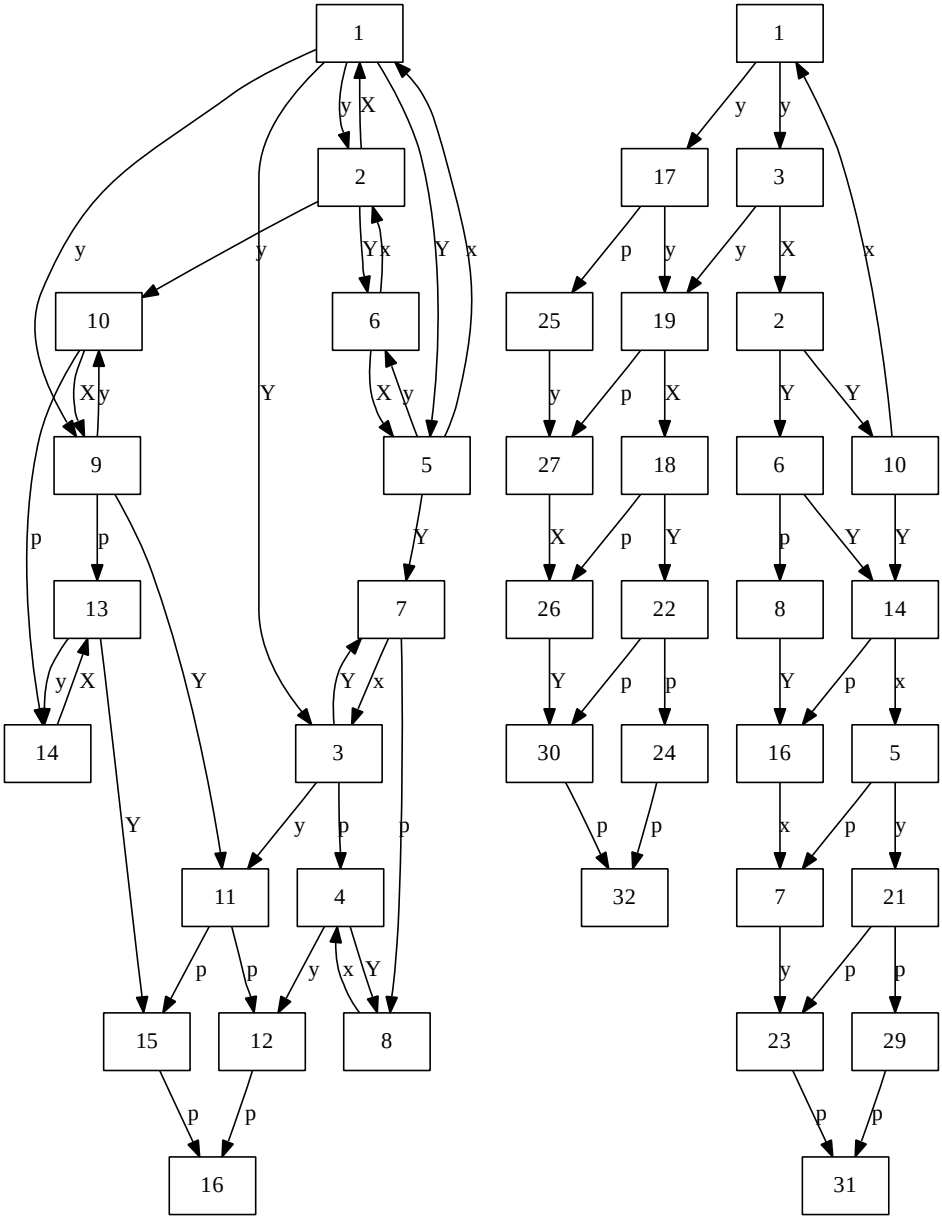
a) Modellieren Sie die Kontrollstruktur der Programme P_1 und P_2 als NFAs M_1 und M_2.

b) Bestimmen Sie $M_1 \sqcup\!\sqcup M_2$.

c) Bestimmen Sie $(M_1 \sqcup\!\sqcup M_2) \circ B_x$. Hierbei ist B_x die Modellierung der bool'schen Variable x als NFA.

d) Wenn P_1 und P_2 parallel ausgeführt werden, kann es dann vorkommen, daß `print` in einem der beiden Programme nicht ausgeführt wird?

Lösungsvorschlag



Aufgabe 1 (10 Punkte)
Gegeben ist der Monoid $M = (\mathbf{Z}, +)$.

1. Ist $\{3, 5, 7\}$ ein Erzeugendensystem für M?
2. Ist $\{-4, 7\}$ ein Erzeugendensystem für M?

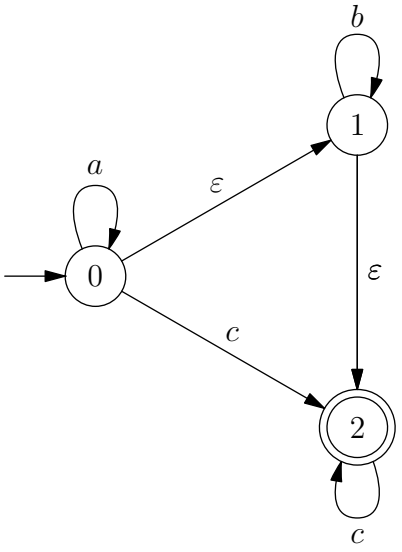
Aufgabe 1 (10 Punkte)
Gegeben ist der Monoid $M = (\mathbf{N}, \cdot)$.
($\mathbf{N} = \{1, 2, 3, \dots\}$)

1. Was ist ein freies Erzeugendensystem?
2. Existiert ein freies Erzeugendensystem für M?

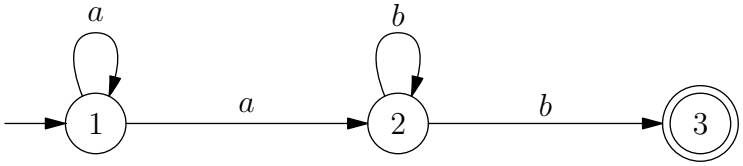
Aufgabe 2 (10 Punkte)
Die Sprache $L = \{ a^n b^m \mid m \geq n \geq 0 \}$ ist nicht regulär. Zeigen Sie damit, daß auch die Sprache $L_1 = \{ a^n b^m c^k \mid n + k = m \}$ nicht regulär ist.

Aufgabe 2 (10 Punkte)
L sei die durch den regulären Ausdruck $(a + b)^* abab$ erzeugte Sprache. Geben Sie einen DFA an, der L erkennt.

Aufgabe 3 (10 Punkte)
Wandeln Sie den folgenden NFA in einen NFA *ohne* Epsilon-Übergänge um.



Aufgabe 3 (10 Punkte)
Führen Sie die Potenzmengenkonstruktion auf dem folgenden NFA durch.



Aufgabe 4 (10 Punkte)
Es sei $L = 0^*1^*$. Geben Sie die Äquivalenzklassen von $\equiv_L$ an (ohne Beweis).

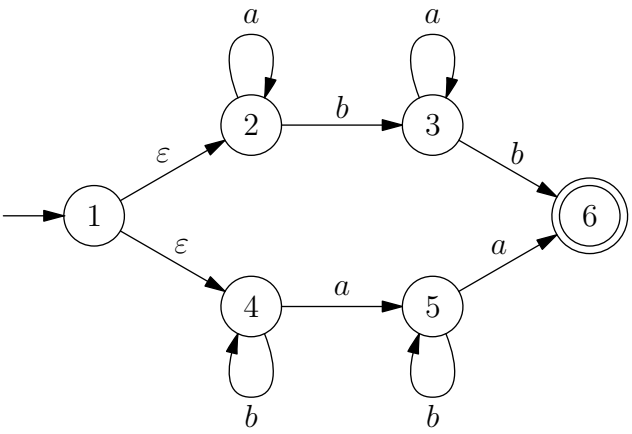
Aufgabe 4 (10 Punkte)
Gegeben sind die folgenden Sprachen:

1. $L_1 = \{0,1\}^+$
2. $L_2 = \emptyset$

Geben Sie die Äquivalenzklassen von $\equiv_{L_1}$ und $\equiv_{L_2}$ an (ohne Beweis).

Aufgabe 5 (10 Punkte)
Konstruieren Sie einen NFA mit ϵ-Übergängen, der die Sprache $a(b+c)^*$ akzeptiert.

Aufgabe 5 (10 Punkte)
Verwandeln Sie folgenden NFA mit ϵ-Übergängen in einen gewöhnlichen NFA.



Aufgabe 6 (10 Punkte)
Ist folgende Sprache regulär? Beweisen Sie Ihre Antwort.

$$\{\, a^i b^j c^k \mid i \geq k \text{ oder } j \geq k \,\}$$

Aufgabe 6 (10 Punkte)
Ist die Sprache aller Palindrome aus $\{0,1\}^*$ regulär? Beweisen Sie Ihre Antwort.
Ein Wort ist ein Palindrom, wenn es rückwärtsgelesen das gleiche Wort ergibt, wie vorwärtsgelesen. Ein Wort w ist also ein Palindrom, wenn $w = w^R$ gilt.

Aufgabe 7 (10 Punkte)
Gegeben sei folgende CFG:

$$\begin{aligned} S &\to S + P \mid S - P \mid P \\ P &\to P \cdot A \mid P/A \mid A \\ A &\to (S) \mid \mathbf{z} \end{aligned}$$

Zeichnen Sie einen Ableitungsbaum für das Wort $\mathbf{z} \cdot \mathbf{z} - \mathbf{z} \cdot (\mathbf{z} + \mathbf{z})$.

Aufgabe 7 (10 Punkte)
Gegeben sei folgende CFG:

$$S \to P + S \mid P - S \mid P$$
$$P \to A \cdot P \mid A/P \mid A$$
$$A \to (S) \mid \mathbf{z}$$

Zeichnen Sie einen Ableitungsbaum für das Wort $\mathbf{z} \cdot \mathbf{z} - \mathbf{z} \cdot (\mathbf{z} + \mathbf{z})$.

Aufgabe 9 (10 Punkte)
Beschreiben Sie ein allgemeines Verfahren, mit dem man die unproduktiven Symbole einer CFG bestimmen kann.
Welche Symbole der folgenden CFG sind unproduktiv?

$$S \to ABC \mid a$$
$$A \to BaC \mid ACb$$
$$B \to Ab \mid BaC$$
$$C \to SaSb \mid CaA$$

Aufgabe 9 (10 Punkte)
Beschreiben Sie ein allgemeines Verfahren, mit dem man die nullierbaren Symbole einer CFG bestimmen kann.
Welche Symbole der folgenden CFG sind nullierbar?

$$S \to ABC \mid \varepsilon$$
$$A \to BaC \mid AC$$
$$B \to SC \mid aAB$$
$$C \to BS \mid AS \mid BC$$

Aufgabe 10 (10 Punkte)
Geben Sie eine CFG für die Sprache $\{ w \in \{a, b\}^* \mid |w|_a > |w|_b \}$ an.

Aufgabe 10 (10 Punkte)
Geben Sie eine CFG für die Sprache $\{ w \in \{a, b\}^* \mid |w|_a = 2|w|_b \}$ an.

Aufgabe 11 (10 Punkte)
Ist folgende Grammatik eindeutig? Falls nein, geben Sie zwei unterschiedliche Ableitungen für ein Wort der erzeugten Sprache an.

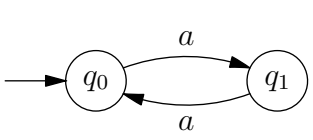
$$\begin{aligned} S &\to SaSBS \\ &\mid SbSaS \\ &\mid \varepsilon \end{aligned}$$

Aufgabe 11 (10 Punkte)

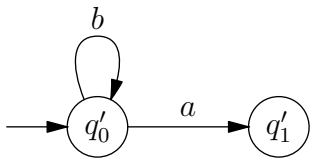
Geben Sie einen Kellerautomaten an, der die Sprache der Wörter mit gleich vielen a und b Symbolen $\{ w \in \{a, b\}^* \mid |w|_a = |w|_b \}$ erkennt.

Aufgabe 12 (10 Punkte)

M_1:



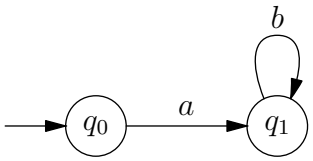
M_2:



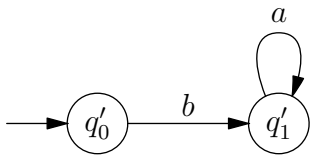
Bestimmen Sie das synchronisierte Produkt $M_1 \circ M_2$.

Aufgabe 12 (10 Punkte)

M_1:



M_2:



Bestimmen Sie das unsynchronisierte Produkt $M_1 \sqcup M_2$.

RWTH Aachen
Lehrgebiet Theoretische Informatik
Mölle—Richter—Rossmanith

SoSe 2005
Probeklausur
24.06.2005

Übung zur Vorlesung Automatentheorie und Formale Sprachen

Aufgabe 1 (10 Punkte)

Welche der folgenden Sprachen sind regulär? Beweisen Sie jeweils Ihre Behauptung!

(a) die Sprache aller deutschen Wörter, die Sie auf diesem Aufgabenblatt finden

(b) $L(a^*(b+c)a^*) \setminus L(a^+(b^+ + c^+)a^+)$

(c) $\{a^n \mid n \text{ ist eine Quadratzahl}\}$

(d) $\{w \in \{a,b\}^* \mid w \text{ enthält mindestens doppelt so viele } a \text{ wie } b\}$

Aufgabe 2 (10 Punkte)

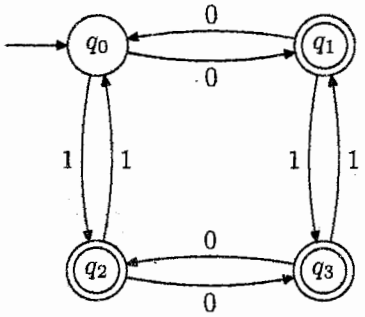
Erzeugen Sie einen NFA für $a^*(bb+cc)^*a^*$, wandeln Sie ihn in einen DFA um und minimieren sie diesen!

Aufgabe 3 (10 Punkte)

Entwerfen Sie eine Grammatik, die genau die Palindrome über dem Alphabet $\{a,b\}$ erzeugt. Beweisen Sie formal und ausführlich, daß Ihre Grammatik das Gewünschte leistet.

Aufgabe 4 (10 Punkte)

Transformieren Sie den folgenden Automaten durch Zustandselimination in einen regulären Ausdruck:



Aufgabe 5 (10 Punkte)

Gegeben ist folgende Grammatik G

$$S \to AB \mid ASA$$
$$A \to aAB \mid a$$
$$B \to bSa \mid b$$

und die Sprache $L = (aba)^*$.

a) Enthält L ein Wort aus $L(G)$?

b) Geben Sie einen regulären Ausdruck für $pre^*(L) \cap \{a,b\}^*$ an.

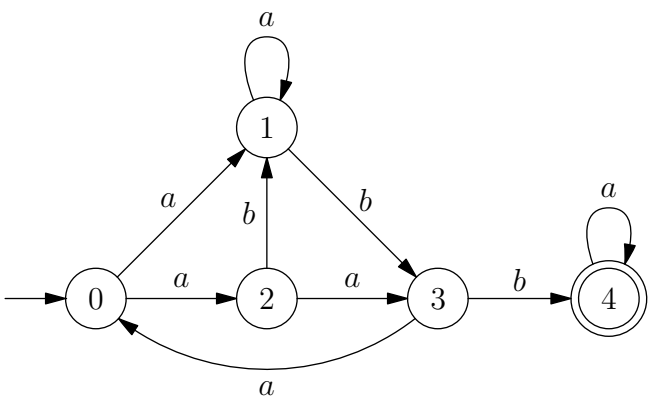
c) Gibt es Satzformen α, β und γ, so daß $\gamma \in L$ und $\alpha aa\beta \stackrel{*}{\Rightarrow} \gamma$?

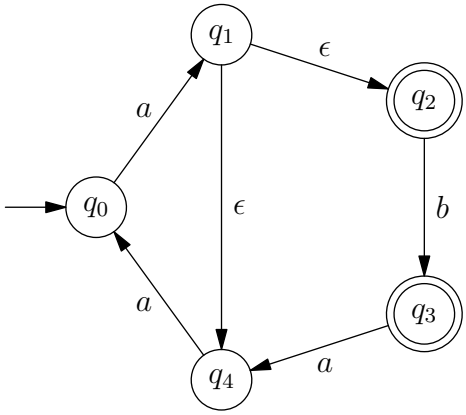
Lösen sie diese Aufgaben, indem Sie zunächst einen NFA für $pre^*(L)$ aufstellen.

Rückgabe:
☐ Zimmer (Mo 0815) ☒ Wright (Mo 1000) ☐ Lambertz (Mo 1145)
☐ Vontin (Mo 1400) ☐ Rabinovich (Di 1545) ☐ Langer (Di 1545) ☐ Drobny (Mi 1145)

Formale Systeme, Automaten und Prozesse

Aufgabe 1 (10 Punkte)
Konstruieren Sie reguläre Ausdrücke für die Sprachen, die von folgenden NFAs akzeptiert werden:



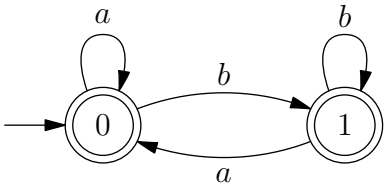


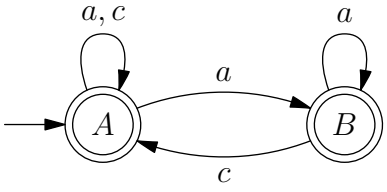
Aufgabe 2 (10 Punkte)
Zeichnen Sie einen deterministischen Kellerautomaten, der die Sprache aller Wörter $w \in \{a, b\}^*$ akzeptiert, die gleich viele as und bs enthalten.

Aufgabe 3 (10 Punkte)
Beweisen Sie mithilfe des Pumpinglemmas, daß folgende Sprache nicht kontextfrei ist:

$$\{\, u\$v\$w \mid u, v, w \in \{a, b\}^* \text{ und } (u = v \text{ oder } u = w) \,\}$$

Aufgabe 4 (10 Punkte)
Berechnen Sie das synchronisierte Produkt der beiden folgenden nichtdeterministischen endlichen Automaten. Bestimmen Sie vom resultierenden Automaten den minimalen DFA.

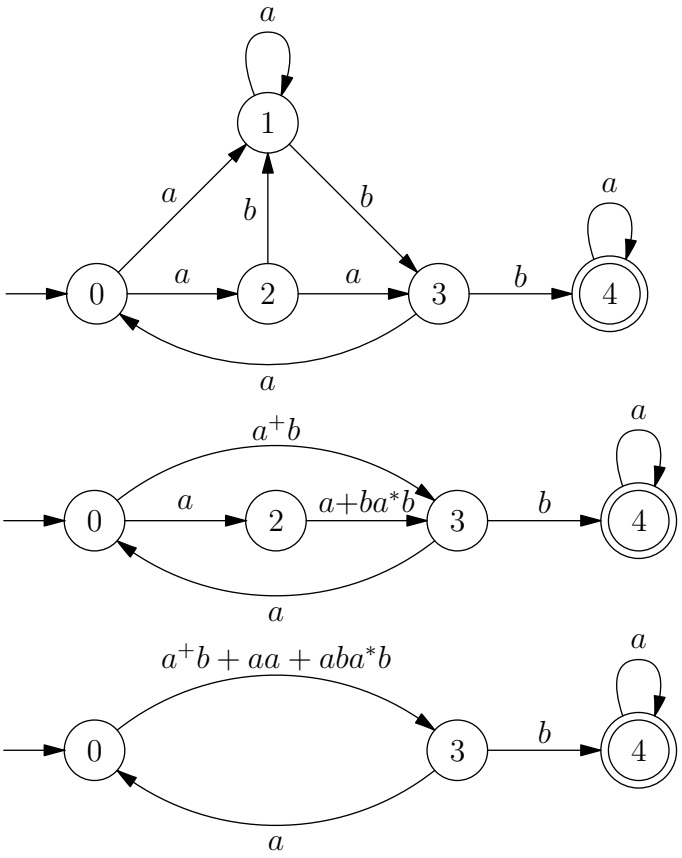




Formale Systeme, Automaten und Prozesse

Lösungsvorschlag

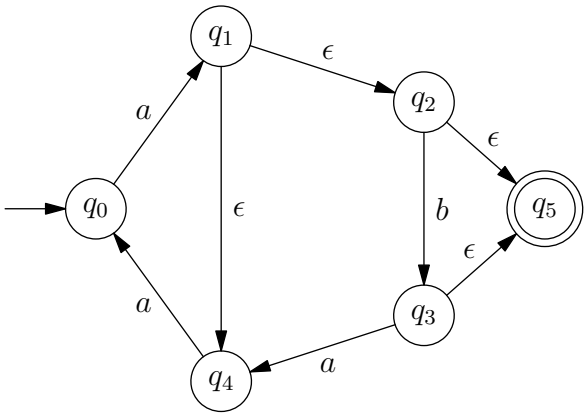
Aufgabe 1 (10 Punkte)
Wir können das Verfahren aus der Vorlesung verwenden, um die Zustände 1 und dann 2 zu eliminieren. (Die umgekehrte Reihenfolge ist vielleicht sogar noch besser.)



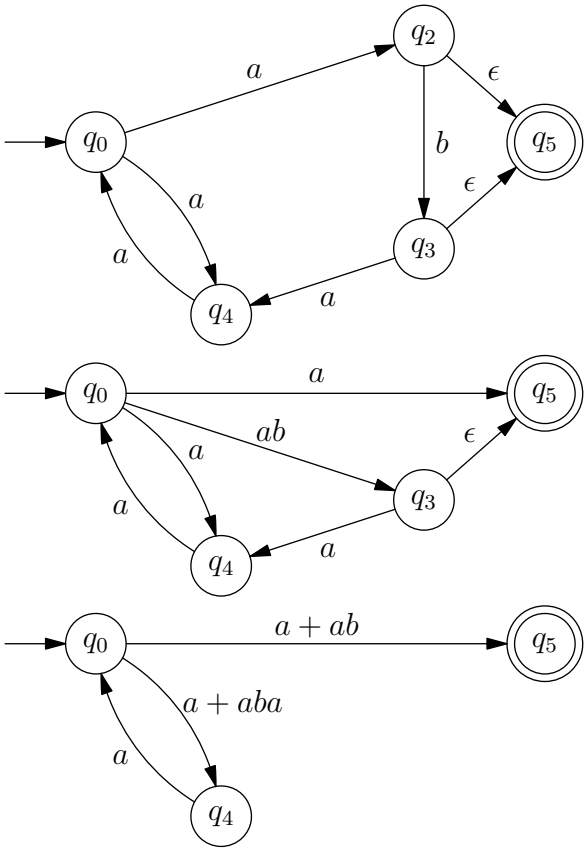
Nun können wir auch noch den Zustand 3 eliminieren oder sofort den regulären Ausdruck ablesen:

$$(a^+ba + aaa + aba^*ba)^*(a^+b + aa + aba^*b)ba^*$$

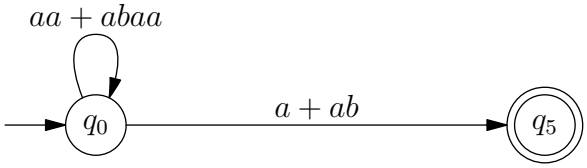
Beim zweiten Automaten stehen wir zunächst vor der Schwierigkeit, daß es zwei Endzustände gibt. Verwenden wir also erst einmal die Standardkonstruktion, welche dieses Problem löst.



Nun eliminieren wir nacheinander die Zustände q_1, q_2 und q_3:



Jetzt können wir den regulären Ausdruck schon bequem ablesen, aber zur Abwechslung eliminieren wir auch noch den Zustand q_4 um einmal stur beim Standardverfahren zu bleiben:



Jetzt ergibt sich der reguläre Ausdruck

$$(aa + abaa)^*(a + ab)$$

und wir sind fertig.

Typische Fehler: Der erste reguläre Ausdruck wurde in der Regel korrekt berechnet. Allerdings gab es hier teilweise Probleme den drittletzten Zustand zu eliminieren. Laut

dem üblichen Vorgehen erzeugt dies eine Schleife an Zustand 0 und eine Kante von 0 nach 4. Oft wurde jedoch die Schleife weggelassen (was möglich ist), dafür aber dann die kompliziertere Beschriftung der Kante von 0 nach 4 falsch angeben.

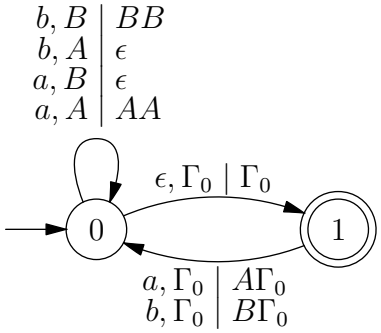
Beim zweiten Automat gab es große Probleme mit den zwei Endzuständen, da das Verfahren aus der Vorlesung nicht angewendet wurde. Oft wurden stattdessen Endzustände einfach eliminiert, beide Endzustände verschmolzen oder versucht, den Ausdruck an einem Automaten mit drei Zuständen abzulesen. In vielen Fällen führte dies zu falschen Lösungen.

Ein weiterer häufiger Fehler trat beim Entfernen der ϵ-Übergänge auf. Dieses Entfernen ist zwar nicht nötig, wurde aber dennoch oft angewendet. Allerdings wurden dabei häufig Kanten vergessen, zum Beispiel die Transition $(1, b, 3)$.

Äußerst umständlich war auch die unnötige Methode, die NFAs vorher mit der Potenzmengenkonstruktion zeitraubend in einen DFA zu überführen, wobei sich häufig Fehler einschlichen.

Aufgabe 2 (10 Punkte)

Wir benötigen einen Zustand 0, in dem wir den Keller benutzen, um die gelesen a's und b's zu zählen. Sobald der Zähler ausgeglichen ist, also nur Γ_0 auf dem Keller liegt, wechseln wir per ϵ-Transition in den akzeptierenden Zustand 1. Sobald in Zustand 1 ein Zeichen gelesen wird, wechseln wir wieder in Zustand 0 um weiter zu zählen. Man beachte das zu jedem beliebigen Zeitpunkt nur A's oder B's auf dem Keller liegen können, nie aber sowohl A's als auch B's.



Typische Fehler:

Sehr häufig wurde ein Automat angegeben, der nicht deterministisch ist. So gab es oft aus einem Zustand sowohl Transitionen der Form $a, \Gamma_0 \mid A\Gamma_0$ (oft auch in der Form $a, X \mid AX$) als auch gleichzeitig die Transition $\epsilon, \Gamma_0 \mid \Gamma_0$. Der Automat entscheidet also nichtdeterministisch ob er ein Zeichen liest oder die ϵ-Transition nutzt.

Dies wurde in der Regel genutzt um das Ende des Wortes zu raten, was deterministisch natürlich nicht möglich ist.

Weiterhin wurde oft ein Automat konstruiert, der mit leerem Keller akzeptiert. In der Regel wurde hier die Akzeptanzbedingung falsch angewendet, da der Automat sofort hält, wenn der Keller leer ist und nicht erst dann, wenn der Keller leer ist und gleichzeitig das Wort zu Ende ist. Auch dieser Ansatz entspricht im wesentlichen dem nichtdeterministischen Raten des Wortendes.

Übrigens ist es unmöglich, diese Sprache mit einem deterministischen Kellerautomaten zu erkennen, der mit leerem Keller akzeptiert, da nach gelesenem Präfix ab nicht klar ist, ob der Automat akzeptieren darf oder nicht.

Aufgabe 3 (10 Punkte)
Wir möchten gerne nachweisen, daß

$$\{\, u\$v\$w \mid u, v, w \in \{a, b\}^* \text{ und } (u = v \text{ oder } u = w)\,\}$$

nicht kontextfrei ist, indem wir zeigen, daß Bob eine Gewinnstrategie beim Pumping-Lemma-Spiel besitzt.

1. Alice wählt also eine beliebige Zahl n
2. Wir lassen Bob das Wort $z = a^n b^n \$ a^n b^n \$$ wählen, das sicher in der Sprache enthalten ist. Bisher hat also Bob nicht verloren.
3. Jetzt wählt Alice Wörter u, v, w, x, y mit $z = uvwxy$, $|vwx| < n$ und $|vx| > 0$.
4. Nun wählt Bob einfach $i = 0$ und wir müssen nachweisen, daß $uv^0wx^0y = uwy$ nicht in der gegebenen Sprache enthalten ist. Dann hat Bob nach den Regeln des Spiels gewonnen.

   Führen wir hierzu eine Fallunterscheidung durch.

   a) Falls vx ein \$ enthält, dann kann uwy nicht zwei \$ enthalten und gehört nicht zur gegebenen Sprache.

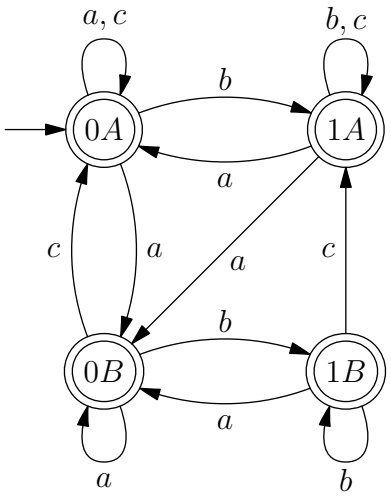
   b) Falls vx kein \$ enthält und vwx als Unterwort von z komplett vor dem ersten \$ in $a^n b^n \$ a^n b^n \$$ liegt, dann ist uv^0wx^0y von der Form $a^j b^k \$ a^n b^n \$$ mit $j < n$ oder $k < n$ und gehört so nicht zur gegebenen Sprache. Ähnlich verhält sich der Fall, daß vwx komplett hinter dem ersten \$ in z liegt.

   c) Falls vx kein \$ enthält und w das erste \$ in z enthält, dann ist $uv^0wx^0y = a^n b^j \$ a^k b^n$ mit $j < n$ oder $k < n$ und gehört so nicht zur Sprache. Falls w das zweite \$ in z enthält, dann ist $x = \epsilon$ und $uv^0wx^0y = a^n b^n \$ a^n b^{n-|v|} \$$ und wieder hat Bob gewonnen.

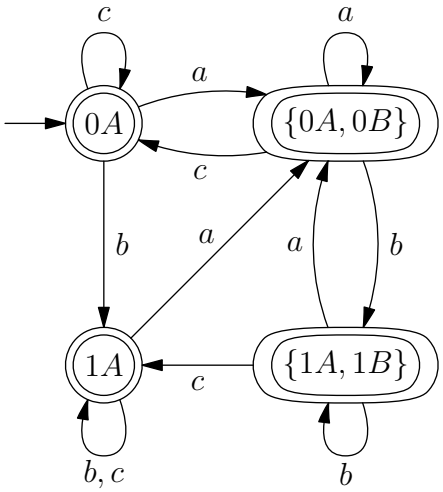
Typische Fehler:
Zunächst schien die Definition der Sprache Schwierigkeiten zu bereiten. So haben viele in ihrer Gewinnstrategie für Bob eine Fallunterscheidung nach den Fällen $u = v$ oder $u = w$ vorgenommen und hierbei die Strategie für Bob (Wahl eines Wortes $z \in L$) mit dem Kriterium für die Mitgliedschaft von Wörtern in der Sprache vermischt. Noch problematischer waren die Versuche, Bob das Wort $z = u^n \$ v^n \$ w^n$ oder ähnliche Konstrukte wählen zu lassen, ohne anzugeben, was u, v, w eigentlich sind. Wegen $u, v, w \notin \{a, b\}$ ist schon $z \notin L$ und damit die Strategie für Bob zum Scheitern verurteilt. Selbst wenn wir annehmen, daß $u, v, w \in \{a, b\}$ gemeint war, lassen sich leicht Gegenbeispiele finden.
Eher klassisch sind die Verständnis- bzw. Anwendungsfehler des Pumpinglemmas: Weder führt gleichzeitiges Argumentieren über alle Wörter $z \in L$ zum Erfolg (leichte Gegenbeispiele), noch ist es korrekt, eine oder nur bestimmte Zerlegungen zu betrachten.
Oft wurde etwa ein Wort $z = a^n \$\$ a^n$ (o.ä.) gewählt und dabei übersehen, daß Alice einfach $z = uvwxy$ mit $vwx = a\$\$a$ und $w = \$\$$ wählen kann, um das Spiel zu gewinnen. Ähnliche Probleme treten auf, wenn keines der Teilwörter zwischen den \$ das leere Wort ist, weil sich dann immer eine passende Zerlegung finden läßt.
Übrigens: Das Pumpinglemma gilt immer, und nicht nur für bestimmte Sprachen oder für bestimmte Wörter.

Aufgabe 4 (10 Punkte)

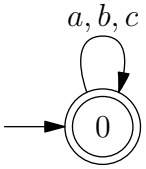
Als synchronisiertes Produkt erhalten wir mit dem Verfahren aus der Vorlesung:



Um den minimalen DFA zu bilden, wenden wir nun die Potenzmengenkonstruktion an und erhalten den folgenden Automaten.



Da alle Zustände auch Endzustände sind (und der Automat deterministisch ist, also insbesondere nicht blockiert), erkennt er die universelle Sprache. Der minimale DFA ist also:



Alternativ kann man auch dem NFA direkt ansehen, daß er nicht nie blockiert und nur Endzustände besitzt.

Typische Fehler: Sofern die Definition des synchronisierten Produkts bekannt war, wurden bei dieser Aufgabe wenig Fehler gemacht.