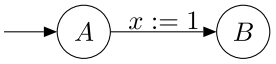
MUSTERLÖSUNG

Für die volle Punktzahl von 60 Punkten ist es notwendig, die ersten drei Aufgaben vollständig zu lösen. Von der vierten Aufgabe müssen Sie lediglich die einführenden einfachen Unteraufgaben lösen. Der Rest der vierten Aufgabe konnte verwendet werden, um zusätzliche 20 Punkte zu bekommen.

Aufgabe K1 (3+6+6+2=17 Punkte)

a) Der folgende Automat M_1 drückt die möglichen Progammabläufe dieses einfachen Programms P_1 aus:



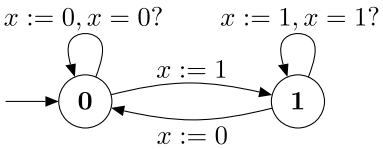

```
/* Programm P1 */
x := 1;
```

Wir betrachten das folgende Programm P_2. Geben Sie einen endlichen Automaten M_2 an, der die möglichen Programmabläufe modelliert (wie bekannt aus der Vorlesung und analog zu oben). Verwenden Sie $1, 2, 3, \ldots$ als Namen für die Zustände.

```
/* Programm P2 */
x := 0;
while(x=1) {
    x := 1;
}
print;
```

b) Konstruieren Sie das unsynchronisierte Produkt $M_{12} = M_1 \sqcup\!\sqcup M_2$. Geben Sie nur erreichbare Zustände an.

c) Der folgende Automat M_x modelliert das Verhalten der boolschen Variablen x:



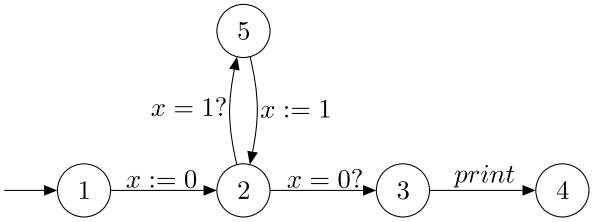
Konstruieren Sie das synchronisierte Produkt $M_{12} \circ M_x$, um das gemeinsame Verhalten der beiden Programme P_1 und P_2 unter Verwendung der Variablen x zu modellieren.

d) Beantworten Sie auf Basis des Automaten aus c) folgende Fragen:

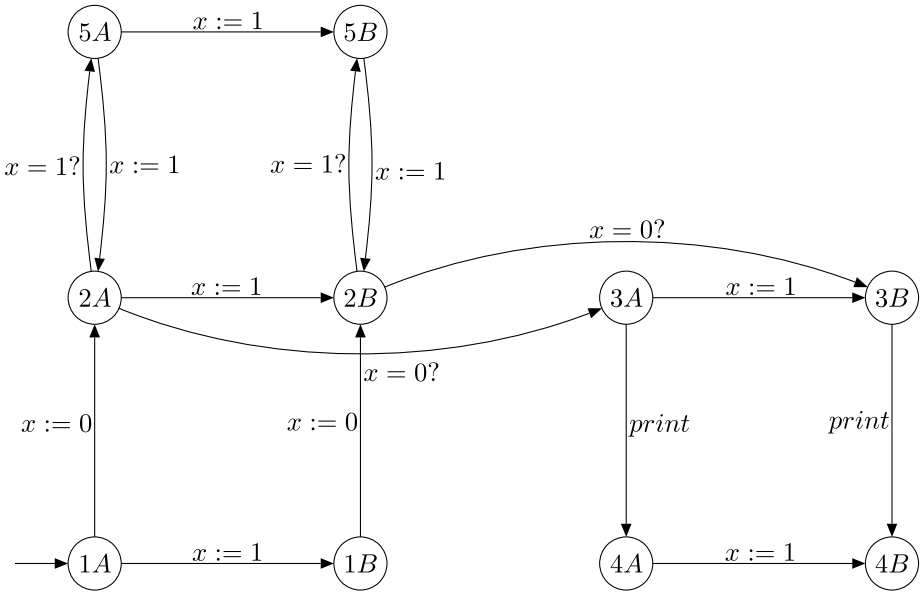
Kann *print* ausgeführt werden? Kann das synchronisierte Programm terminieren und dabei *print* nicht ausführen?

Lösungsvorschlag

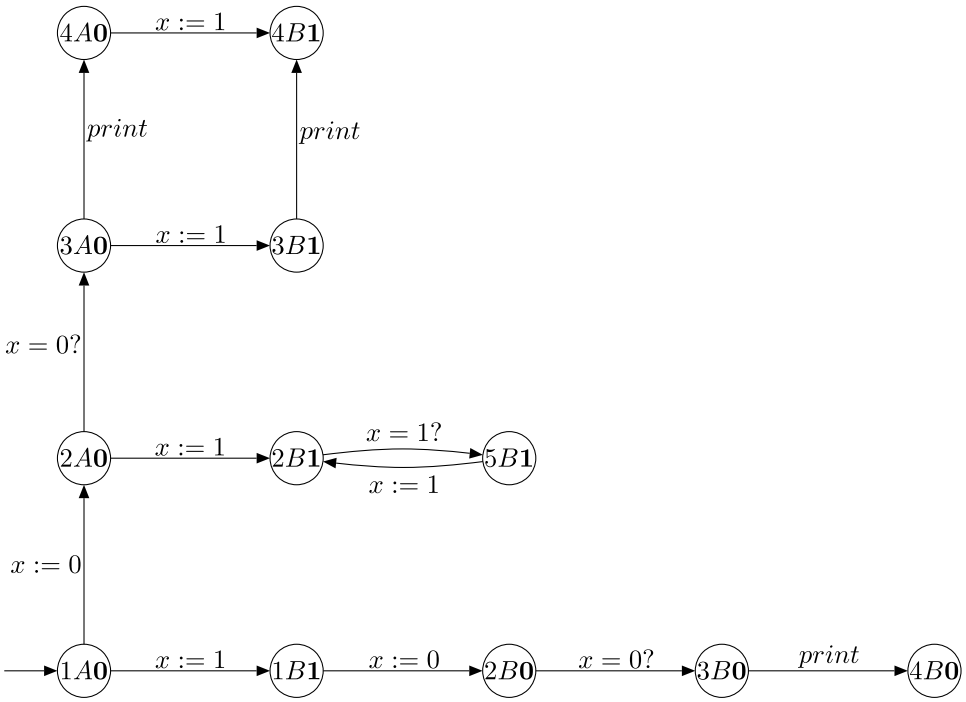
a) M_2:



b) M_{12}:

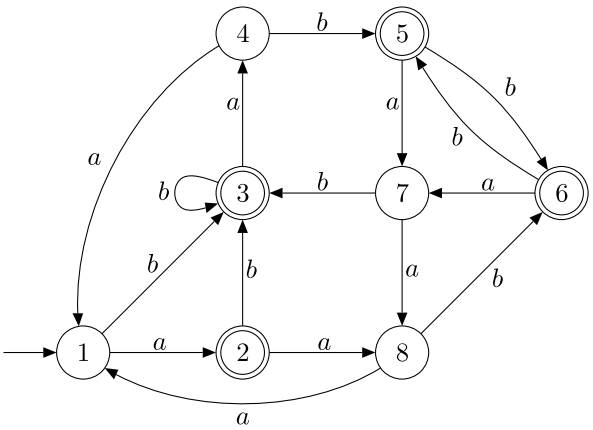


c) $M_{12} \circ M_x$:



d) *print* ist klar erreichbar, aber die einzige Möglichkeit einer Endlosschleife sind die Zustände $2B1$ und $5B1$, die nur erreichbar sind, wenn *print* nicht ausgeführt wird. Falls das Programm terminiert, führt es also stets *print* aus.

Aufgabe K2 (6+2+4+8+1=21 Punkte)
Es sei N der folgende DFA.



a) Minimieren Sie N mittels eines geeigneten Verfahrens. Geben Sie den minimalen Automaten graphisch an.

b) Geben Sie für jede Myhill–Nerode-Äquivalenzklasse genau einen Repräsentanten an.

Fortsetzung auf nächster Seite. Sei nun $L = \{w\$w \mid w \in \{a,b\}^*\} \subseteq \{a,b,\$\}^*$.

c) Betrachten Sie die folgenden Wörter. Geben Sie an, ob diese in derselben Äquivalenzklasse bzgl. der Myhill–Nerode-Äquivalenzrelation $\equiv_L$ von L liegen. Begründen Sie Ihre Antwort.

- a
- $ab\$a$
- $b\$$
- $aa\$b$

d) Geben Sie *alle* Myhill-Nerode-Äquivalenzklassen von L an. Geben Sie dazu die Sprache jeder Klasse explizit und jeweils einen (möglichst kurzen) Repräsentanten an. Begründen Sie außerdem, dass Wörter in der gleichen Äquivalenzklasse äquivalent sind und dass Wörter in verschiedenen Klassen nicht äquivalent sind.

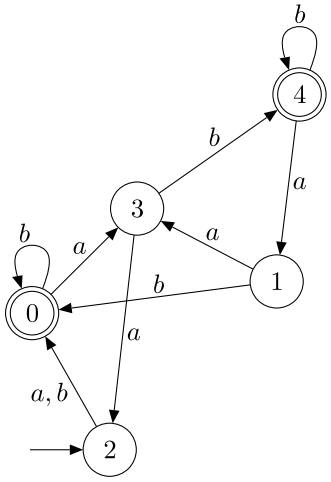
e) Ist L regulär? Begründen Sie!

Lösungsvorschlag

a) Die Tabelle vom Markierungsalgorithmus. Die Zahlen bezeichnen in welcher Phase die Unterscheidung stattgefunden hat. Wir starten mit Phase 1, die die End- von den Nicht-Endzuständen unterscheidet. Leeres Feld zeigt, dass das Zustandspaar nicht unterscheidbar ist.

	1	2	3	4	5	6	7	8
1		1	1	2	1	1	2	2
2	1			1	4	4	1	1
3	1			1	4	4	1	1
4	2	1	1		1	1	3	
5	1	4	4	1			1	1
6	1	4	4	1			1	1
7	2	1	1	3	1	1		3
8	2	1	1		1	1	3	

Die nicht-unterscheidbaren Zustände werden verschmolzen, also 4 mit 8 und 5 mit 6 und 2 mit 3.



b) $\varepsilon, a, aa, aab, aaba$

c)
- $ab\$a \equiv_L b\$$: Das einzige Wort, dass man an $ab\$a$ hängen kann, damit es in L ist, ist b. Das selbe gilt für $b\$$. Damit gilt die Äquivalenz.
- $a \not\equiv b\$$ (und damit $a \not\equiv ab\$a$) mit dem trennenden Wort b: $ab \notin L, b\$b \in L$
- $a \not\equiv aa\$b$ mit trennendem Wort $\$a$: $a\$a \in L, aab\$ba\$ \notin L$
- $aa\$b \not\equiv b\$$ (und damit $aa\$b \not\equiv ab\$a$) mit trennendem Wort b.

d) Es gibt folgende Äquivalenzklassen:

- $X_u = \{u\}$ für alle $u \in \{a, b\}^*$ je mit Repräsentanten u
- $Y_u = \{wu\$w \mid w \in \{a, b\}^*\}$ je mit Repräsentanten $u\$$
- $Z = L \setminus (\bigcup_{u \in \{a,b\}^*} X_u \cup Y_u) = \{u\$v \mid u \text{ ist kein Präfix von } v\} \cup \{u \mid \text{u enthält zwei } \$\}$ mit Repräsentant $\$\$$ und $a\$b$.

Seien $u \in \{a, b\}^*$ und $v \in \{a, b, \$\}^*$ mit $u \neq v$. Dann ist $u \not\equiv_L v$ da $\$u$ ein trennendes Wort ist, da $u\$u \in L$ aber $u\$v \notin L$.

Seien $vu\$v \neq wu\$w \in Y_u$. Ausschließlich durch das Anhängen von u kann man erreichen, dass die Wörter in der Sprache sind. Damit kann es kein trennendes Wort geben und somit sind sie äquivalent.

Sei $u \in Z$. Dann gibt es keine Möglichkeit das Wort zu erweitern, sodass es in der Sprache ist. Damit ist Z eine gültige Äquivalenzklasse. Sei $w \notin Z$. Dann gibt es ein trennendes Wort zwischen u und w, nämlich ein v sodass $wv \in L$.

e) Da es unendlich viele Myhill-Nerode-Klassen von L gibt, gilt $index(\equiv_L) = \infty$ und nach dem Satz von Myhill-Nerode ist L nicht regulär.

Aufgabe K3 (3+5+6+4=18 Punkte)
Sei G die folgende kontextfreie Grammatik:

$$S \to aS \mid SA \mid aA$$
$$A \to aB \mid b$$
$$B \to BS \mid b$$

a) Ist G eindeutig? Begründen Sie!

b) Sei $w = abaab$. Beantworten Sie durch eine geeignete pre^*-Konstruktion: Gilt $w \in L(G)$?

c) Wandeln Sie G in die Greibach-Normalform um. Gehen Sie dabei von der Ordnung $S < A < B$ aus und dass alle neu eingeführten Nichtterminale in der Ordnung größer sind.

d) Geben Sie einen Kellerautomaten an, der $L(G)$ erkennt und mit leerem Keller akzeptiert.

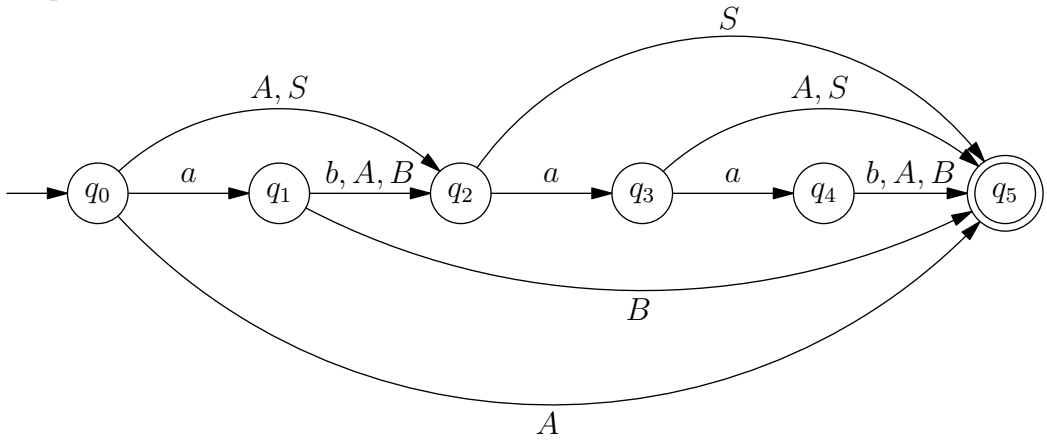
Lösungsvorschlag

a) Nein, z.B. für das Wort aab gibt es zwei verschiedene Linksableitungen:

$S \Rightarrow aS \Rightarrow aaA \Rightarrow aab$

$S \Rightarrow aA \Rightarrow aaB \Rightarrow aab$

b) Nein, da im gesättigten Automaten von $pre^*(w)$ keine S-Transition von q_0 zum Endzustand q_5 existiert:



c) Da die Grammatik nur Regeln enthält, die bereits in Form der Chomsky-Normalform oder der Greibach-Normalform sind, können wir diese direkt umwandeln:

Ursprungsgrammatik G:

$$S \to aS \mid SA \mid aA$$
$$A \to aB \mid b$$
$$B \to BS \mid b$$

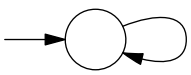
Elimination von Linksrekursionen:

$$S \to aSZ_1 \mid aAZ_1 \mid aS \mid aA$$
$$A \to aB \mid b$$
$$B \to b \mid bZ_2$$
$$Z_1 \to A \mid AZ_1$$
$$Z_2 \to S \mid SZ_2$$

Elimination der Regeln von Z_1 und Z_2 durch einsetzen:

$$
\begin{aligned}
S &\to aSZ_1 \mid aAZ_1 \mid aS \mid aA \\
A &\to aB \mid b \\
B &\to b \mid bZ_2 \\
Z_1 &\to aB \mid b \mid aBZ_1 \mid bZ_1 \\
Z_2 &\to aS \mid aSZ_1 \mid aA \mid aAZ_1 \mid aSZ_2 \mid aSZ_1Z_2 \mid aAZ_2 \mid aAZ_1Z_2
\end{aligned}
$$

d) Sei S das Kellerbodensymbol:



ϵ, S	aS
ϵ, S	SA
ϵ, S	aA
ϵ, A	aB
ϵ, A	b
ϵ, B	BS
ϵ, B	b
a, a	ϵ
b, b	ϵ

Aufgabe K4 (2+2+6*+6*+8*=24 Punkte)

Wir definieren ein $0L$-System als ein Dreitupel $S = (\Sigma, h, w)$ wobei Σ ein Alphabet ist, $h\colon \Sigma^* \to \Sigma^*$ ein Homomorphismus und $w \in \Sigma^*$ das Startwort ist. Die Sprache $L(S)$ ist so definiert:

$$L(S) = \{w, h(w), h(h(w)), h(h(h(w))), \ldots\}$$

Eine Sprache L ist eine $0L$-Sprache, falls es ein $0L$-System S gibt mit $L = L(S)$. In dieser Aufgabe wollen wir untersuchen, wie sich $0L$-Systeme im Vergleich zu den in der Vorlesung behandelten Systemen verhalten.

a) Sei $S_1 = (\{a, b\}, h, a)$ mit $h(a) = ab$ und $h(b) = a$. Geben Sie alle Wörter in $L(S_1)$ an, die höchstens Länge drei haben.

b) Geben Sie ein Wort aus dem Alphabet $\{a, b\}$ an, das nicht in $L(S_1)$ enthalten ist. Begründen Sie Ihre Antwort kurz.

c) Geben Sie eine *kontextsensitive* Grammatik an, welche $L(S_1)$ erzeugt. Erläutern Sie Ihre Grammatik.

d) Beweisen oder widerlegen Sie: Alle $0L$-Sprachen sind kontextfrei.

e) Beweisen oder widerlegen Sie: Alle $0L$-Sprachen sind kontextsensitiv.

Lösungsvorschlag

a) Da $|h(w)| \geq |w|$ gilt, können wir einfach $w, h(w), \ldots$ aufzählen, bis wir ein Wort finden, dessen Länge mindestens vier ist. Dabei erhalten wir a, ab, aba, $abaab$ und können schon aufhören. Die gesuchten Wörter sind also a, ab, aba.

b) $\epsilon \notin L(S)$, da alle Wörter mindestens so lang wie das Startwort a sind.

c) Eine Chomsky-1-Grammatik könnte so aussehen:

$$\begin{aligned}
S &\to A_l B A_r \mid a \mid ab \\
A_l &\to A_l \dot{B} \\
\dot{X} A &\to X A \dot{B} \\
\dot{X} B &\to X \dot{A} \\
\dot{X} A_r &\to X A B_r \\
\dot{X} B_r &\to X A_r \\
A &\to a \\
B &\to b \\
A_l &\to a \\
A_r &\to a \\
B_r &\to b
\end{aligned}$$

Hier steht X für A, A_l oder B.

Ein „Punkt“ kann von links nach rechts wandern und dabei den Homomorphismus anwenden. Das kann beliebig oft passieren. Erreicht der Punkt das rechte Ende, verschwindet er. Die ganz linken und rechten Symbole sind speziell markiert und diese Invariante wird immer aufrechterhalten. Ein Homomorphismus muss auf das ganze Wort angewendent

werden: Sonst bliebe ein Punkt übrig, der nicht zu einem Terminalsymbol gemacht werden kann.

Die Regeln dieser Grammatik sind längenmonoton. Eine solche Regel $ABC \to DEF$ lässt sich als $ABC \to XBC \to XYC \to XYZ \to DYZ \to DEZ \to DEF$ schreiben, was kontextsensitiv ist (aber länger).

d) Nein, wie folgendes Gegenbeispiel zeigt: $S = (\{a\}, a \mapsto aa, a)$. Offensichtlich ist $L(S) = \{\, a^{2^n} \mid n \geq 0 \,\}$. Diese Sprache ist aber nicht kontextfrei, wie man zum Beispiel mit dem Pumpinglemma zeigen kann. Wenn n die Konstante des PL ist, wählen wir $w = a^{2^n}$. Wenn $uvwxy = w$, $|vx| > 0$ und $|vwx| \leq n$, dann ist offensichtlich $|uv^2wx^2y|$ mindestens $2^n + 1$ und höchstens $2^n + n$ und liegt damit nicht in $L(S)$.

e) Eine kontextsensitive Grammatik für $S = (\Sigma, h, w)$ mit $|w| = k$ kann nach folgendem Prinzip funktionieren. Vom Startsymbol S lassen sich Wörter LwH^n für $n \geq 1$ erzeugen. Die Regeln sorgen dafür, dass L und H nur auf eine Weise verschwinden können, nämlich wenn sie als LH nebeneinanderstehen. Es gibt Regeln $aH \mapsto Hh(a)$, welche das Symbol H nach links gehen lassen und dabei das übersprungene Symbol a durch $h(a)$ ersetzen. Dadurch wird insgesamt w durch $h(w)$ usw. ersetzt. Jetzt muss aber noch das LH^n verschwinden, was nicht erlaubt ist. Dieses Problem können wir so lösen: Es läßt sich leicht beweisen, dass wenn wir $LH^nh^n(w)$ erreicht haben, $|h^n(w)| = \Omega(n)$ gilt, also dass die Anzahl der H höchstens linear in der Gesamtlänge ist. Wir verändern das Alphabet und die Grammatik so, dass neue Symbole gleich mehrere alte Symbole zu Blöcken zusammenfassen. Ist zum Beispiel $\Sigma = \{a, b\}$, dann enthält das neue Alphabet unter anderem $\boxed{aaa}$, $\boxed{bab}$, $\boxed{Lab}$, $\boxed{aHb}$ und alle anderen Kombinationen bis zur Länge drei, welche wir aber auch höher wählen können. Die neuen Regeln simulieren die alten Regeln: Gibt es zum Beispiel die Regeln $aA \to aBbc$, $B \to L$, $Caa \to aDaa$, dann führen wir unter anderem die neue Regel $\boxed{aAB}\boxed{Caa} \to \boxed{aBb}\boxed{cLa}\boxed{Daa}$ ein. Wir müssen darauf achten, dass es mit den Längen gut geht, welche alle ein Vielfaches von hier drei sein müssen. Klappt das nicht, dann können wir beispielsweise ein spezielles Symbol E in die Blöcke einfügen.

Damit $LH^nh^n(w)$ in $h^n(w)$ umgewandelt wird, können wir eine Regel $LH \to LZ$ einführen und weitere, die dafür sorgen, dass Z (und E) nach rechts und links gleiten kann.

Am Ende kann durch Regeln $\boxed{aba} \mapsto aba$ (bzw. $\boxed{aEa} \to ab$ und $\boxed{aZb} \to ab$) aus den Blöcken wieder normale Terminalworte erzeugt werden. Dabei werden auch die jetzt unnötigen Symbole L, E, Z entfernt. Da ihre Anzahl linear klein ist, können wir die Blocklänge so wählen, dass keine Blöcke nur aus solchen Symbolen bestehen und es keine ϵ-Regeln gibt.

Ein weiteres Problem ergibt sich, wenn zum Beispiel $h(a) = \epsilon$ gilt. Das können wir aber durch $h(a) = E$ ersetzen. Auch nach dieser Änderung bleibt die Anzahl der E klein genug.

Formale Systeme, Automaten, Prozesse, SS 2020
Prof. Dr. P. Rossmanith
J. Dreier, T. Hartmann, H. Lotze, D. Mock

Datum: 16.09.2020

Klausur mit Lösungen 02