2. Übungsblatt (mit Lösungen)

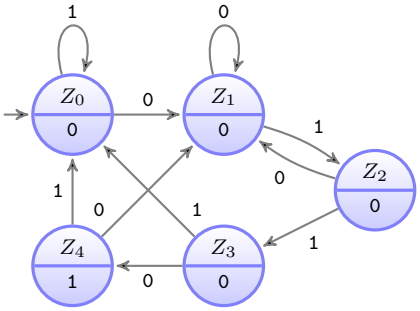
3.0 VU Formale Modellierung

Marion Brandsteidl, Lara Spendier, Gernot Salzer

3. Jänner 2013

Aufgabe 1 (0.3 Punkte)

Sei $\mathcal{A}$ der folgende Moore-Automat.



(a) Geben Sie die Ausgaben zu folgenden Eingaben an: `101011`, `100110`, `010111`.

(b) Berechnen Sie schrittweise $\delta^*(Z_0, \texttt{10011101101})$ und $\gamma^*(Z_0, \texttt{10011101101})$.

(c) Beschreiben Sie die Übersetzungsfunktion $[\mathcal{A}]$.

Lösung

(a)

w:	`101011`	`100110`	`010111`
$[\mathcal{A}](w)$:	`000000`	`000001`	`000000`

(b) Übergangsfunktion $\delta^*(Z_0, \texttt{10011101101})$:

$$\begin{aligned}
\delta^*(Z_0, \texttt{10011101101}) &= \delta^*(\delta(Z_0, \texttt{1}), \texttt{0011101101}) &&= \delta^*(Z_0, \texttt{0011101101}) \\
&= \delta^*(\delta(Z_0, \texttt{0}), \texttt{011101101}) &&= \delta^*(Z_1, \texttt{011101101}) \\
&= \delta^*(\delta(Z_1, \texttt{0}), \texttt{11101101}) &&= \delta^*(Z_1, \texttt{11101101}) \\
&= \delta^*(\delta(Z_1, \texttt{1}), \texttt{1101101}) &&= \delta^*(Z_2, \texttt{1101101}) \\
&= \delta^*(\delta(Z_2, \texttt{1}), \texttt{101101}) &&= \delta^*(Z_3, \texttt{101101}) \\
&= \delta^*(\delta(Z_3, \texttt{1}), \texttt{01101}) &&= \delta^*(Z_0, \texttt{01101}) \\
&= \delta^*(\delta(Z_0, \texttt{0}), \texttt{1101}) &&= \delta^*(Z_1, \texttt{1101}) \\
&= \delta^*(\delta(Z_1, \texttt{1}), \texttt{101}) &&= \delta^*(Z_2, \texttt{101}) \\
&= \delta^*(\delta(Z_2, \texttt{1}), \texttt{01}) &&= \delta^*(Z_3, \texttt{01}) \\
&= \delta^*(\delta(Z_3, \texttt{0}), \texttt{1}) &&= \delta^*(Z_4, \texttt{1}) \\
&= \delta^*(\delta(Z_4, \texttt{1}), \varepsilon) &&= \delta^*(Z_0, \varepsilon) \\
&= Z_0
\end{aligned}$$

Ausgabefunktion $\gamma^*(Z_0, \texttt{10011101101})$:

$$\begin{aligned}
\gamma^*(Z_0, \texttt{10011101101}) &= \gamma(Z_0, \texttt{1}) \cdot \gamma^*(\delta(Z_0, \texttt{1}), \texttt{0011101101}) &&= \texttt{0} \cdot \gamma^*(Z_0, \texttt{0011101101}) \\
&= \texttt{0} \cdot \gamma(Z_0, \texttt{0}) \cdot \gamma^*(\delta(Z_0, \texttt{0}), \texttt{011101101}) &&= \texttt{00} \cdot \gamma^*(Z_1, \texttt{011101101}) \\
&= \texttt{00} \cdot \gamma(Z_1, \texttt{0}) \cdot \gamma^*(\delta(Z_1, \texttt{0}), \texttt{11101101}) &&= \texttt{000} \cdot \gamma^*(Z_1, \texttt{11101101}) \\
&= \texttt{000} \cdot \gamma(Z_1, \texttt{1}) \cdot \gamma^*(\delta(Z_1, \texttt{1}), \texttt{1101101}) &&= \texttt{0000} \cdot \gamma^*(Z_2, \texttt{1101101}) \\
&= \texttt{0000} \cdot \gamma(Z_2, \texttt{1}) \cdot \gamma^*(\delta(Z_2, \texttt{1}), \texttt{101101}) &&= \texttt{00000} \cdot \gamma^*(Z_3, \texttt{101101}) \\
&= \texttt{00000} \cdot \gamma(Z_3, \texttt{1}) \cdot \gamma^*(\delta(Z_3, \texttt{1}), \texttt{01101}) &&= \texttt{000000} \cdot \gamma^*(Z_0, \texttt{01101}) \\
&= \texttt{000000} \cdot \gamma(Z_0, \texttt{0}) \cdot \gamma^*(\delta(Z_0, \texttt{0}), \texttt{1101}) &&= \texttt{0000000} \cdot \gamma^*(Z_1, \texttt{1101}) \\
&= \texttt{0000000} \cdot \gamma(Z_1, \texttt{1}) \cdot \gamma^*(\delta(Z_1, \texttt{1}), \texttt{101}) &&= \texttt{00000000} \cdot \gamma^*(Z_2, \texttt{101}) \\
&= \texttt{00000000} \cdot \gamma(Z_2, \texttt{1}) \cdot \gamma^*(\delta(Z_2, \texttt{1}), \texttt{01}) &&= \texttt{000000000} \cdot \gamma^*(Z_3, \texttt{01}) \\
&= \texttt{000000000} \cdot \gamma(Z_3, \texttt{0}) \cdot \gamma^*(\delta(Z_3, \texttt{0}), \texttt{1}) &&= \texttt{0000000001} \cdot \gamma^*(Z_4, \texttt{1}) \\
&= \texttt{0000000001} \cdot \gamma(Z_4, \texttt{1}) \cdot \gamma^*(\delta(Z_4, \texttt{1}), \varepsilon) &&= \texttt{00000000010} \cdot \gamma^*(Z_0, \varepsilon) \\
&= \texttt{00000000010} \cdot \varepsilon &&= \texttt{00000000010}
\end{aligned}$$

(c) [$\mathcal{A}$]: Der Automat signalisiert die Folge `0110` in der Eingabe: Wird `0110` eingelesen, so ist die Ausgabe `1`, sonst `0`.

Aufgabe 2 (0.3 Punkte)

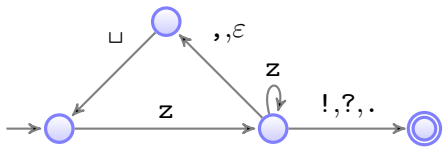
Geben Sie endliche Automaten für folgende Sprachen L an.

(a) $L = \{\, w \in \{\texttt{z}, \sqcup, \texttt{,}, \texttt{!}, \texttt{?}, \texttt{.}\}^+ \mid w \text{ ist ein Satz} \,\}$
Ein Satz besteht aus Wörtern, die durch ein Leerzeichen ($\sqcup$) oder durch einen Beistrich plus Leerzeichen voneinander getrennt sind. Jeder Satz wird durch eines der Zeichen `!`,`?` oder `.` beendet. Jedes Wort enthält ein oder mehrere `z`'s.
Beispiel: $\texttt{zzzzzz}\sqcup\texttt{zzzz}\sqcup\texttt{zzz}\sqcup\texttt{zz,}\sqcup\texttt{zz,}\sqcup\texttt{zzzzzzzz!} \in L$

(b) $L = \{\, w \in \{\texttt{0}, \texttt{1}\}^+ \mid \text{Binärzahl } w \text{ ist durch 5 teilbar} \,\}$
Beispiele für Wörter in der Sprache sind `000`, `101` und `0011001`, da die Zahlen 0, 5 und 25 durch 5 teilbar sind.

Lösung

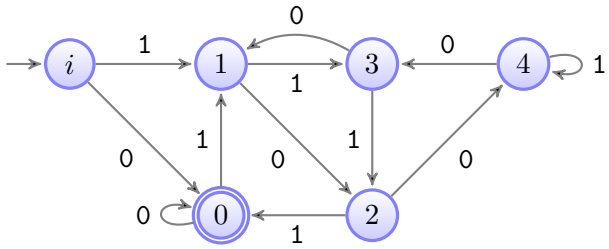
(a)



(b) Da es um Teilbarkeit modulo 5 geht, sehen wir die Zustände 0, 1, 2, 3 und 4 für die fünf möglichen Divisionsreste vor; befindet sich der Automat im Zustand n, so soll das bedeuten, dass das bisher gelesene Binärnumeral den Rest n bei Division durch 5 liefert. Zustand 0 ist Endzustand, da er den durch 5 teilbaren Zahlen entspricht.

Um sicherzustellen, dass die akzeptierte Zahl aus mindestens einer Ziffer besteht, benötigen wir zusätzlich den Startzustand i; er entspricht ebenfalls dem Divisionsrest 0. Würden wir den Zustand 0 selber zum Startzustand machen, würde der Automat auch das Leerwort akzeptieren.

Zur Festlegung der Übergänge überlegen wir, wie sich der Divsionsrest ändert, wenn auf das bisher verarbeitete Numeral eine `0` bzw. eine `1` folgt. War der bisherige Rest n, so ist der Rest nach einem `0`er $(2n+0)$ mod 5 und nach einem `1`er $(2n+1)$ mod 5. Somit bestimmt sich der Folgezustand von 3 bei einem `0`er zu $(2 \cdot 3 + 0) \bmod 5 = 1$ und bei einem `1`er zu $(2 \cdot 3 + 1) \bmod 5 = 2$. Wir erhalten den folgenden Automaten.



Aufgabe 3 (0.3 Punkte)

Sei L die Sprache

$$\{\, w \in \{0,1\}^* \mid w \text{ enthält das Teilwort } 001,\ 010 \text{ oder } 100 \,\}\ .$$

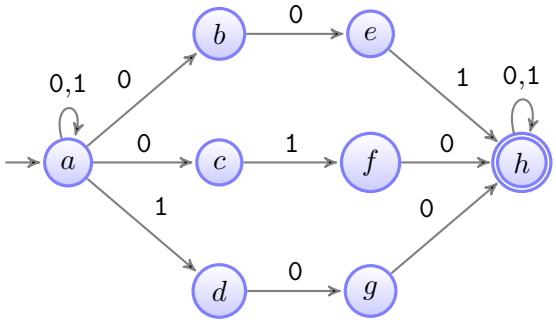
Geben Sie einen *deterministischen* Automaten für L an. Geben Sie folgendermaßen vor:

1. Konstruieren Sie einen indeterministischen Automaten für diese Sprache.
2. Wandeln Sie den indeterministischen Automaten mit Hilfe des in der Vorlesung besprochenen Verfahrens in einen deterministischen um.

Ist der so gewonnene Automat minimal, d.h., ist er unter den deterministischen Automaten für L jener mit der kleinstmöglichen Zahl von Zuständen?

Lösung

Wir konstruieren zunächst auf möglichst direktem Weg einen beliebigen Automaten für die gesuchte Sprache. Dieser ist im Zustand a indeterministisch: Für das Symbol `0` gibt es drei und für das Symbol `1` zwei Folgezustände. Zusätzlich zur graphischen Darstellung geben wir die Übergangsfunktion auch als Tabelle an, da diese bei der Determinisierung hilft.



δ^*	0	1
a	$\{a,b,c\}$	$\{a,d\}$
b	$\{e\}$	$\{\}$
c	$\{\}$	$\{f\}$
d	$\{g\}$	$\{\}$
e	$\{\}$	$\{h\}$
f	$\{h\}$	$\{\}$
g	$\{h\}$	$\{\}$
h	$\{h\}$	$\{h\}$

Einen deterministischen Automaten erhalten wir, indem wir den indeterminischen Automaten simulieren. Ein Zustand des deterministischen Automaten repräsentiert dabei jene Zustände des indeterministischen, in denen sich dieser zu diesem Zeitpunkt befinden kann. Der Startzustand wird mit $\{a\}$ bezeichnet, da sich der indeterministische Automat zu Beginn im Zustand a (und nur in diesem) befindet. Von diesem Zustand ausgehend erstellen wir zeilenweise die Tabelle für die Übergangsfunktion des deterministischen Automaten.

$\hat{\delta}$	0	1
$\{a\}$	$\{a,b,c\}$	$\{a,d\}$
$\{a,b,c\}$	$\{a,b,c,e\}$	$\{a,d,f\}$
$\{a,d\}$	$\{a,b,c,g\}$	$\{a,d\}$
$\{a,b,c,e\}$	$\{a,b,c,e\}$	$\{a,d,f,h\}$
$\{a,d,f\}$	$\{a,b,c,g,h\}$	$\{a,d\}$
$\{a,b,c,g\}$	$\{a,b,c,e,h\}$	$\{a,d,f\}$
$\{a,d,f,h\}$	$\{a,b,c,g,h\}$	$\{a,d,h\}$
$\{a,b,c,g,h\}$	$\{a,b,c,e,h\}$	$\{a,d,f,h\}$
$\{a,b,c,e,h\}$	$\{a,b,c,e,h\}$	$\{a,d,f,h\}$
$\{a,d,h\}$	$\{a,b,c,g,h\}$	$\{a,d,h\}$

Jene Zustände, die einer Situation entsprechen, in der der indeterministische Automat einen Endzustand erreicht hat, sind die Endzustände des deterministischen Automaten; in diesem Beispiel sind das alle Zustände, deren Bezeichnung h enthält. Dieser wird somit durch das Tupel $\langle \widehat{Q}, \{0,1\}, \hat{\delta}, \{a\}, \widehat{F} \rangle$ beschrieben, wobei

$$\widehat{F} = \{\{a,d,f,h\},\{a,b,c,g,h\},\{a,b,c,e,h\},\{a,d,h\}\}$$
$$\widehat{Q} = \{\{a\},\{a,b,c\},\{a,d\},\{a,b,c,e\},\{a,d,f\},\{a,b,c,g\}\} \cup \widehat{F}$$

Dieser Automat ist nicht der kleinstmögliche deterministische Automat. In der Vorlesung wurde zwar kein Minimierungsverfahren besprochen, man sieht aber, dass von den

Endzuständen mit jedem der beiden Symbole `0` und `1` wieder nur Endzustände erreichbar sind. Man kann diese daher zu einem einzigen Endzustand zusammenfassen, ohne die akzeptierte Sprache zu verändern.

Aufgabe 4 (0.3 Punkte)

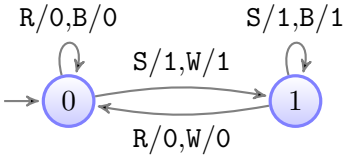
Sei $\Sigma = \{\texttt{S}, \texttt{R}, \texttt{W}, \texttt{B}\}$ das Eingabe- und $\Gamma = \{\texttt{0}, \texttt{1}\}$ das Ausgabealphabet. Die Eingabesymbole stehen für folgende Aktivitäten:

- `S`: Das Symbol `1` wird ausgeben (**S**et)
- `R`: Das Symbol `0` wird ausgeben (**R**eset)
- `W`: Die Ausgabe **W**echselt. War das letzte Symbol eine `1`, wird `0` ausgegeben, sonst `1`.
- `B`: Das zuletzt ausgegebene Symbol wird nochmals ausgegeben (**B**ewahren).

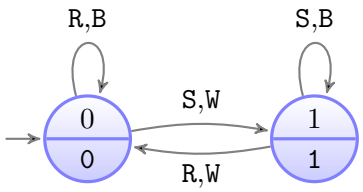
Beispielsweise führt die Eingabe `SBWRRBW` zur Ausgabe `1100001`.
Geben Sie sowohl einen Mealy- als auch einen Moore-Automaten an, der diese Funktion berechnet.

Lösung

Die Aktionen `W` und `B` beziehen sich auf die letzte Ausgabe, daher muss sich der Automat die beiden Möglichkeiten dafür, `0` und `1`, in unterschiedlichen Zuständen „merken“. Wir erhalten den folgenden Mealy-Automaten.



Der Startzustand ist willkürlich gewählt und bewirkt, dass `B` bzw. `W` als erste Aktion die Ausgabe `0` bzw. `1` liefert.
Der angegebene Mealy-Automat besitzt die Eigenschaft, dass alle Übergänge, die in denselben Zustand führen, dieselbe Ausgabe erzeugen. Diese ist also vom Übergang unabhängig und kann dem Zielzustand zugeordnet werden. Der Automat kann daher unverändert auch als Moore-Automat aufgefasst werden.

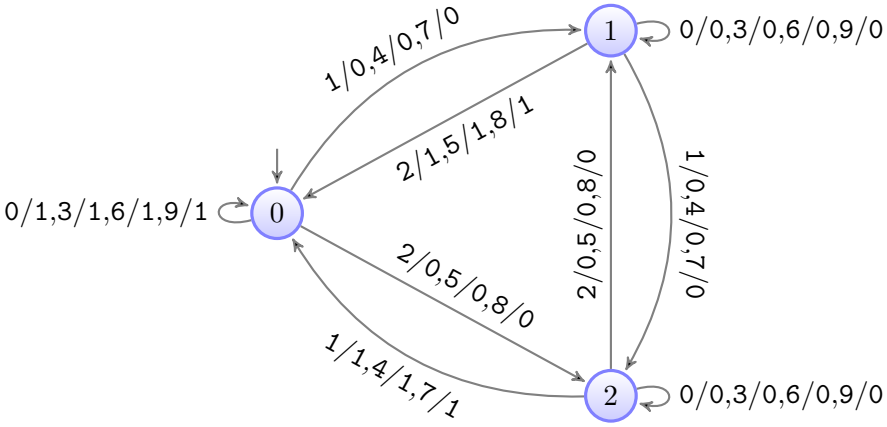


Aufgabe 5 (0.3 Punkte)

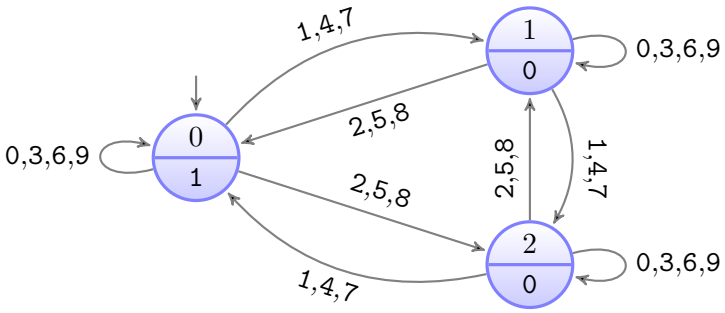
Sei $\Sigma = \{0, 1, 2, \ldots, 9\}$ das Eingabe- und $\Gamma = \{0, 1\}$ das Ausgabealphabet. Geben Sie sowohl einen Mealy- als auch einen Moore-Automaten an, der ausgibt, ob die bisher eingegebene Zahl durch 3 teilbar ist (1 für Ja und 0 für Nein).
Hinweis: Verwenden Sie die Tatsache, dass eine Zahl genau dann durch 3 teilbar ist, wenn sich die Quersumme der Zahl durch 3 teilen lässt.

Lösung

Wir verwenden drei Zustände 0, 1 und 2, die angeben, ob die Quersumme des bisher gelesenen Numerals durch 3 teilbar ist oder den Rest 1 bzw. 2 ergibt. Ist das nächste Symbol 0, 3, 6 oder 9, ändert sich am Divisionsrest der Quersumme nichts; bei 1, 4 und 7 erhöht er sich um 1, und bei 2, 5 und 8 um 2. Wir erhalten den folgenden Mealy-Automaten.



Die Ausgabe hängt wie in der Lösung zu Aufgabe 4 nur vom Zielzustand ab, daher ist der Moore-Automat im Wesentlichen identisch mit dem Mealy-Automaten.



Aufgabe 6 (0.3 Punkte)

Sind folgende Gleichungen für beliebige Sprachen L gültig? Falls ja, begründen Sie warum, falls nein, geben Sie ein Gegenbeispiel an.

(a) $L \cup \{\} = L \cdot \{\varepsilon\}$

(b) $\{\varepsilon\} \cdot L^* = L^+$

(c) $(L \cdot L)^* = L^* \cdot L^*$

(d) $L^+ \cup \{\varepsilon\} = L^* \cdot \{\}$

Lösung

(a) Diese Gleichung gilt für beliebige Sprachen L, da $L \cup \{\} = L = L \cdot \{\varepsilon\}$.

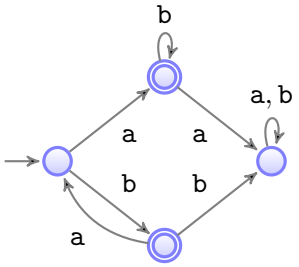
(b) Diese Gleichung gilt nicht allgemein. Wegen $\{\varepsilon\} \cdot L^* = L^*$ ist die Gleichung äquivalent zu $L^* = L^+$. Diese Gleichung ist genau dann erfüllt, wenn L das Leerwort enthält, i.e., wenn $\varepsilon \in L$. Ein Gegenbeispiel wäre $L = \{\mathtt{a}\}$.

(c) Diese Gleichung gilt in der Regel nicht. Gegenbeispiel: $L = \{\mathtt{a}\}$; wir erhalten $(\{\mathtt{a}\} \cdot \{\mathtt{a}\})^* = \{\mathtt{aa}\}^* \neq \{\mathtt{a}\}^* = \{\mathtt{a}\}^* \cdot \{\mathtt{a}\}^*$.

(d) Diese Gleichung wird von keiner Sprache erfüllt. Die linke Seite vereinfacht sich zu $L^+ \cup \{\varepsilon\} = L^*$; diese Sprache enthält offenbar das Leerwort. Die rechte Seite vereinfacht sich zu $L^* \cdot \{\} = \{\}$; diese Sprache enthält gar kein Wort, insbesondere nicht das Leerwort.

Aufgabe 7 (0.4 Punkte)

Sei w^r das Spiegelwort zum Wort w, d.h., w^r ist das Wort w von hinten nach vorne gelesen. Beispielsweise gilt $(\mathtt{abac})^r = \mathtt{caba}$. Sei weiters L^r die Spiegelsprache zur Sprache L, die dadurch entsteht, dass jedes Wort in L gespiegelt wird: $L^r = \{\, w^r \mid w \in L \,\}$.
Sei $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ein beliebiger deterministischer Automat und $L = \mathcal{L}(\mathcal{A})$ die von ihm akzeptierte Sprache. Geben Sie ein Verfahren an, um daraus einen Automaten $\mathcal{A}^r$ für die Spiegelsprache zu erhalten; es soll also $\mathcal{L}(\mathcal{A}^r) = L^r$ gelten. Welche Eigenschaft regulärer Sprachen ergibt sich daraus? Lässt sich das Verfahren auch auf nicht-deterministische Automaten anwenden? Wenden Sie Ihr Verfahren auf den folgenden Automaten an und konstruieren Sie einen Automaten für die Spiegelsprache.

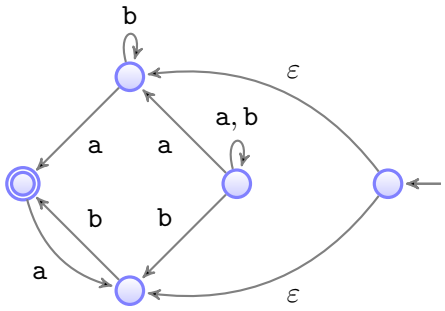


Lösung

Die grundlegende Idee ist, alle Übergänge zu spiegeln und Start- und Endzustände zu vertauschen. Falls $\mathcal{A}$ mehrere Endzustände besitzt, gäbe es aber mehrere Startzustände in $\mathcal{A}^r$. Um das zu vermeiden, muss man einen neuen Startzustand einführen, von dem aus man mit dem Leerwort in die ursprünglichen Endzustände wechseln kann. Im Allgemeinen ist $\mathcal{A}^r$ also aus zwei Gründen indeterministisch: einerseits ε-Kanten vom neuen Startzustand zu den alten Endzuständen, andererseits besitzt ein Zustand q für ein Eingabesymbol nun mehrere Folgezustände, wenn man in $\mathcal{A}$ (deterministsch) von diesen Folgezuständen nach q gelangt.
Formale Darstellung: $\mathcal{A}^r = \langle Q \cup \{i\}, \Sigma, \delta^r, i, \{q_0\}\rangle$, wobei die Übergangsrelation δ^r definiert ist durch

$$\delta^r = \{\, (\delta(q,s), s, q) \mid q \in Q,\ s \in \Sigma \,\} \cup \{\, (i, \varepsilon, q) \mid q \in F \,\}$$

Da die regulären Sprachen genau jene Sprachen sind, die sich durch endliche Automaten darstellen lassen, folgt daraus, dass die regulären Sprachen unter Spiegelung abgeschlossen sind. Das Verfahren kann auf beliebige Automaten angewendet werden, sie müssen nicht deterministisch sein.
Angewendet auf den Beispielautomaten erhalten wir:



Aufgabe 8 (0.3 Punkte)

Um Dollar-Buchungen automatisiert verarbeiten zu können, muss der Geldbetrag folgendermaßen angegeben werden:

- Der Betrag kann beliebig hoch sein, allerdings sind nach je drei Stellen Tausender-Trennzeichen in Form eines Kommas einzufügen, wie zum Beispiel in `34,287,564` .
- Anschließend können optional, durch einen Punkt getrennt, zwei Kommastellen folgen.
- Am Ende steht das Symbol `$`. Dieses kann optional durch ein Leerzeichen vom Geldbetrag getrennt sein.

Beispiele für Geldbeträge: `10,000.00 $`, `10,000.00$`, `10,000$`

(a) Geben Sie einen regulären Ausdruck in algebraischer Notation an, der die Menge aller solchen Geldbeträge beschreibt.

(b) Geben Sie einen Posix Extended Regular Expression an, der alle Zeilen beschreibt, die ausschließlich einen Geldbetrag enthalten.

(c) Zeichnen Sie das Syntaxdiagramm, das Ihrem regulären Ausdruck aus Teil a entspricht.

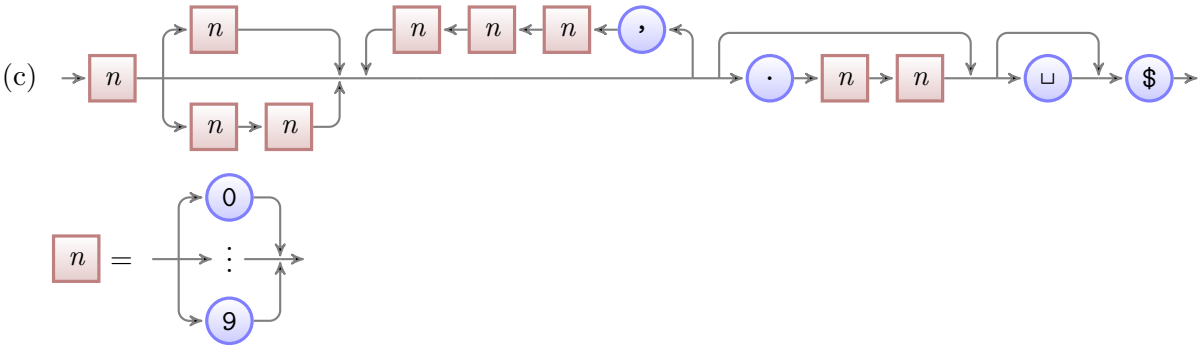
Anmerkung: Moderne Konventionen erlauben den Punkt (vor allem anglo-sächsischer Sprachraum) oder das Komma als Dezimaltrennzeichen. Wenn überhaupt, dürfen als Tausender-Trennzeichen nur Leerzeichen bzw. kleine Abstände verwendet werden.

Lösung

(a) $n(\varepsilon + n + nn)(\texttt{,}\, nnn)^*(\varepsilon + \texttt{.}\, nn)(\varepsilon + \texttt{␣})\texttt{\$}$
wobei n eine Abkürzung ist für $(\texttt{0} + \cdots + \texttt{9})$.

(b) `^[0-9]{1,3}(,[0-9]{3})*(\.[0-9]{2})?␣?\$$`

(c)



Aufgabe 9 (0.3 Punkte)

Konstruieren Sie endliche Automaten, die dieselbe Sprache beschreiben wie die folgenden regulären Ausdrücke.

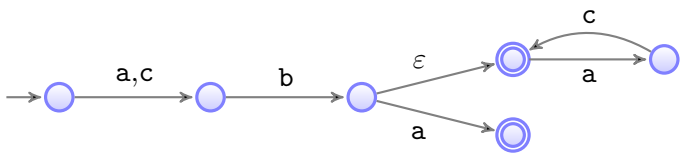
(a) $(\texttt{ab} + \texttt{cb})((\texttt{ac})^* + \texttt{a})$

(b) $(\texttt{b} + \texttt{c})^*(\texttt{b}(\texttt{a} + \varepsilon))^+$

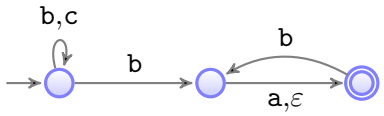
Lösung

Die gesuchten Automaten können mit dem allgemeinen Verfahren konstruiert werden, enthalten dann aber in der Regel viel mehr Zustände und ε-Kanten als notwendig. Die folgenden Automaten wurden bereits vereinfacht.

(a)

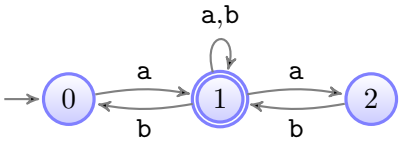


(b)



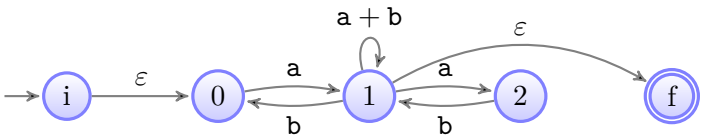
Aufgabe 10 (0.3 Punkte)

Konstruieren Sie zu folgendem endlichen Automaten einen regulären Ausdruck. Orientieren Sie sich am Algorithmus, der in der Vorlesung besprochen wurde und geben Sie den Automaten nach jeder Zustandselimination an!



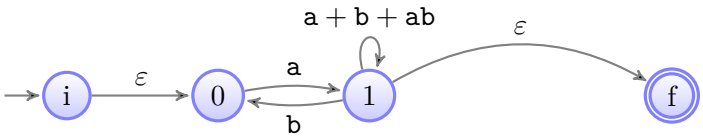
Lösung

Neuer Anfangs- und Endzustand:

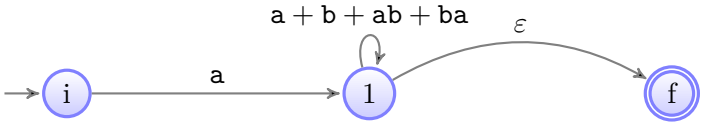


Wir eliminieren die Zustände in der Reihenfolge 2, 0 und 1; andere Reihenfolgen sind ebenfalls möglich.

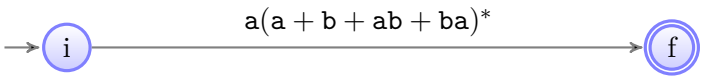
Elimination von Zustand 2:



Elimination von Zustand 0:



Elimination von Zustand 1:



Da $(\mathtt{a}+\mathtt{b})^*$ bereits alle möglichen Wörter über $\{\mathtt{a},\mathtt{b}\}$ enthält, gilt $(\mathtt{a}+\mathtt{b}+\cdots)^* = (\mathtt{a}+\mathtt{b})^*$, der reguläre Ausdruck vereinfacht sich damit zu $\mathtt{a}(\mathtt{a}+\mathtt{b})^*$.

Aufgabe 11 (0.3 Punkte)

Ein oftmals in Songtexten verwendetes Füllwort ist „Schubbidubbidu“ oder „Schubbidubbiduahhhhh“. Die Grammatik $G = \langle N, T, P, S\rangle$ erzeugt solche Füllwörter, wobei

$$
\begin{aligned}
N &= \{S, T, U, V\} \\
T &= \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{h}, \mathtt{i}, \mathtt{s}, \mathtt{u}\} \\
P &= \{\, S \to \mathtt{schub}\,T \ , \\
&\qquad T \to \mathtt{bi}\,U \mid \mathtt{schub}\,T \ , \\
&\qquad U \to \mathtt{dub}\,T \mid V \mid T \ , \\
&\qquad V \to \mathtt{du}\,V \mid \mathtt{ahhhhh} \mid \varepsilon \ \}
\end{aligned}
$$

(a) Überprüfen Sie für die nachfolgenden Wörter, ob sie in der von der Grammatik G spezifizierten Sprache $\mathcal{L}(G)$ liegen. Falls ja, geben Sie eine Ableitung an. Falls nein, argumentieren Sie, warum nicht:

(1) `schubbidududu`

(2) `schubdubbiduahhhhh`

(3) `schubbidubschubbischubschubbidubbiduduahhhhh`

(b) Sind folgende Aussagen über die Sprache $\mathcal{L}(G)$ korrekt? Wenn ja, warum? Wenn nein, geben Sie ein Gegenbeispiel!

(1) In jedem Wort der Sprache ist die Anzahl der `schub`-Silben ungerade.

(2) Jedes Wort enthält mindestens ein `bi`.

(3) Man kann beliebig lange Wörter bilden, in denen keine zwei gleichen Silben (`schub`, `bi`, `dub`, `du` oder `ahhhhh`) aufeinander folgen.

(4) In jedem Wort ist die Anzahl der `dub` kleiner als die Anzahl der `bi`.

(c) Geben Sie einen endlichen Automat für die Sprache $\mathcal{L}(G)$ an.

Lösung

(a) (1) Ja, das Wort liegt in der Sprache $\mathcal{L}(G)$:

$$\begin{aligned} S &\Rightarrow \texttt{schub}\,T \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,U \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,V \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{du}\,V \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{du}\,\texttt{du}\,V \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{du}\,\texttt{du}\,\texttt{du}\,V \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{du}\,\texttt{du}\,\texttt{du}\,\varepsilon \\ &= \texttt{schub}\,\texttt{bi}\,\texttt{du}\,\texttt{du}\,\texttt{du} \end{aligned}$$

(2) Das Wort ist nicht Teil der Sprache $\mathcal{L}(G)$. Das Teilwort `schub` wird durch die Produktionen für S bzw. T eingeführt. In beiden Fällen folgt das Nonterminal T, das entweder durch $\texttt{bi}\,U$ oder $\texttt{schub}\,T$ ersetzt werden kann. Damit muss aber jedem `schub` entweder `bi` oder ein weiteres `schub` folgen. Im vorliegenden Wort folgt auf `schub` aber ein `d`, es kann also nicht mit dieser Grammatik generiert werden.

(3) Ja, das Wort liegt in der Sprache $\mathcal{L}(G)$:

$$\begin{aligned} S &\Rightarrow \texttt{schub}\,T \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,U \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,T \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,T \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,U \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,T \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,T \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,\texttt{schub}\,T \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,\texttt{schub}\,\texttt{bi}\,U \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,\texttt{schub}\,\texttt{bi}\,\texttt{dub}\,T \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,\texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{bi}\,U \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,\texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{bi}\,V \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,\texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{bi}\,\texttt{du}\,V \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,\texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{bi}\,\texttt{du}\,\texttt{du}\,V \\ &\Rightarrow \texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{schub}\,\texttt{bi}\,\texttt{schub}\,\texttt{schub}\,\texttt{bi}\,\texttt{dub}\,\texttt{bi}\,\texttt{du}\,\texttt{du}\,\texttt{ahhhhh} \end{aligned}$$

(b) (1) Falsch. Z.B. ist `schubschubbi` ein Wort der Sprache $\mathcal{L}(G)$ mit einer geraden Anzahl an `schub`-Silben.

(2) Richtig. Jede Ableitung beginnt mit

$$S \Rightarrow \texttt{schub}\,T \Rightarrow \cdots \Rightarrow (\texttt{schub})^n\,T \Rightarrow (\texttt{schub})^n\,\texttt{bi}\,U$$

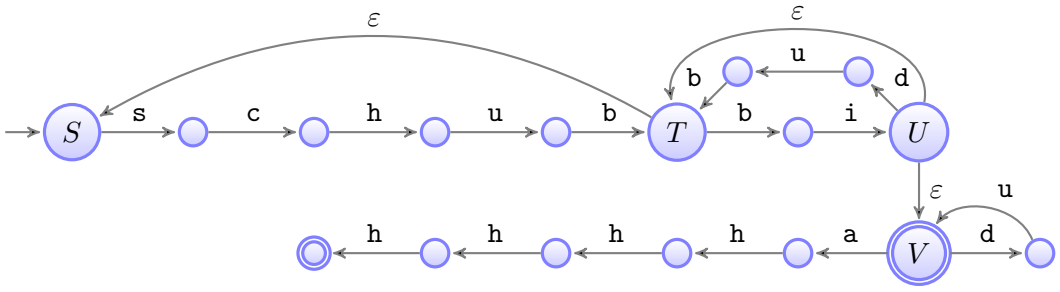
für $n \geq 1$. Somit enthält jedes ableitbare Wort die Silbe `bi`.

(3) Richtig. Z.B. ist das Wort $(\texttt{schub bi})^n$ für jedes $n \geq 1$ Teil der Sprache $\mathcal{L}(G)$:

$$\begin{aligned} S &\Rightarrow \texttt{schub}\,T \Rightarrow \texttt{schub bi}\,U \\ &\Rightarrow \texttt{schub bi}\,T \Rightarrow \texttt{schub bi schub}\,T \Rightarrow \texttt{schub bi schub bi}\,U \\ &\Rightarrow \cdots \Rightarrow (\texttt{schub bi})^n\,U \\ &\Rightarrow (\texttt{schub bi})^n\,V \Rightarrow (\texttt{schub bi})^n\,\varepsilon = (\texttt{schub bi})^n \end{aligned}$$

(4) Richtig. Enthält ein Wort der Sprache die Silbe `dub` n Mal, dann kommt die Silbe `bi` mindestens $n+1$ Mal vor. Das lässt sich folgendermaßen sehen. Die Silbe `dub` kann nur durch die Produktion $U \to \texttt{dub}\,T$ in ein Wort gelangen. Das Nonterminal U wiederum kann nur durch die Produktion $T \to \texttt{bi}\,U$ entstehen. Somit steht vor `dub` immer die Silbe `bi`. Weiters kann jedes Wort nur über V-Produktionen beendet werden. V kann aber nur über die Ableitungsschritte $w\,T \Rightarrow w\,\texttt{bi}\,U \Rightarrow w\,\texttt{bi}\,V$ erreicht werden, d.h., nach dem letzten `dub` kommt mindestens noch ein weiteres `bi`.

(c) Da das Alphabet laut Angabe aus einzelnen Buchstaben besteht und Übergänge nur für einzelne Symbole definiert sind, müssen wir die Silben mit Hilfe von zusätzlichen Zuständen in einzelne Buchstaben zerlegen.



Aufgabe 12 (0.3 Punkte)

DATALOG-Programme besitzen folgenden Aufbau.

- Ein *Programm* ist eine möglicherweise leere Folge von Klauseln. Eine *Klausel* ist entweder ein Faktum oder eine Regel.
- Ein *Faktum* besteht aus einer Atomformel gefolgt von einem Punkt.
- Eine *Regel* besteht aus einer Atomformel, gefolgt von den Zeichen `:-` sowie einer nicht-leeren Liste von Atomformeln, die durch Kommas (`,`) getrennt werden. Regeln enden ebenfalls mit einem Punkt.

- Eine *Atomformel* ist ein Name, dem optional eine in runden Klammern eingeschlossene Argumentliste folgen kann.
- Eine *Argumentliste* ist eine nicht-leere Folge von Namen und Variablen in beliebiger Reihenfolge, die voneinander durch Kommas getrennt werden.
- Ein *Name* ist eine nicht-leere Folge von Buchstaben und Ziffern, die mit einem Kleinbuchstaben beginnt.
- Eine *Variable* ist eine nicht-leere Folge von Buchstaben und Ziffern, die mit einem Großbuchstaben beginnt.

Das folgende Beispiel besteht aus zwei Fakten und drei Regeln; `adam`, `seth`, `istKindVon` usw. sind Namen, `X` und `Y` sind Variablen.

```
istKindVon(seth,adam).
istKindVon(enosh,seth).
istNachfahreVon(X,Y) :- istKindVon(X,Y).
istNachfahreVon(X,Z) :- istKindVon(X,Y), istNachfahreVon(Y,Z).
istMensch(X) :- istNachfahreVon(X,adam).
```

Beschreiben Sie die zulässigen DATALOG-Programme mittels einer kontextfreien Grammatik. Verwenden Sie so weit als möglich EBNF-Notationen, um die Grammatik übersichtlich zu halten und rekursive Regeln zu vermeiden.

Lösung

$\langle N, T, P, \mathit{Programm}\rangle$, wobei

$$
\begin{aligned}
N &= \{\mathit{Programm}, \mathit{Klausel}, \mathit{Faktum}, \mathit{Regel}, \mathit{Atom}, \dots\},\\
T &= \{\cdots \textit{alle Zeichen in den Produktionen zwischen Anführungszeichen} \cdots\},\\
P &= \{\ \mathit{Programm} \to \{\mathit{Klausel}\},\\
&\qquad \mathit{Klausel} \to \mathit{Faktum} \mid \mathit{Regel},\\
&\qquad \mathit{Faktum} \to \mathit{Atom}\texttt{"."},\\
&\qquad \mathit{Regel} \to \mathit{Atom}\texttt{":-"}\mathit{Atom}\{\texttt{","}\mathit{Atom}\}\texttt{"."},\\
&\qquad \mathit{Atom} \to \mathit{Name}[\texttt{"("}\mathit{Argliste}\texttt{")"}],\\
&\qquad \mathit{Argliste} \to \mathit{Arg}\{\texttt{","}\mathit{Arg}\},\\
&\qquad \mathit{Arg} \to \mathit{Name} \mid \mathit{Var},\\
&\qquad \mathit{Name} \to \mathit{KB}\{\mathit{Alphanum}\},\\
&\qquad \mathit{Var} \to \mathit{GB}\{\mathit{Alphanum}\},\\
&\quad \mathit{Alphanum} \to \mathit{KB} \mid \mathit{GB} \mid \mathit{Num},\\
&\qquad \mathit{KB} \to \texttt{"a"} \mid \cdots \mid \texttt{"z"},\\
&\qquad \mathit{GB} \to \texttt{"A"} \mid \cdots \mid \texttt{"Z"},\\
&\qquad \mathit{Num} \to \texttt{"0"} \mid \cdots \mid \texttt{"9"}\ \}.
\end{aligned}
$$

Aufgabe 13 (0.3 Punkte)

Seien $K/2$, $Z/1$, $H/1$ und $V/1$ Prädikatensymbole sowie *ork* und *uruk* Konstantensymbole mit folgender Bedeutung:

$K(x, y)$	... x kämpft mit y
$Z(x)$	... x ist ein Zwerg
$H(x)$	... x ist ein Hobbit
$V(x)$	... x verliert einen Kampf
ork	... Ork
uruk	... Uruk-Hai

Drücken Sie die nachfolgenden prädikatenlogischen Formeln als deutsche Sätze aus.

(a) $\exists x\,(Z(x) \wedge V(x))$

(b) $\forall x\,(Z(x) \supset (K(x, ork) \not\equiv K(x, uruk)))$

(c) $\neg\exists x\exists y\,(Z(x) \wedge H(y) \wedge K(x, y) \wedge V(x))$

Drücken Sie die nachfolgenden deutsche Sätze mit Hilfe der angegebenen Prädikaten- und Konstantensymbole in prädikatenlogischen Formeln aus.

(d) Jeder Hobbit kämpft mit einem Uruk-Hai und verliert dabei.

(e) Manche Hobbits kämpfen mit Zwergen, aber verlieren nie.

(f) Alle Zwerge, die Kämpfe verlieren, kämpfen entweder mit Hobbits, oder mit Orks.

Lösung

(a) Manche Zwerge verlieren einen Kampf. / Es gibt einen Zwerg, der einen Kampf verliert.

(b) Alle Zwerge kämpfen mit einem Ork oder mit einem Uruk-Hai, aber nicht mit beiden.

(c) Es gibt keinen Zwerg, der mit einem Hobbit kämpft und verliert.

(d) $\forall x\,(H(x) \supset (K(x, uruk) \wedge V(x)))$

(e) $\exists x\exists y\,(H(x) \wedge Z(y) \wedge K(x, y) \wedge \neg V(x))$

(f) $\forall x\exists y\,((Z(x) \wedge V(x)) \supset ((H(y) \wedge K(x, y)) \not\equiv K(x, ork)))$

Aufgabe 14 (0.3 Punkte)

Seien *Verprügelt*, *Angriffslustig*, *Gallier* und *Römer* Prädikatensymbole und *obelix*, *lacmus* und *cäsar* Konstantensymbole mit folgender Bedeutung:

$Verprügelt(x, y)$	... x verprügelt y	*obelix*	... Obelix
$Angriffslustig(x)$	... x ist angriffslustig	*lacmus*	... Lacmus
$Gallier(x)$	... x ist ein Gallier	*cäsar*	... Cäsar
$Römer(x)$	... x ist ein Römer		

Verwenden Sie diese Symbole, um die beiden nachfolgenden Sätze in prädikatenlogische Formeln zu übersetzen.

(a) Alle Römer werden von Obelix aber nicht von Cäsar verprügelt.

(b) Manche Gallier verprügeln alle angriffslustigen Römer.

Sei weiters folgende Interpretation gegeben:

$$\begin{aligned}
\mathcal{U} &= \{\text{Aerobus, Asterix, Bonus, Cäsar, Lacmus, Obelix, Troubadix, Verleihnix}\} \\
I(Gallier) &= \{\text{Asterix, Obelix, Verleihnix}\} \\
I(Römer) &= \{\text{Aerobus, Bonus, Cäsar}\} \\
I(Verprügelt) &= \{(\text{Asterix, Aerobus}), (\text{Asterix, Lacmus}), (\text{Obelix, Aerobus}), (\text{Obelix, Bonus}), \\
&\qquad (\text{Obelix, Cäsar}), (\text{Obelix, Lacmus}), (\text{Troubadix, Bonus}), (\text{Verleihnix, Lacmus})\} \\
I(lacmus) &= \text{Lacmus}
\end{aligned}$$

Geben Sie an, ob die nachfolgenden Formeln in dieser Interpretation wahr oder falsch sind. Begründen Sie Ihre Antwort mit einem konkreten Beispiel; es ist keine formale Auswertung erforderlich.

(c) $\forall x\, Verprügelt(x, lacmus)$

(d) $\exists x \forall y (Römer(y) \supset Verprügelt(x, y))$

(e) $\exists x \forall y (Römer(x) \wedge (Gallier(y) \supset Verprügelt(y, x)))$

Bestimmen Sie unter Verwendung der Evaluierungsfunktion den Wahrheitswert der Formel

(f) $\forall x (Gallier(x) \supset Verprügelt(x, lacmus))$

Lösung

(a) $\forall x (Römer(x) \supset (Verprügelt(obelix, x) \wedge \neg Verprügelt(cäsar, x)))$

(b) $\exists x \forall y (Gallier(x) \wedge (Angriffslustig(y) \wedge Römer(y) \supset Verprügelt(x, y)))$

(c) Falsch, da z.B. (Aerobus,Lacmus) $\notin I$

(d) Wahr, da Obelix alle Römer verprügelt.

(e) Falsch, da kein Römer existiert, der von allen Galliern verprügelt wird (Lacmus ist kein Römer).

(f) $\mathsf{val}_{I,\sigma}(\forall x(\mathit{Gallier}(x) \supset \mathit{Verprügelt}(x, \mathit{lacmus}))) = 1$
$\iff$ Für alle $\sigma' \overset{x}{\sim} \sigma$ gilt: $\mathsf{val}_{I,\sigma'}(\mathit{Gallier}(x) \supset \mathit{Verprügelt}(x, \mathit{lacmus})) = 1$
$\iff$ Für alle $\sigma' \overset{x}{\sim} \sigma$ gilt:
Wenn $\mathsf{val}_{I,\sigma'}(\mathit{Gallier}(x)) = 1$, dann $\mathsf{val}_{I,\sigma'}(\mathit{Verprügelt}(x, \mathit{lacmus}) = 1$.
$\iff$ Für alle $\sigma' \overset{x}{\sim} \sigma$ gilt:
Wenn $\sigma'(x) \in I(\mathit{Gallier})$, dann $(\sigma'(x), \text{Lacmus}) \in I(\mathit{Verprügelt})$.

Wegen $I(\mathit{Gallier}) = \{\text{Asterix}, \text{Obelix}, \text{Verleihnix}\}$ müssen wir drei Möglichkeiten für $\sigma'(x)$ untersuchen.

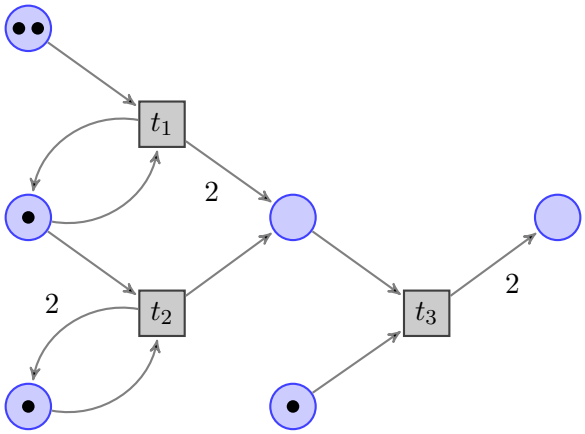
$\sigma'(x)$	$(\sigma'(x), \text{Lacmus}) \in I(\mathit{Verprügelt})$?	
Asterix	$(\text{Asterix}, \text{Lacmus}) \in I(\mathit{Verprügelt})$	✓
Obelix	$(\text{Obelix}, \text{Lacmus}) \in I(\mathit{Verprügelt})$	✓
Verleihnix	$(\text{Verleihnix}, \text{Lacmus}) \in I(\mathit{Verprügelt})$	✓

Die Formel ist daher wahr in der gegebenen Interpretation.

Anmerkung: Unbeabsichtigterweise wurde die Interpretation in der Angabe unvollständig spezifiziert: Den Konstantensymbolen *obelix* und *cäsar* wurden keine Elemente des Universums als ihre Bedeutung zugeordnet. Diese Auslassung spielt für die Lösung der Aufgabe keine Rolle, da die beiden Konstantensymbole nicht benötigt werden; ein Fehler ist es trotzdem.
Umgekehrt ist es aber der Normalfall, dass nicht alle Elemente des Universums in der Formelsprache durch Konstantensymbole repräsentiert sind. Es ist also kein Fehler, dass Verleihnix kein Konstantensymbol besitzt. Betrachtet man beispielsweise Formeln über den reellen Zahlen, so können höchstens Zahlen mit endlich vielen Dezimalstellen als Konstantensymbole angegeben werden. Periodische oder irrationale Zahlen können bestenfalls indirekt durch Gleichungen spezifiziert werden, deren Lösung sie sind. Ein anderes Beispiel sind Listen. Meist gibt es nur ein Konstantensymbol für die leere Liste, alle anderen Listen müssen durch Konstrukturfunktionen beschrieben werden.

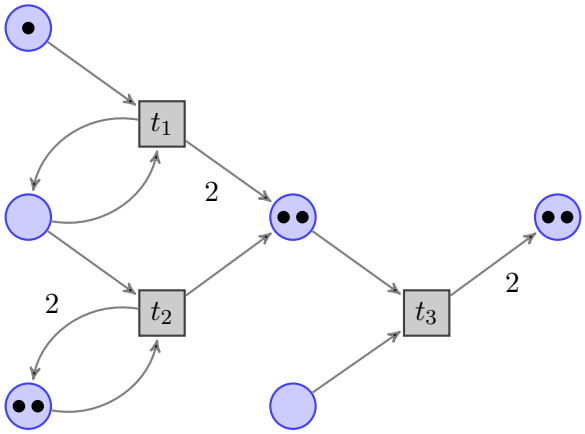
Aufgabe 15 (0.3 Punkte)

Bestimmen Sie eine Folge von Transitionen, die nacheinander feuern können. Die Folge soll jede Transition mindestens einmal enthalten. Nach dem Feuern der letzten Transition soll keine Transition mehr aktiviert sein. Geben Sie die Transitionen dieser Folge sowie die erreichte Endmarkierung an.

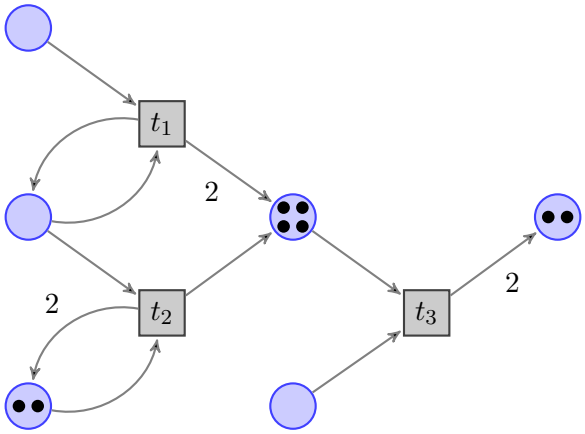


Lösung

Die Transitionsfolgen t_1–t_2–t_3 und t_1–t_3–t_2 liefern:



Die Transitionsfolgen t_1–t_1–t_2–t_3, t_1–t_1–t_3–t_2 und t_1–t_3–t_1–t_2 liefern:

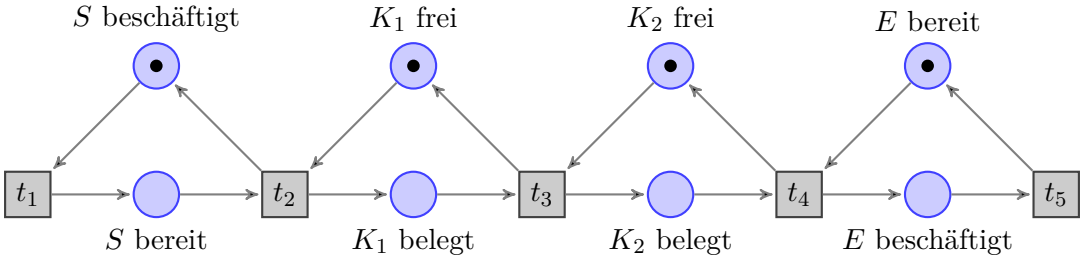


Aufgabe 16 (0.4 Punkte)

Ein Sender und ein Empfänger kommunizieren miteinander über einen Kanal mit Kapazität 2, das heißt, der Kanal ist ein FIFO-Buffer (First-In-First-Out, Queue) mit zwei Plätzen. Wenn der Sender eine Nachricht sendet, belegt diese Platz 1. Nachrichten in Platz 1 werden auf Platz 2 verschoben. Der Empfänger empfängt schließlich die Nachricht, die sich auf Platz 2 befindet.
Der Sender und der Empfänger können sich entweder im Zustand *belegt* oder *bereit* befinden. Der Sender kann nur senden, wenn er sendebereit ist; der Empfänger kann nur empfangen, wenn er empfangsbereit ist und Platz 2 des Kanals eine Nachricht enthält. Nach einer erfolgreichen Sendeoperation (bzw. Empfangsoperation) geht der Sender (bzw.der Empfänger) in den Zustand *belegt* und wechselt nach einer kurzen Zeit wieder in den Zustand *bereit*.

Modellieren Sie das beschriebene System mit Hilfe eines Petri-Netzes.

Lösung



Die beiden Stellen links modellieren den Sender. „S beschäftigt“ bedeutet, dass der Sender die Nachricht erstellt, wohingegen eine Markierung bei „S bereit“ anzeigt, dass die Nachricht sendebereit ist. Die Bereitschaft der Nachricht kann von weiteren Bedingungen abhängig gemacht werden, indem weitere Stellen mit t_1 verbunden werden.
Jeder der beiden Kanal-Plätze wird durch zwei Stellen modelliert. Eine Markierung auf der Stelle „K_i frei“ bzw. „K_i belegt“ zeigt an, dass der Platz i frei bzw. belegt ist.
Analog zeigt eine Markierung bei „E bereit“ bzw. „E beschäftigt“ an, dass der Empfänger bereit zum Nachrichtenempfang bzw. beschäftigt mit der Verarbeitung der letzten Nachricht ist. Die Ankunft einer Nachricht kann weitere Aktionen auslösen, indem t_5 mit weiteren Stellen verbunden wird.
Durch die Art der Verbindungen und die gewählten Markierungen ist sichergestellt, dass eine Nachricht

- ... nur gesendet werden kann, wenn K_1 frei ist;
- ... nur dann von K_1 nach K_2 gelangt, wenn K_2 frei ist;
- ... nur empfangen werden kann, wenn der Empfänger bereit ist.