2. Übungsblatt (mit Lösungen)

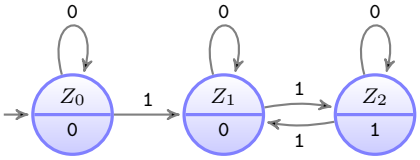
3.0 VU Formale Modellierung

Marion Scholz, Gernot Salzer

November 2014

Aufgabe 1 (0.3 Punkte)

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



(a) Geben Sie die Ausgaben zu folgenden Eingaben an: `00111`, `01111`, `10100`.

(b) Berechnen Sie schrittweise $\delta^*(Z_0, \texttt{01101})$ und $\gamma^*(Z_0, \texttt{01101})$.

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

Lösung

(a)

w:	`00111`	`01111`	`10100`
$[\mathcal{A}](w)$:	`00010`	`00101`	`00111`

(b) $\begin{aligned}\delta^*(Z_0, \texttt{01101}) &= \delta^*(\delta(Z_0, \texttt{0}), \texttt{1101}) &&= \delta^*(Z_0, \texttt{1101}) \\ &= \delta^*(\delta(Z_0, \texttt{1}), \texttt{101}) &&= \delta^*(Z_1, \texttt{101}) \\ &= \delta^*(\delta(Z_1, \texttt{1}), \texttt{01}) &&= \delta^*(Z_2, \texttt{01}) \\ &= \delta^*(\delta(Z_2, \texttt{0}), \texttt{1}) &&= \delta^*(Z_2, \texttt{1}) \\ &= \delta^*(\delta(Z_2, \texttt{1}), \varepsilon) &&= \delta^*(Z_1, \varepsilon) \\ &= Z_1\end{aligned}$

$\begin{aligned}\gamma^*(Z_0, \texttt{01101}) &= \gamma(\delta(Z_0, \texttt{0})) \cdot \gamma^*(\delta(Z_0, \texttt{0}), \texttt{1101}) \\ &= \gamma(Z_0) \cdot \gamma^*(Z_0, \texttt{1101}) \\ &= 0 \cdot \gamma(\delta(Z_0, \texttt{1})) \cdot \gamma^*(\delta(Z_0, \texttt{1}), \texttt{101}) \\ &= 0 \cdot \gamma(Z_1) \cdot \gamma^*(Z_1, \texttt{101}) \\ &= 00 \cdot \gamma(\delta(Z_1, \texttt{1})) \cdot \gamma^*(\delta(Z_1, \texttt{1}), \texttt{01}) \\ &= 00 \cdot \gamma(Z_2) \cdot \gamma^*(Z_2, \texttt{01}) \\ &= 001 \cdot \gamma(\delta(Z_2, \texttt{0})) \cdot \gamma^*(\delta(Z_2, \texttt{0}), \texttt{1}) \\ &= 001 \cdot \gamma(Z_2) \cdot \gamma^*(Z_2, \texttt{1}) \\ &= 0011 \cdot \gamma(\delta(Z_2, \texttt{1})) \cdot \gamma^*(\delta(Z_2, \texttt{1}), \varepsilon) \\ &= 0011 \cdot \gamma(Z_1) \cdot \gamma^*(Z_1, \varepsilon) \\ &= 00110 \cdot \varepsilon \\ &= 00110\end{aligned}$

(c) [$\mathcal{A}$]: Der Automat erkennt eine gerade Anzahl an `1`ern in der Eingabe: War die Anzahl der `1`er bisher gerade (aber größer 0), so ist die Ausgabe `1`, sonst `0`.

Aufgabe 2 (0.3 Punkte)

Sei L die Sprache

$$\{\, w \in \{\texttt{a}, \texttt{i}, \texttt{l}, \texttt{x}\}^* \mid w \text{ enthält das Teilwort } \texttt{lila} \text{ oder } \texttt{ill} \,\} \ .$$

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

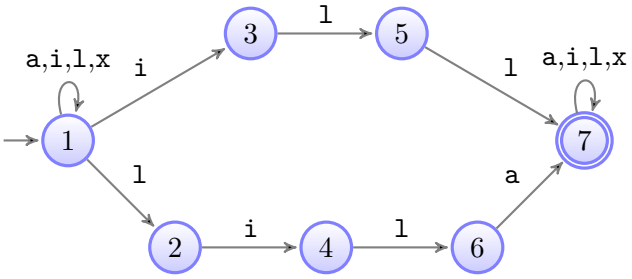
(a) Konstruieren Sie einen indeterministischen Automaten für diese Sprache.

(b) 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 Automat ist im Zustand 1 indeterministisch: Sowohl für das Symbol `i` als auch `l` gibt es zwei mögliche Folgezustände, nämlich 1 und 3 bzw. 1 und 2
Zusätzlich zur graphischen Darstellung geben wir die Übergangsfunktion auch als Tabelle an, da diese bei der Determinisierung hilft.

δ^*	a	i	l	x
1	$\{1\}$	$\{1,3\}$	$\{1,2\}$	$\{1\}$
2	$\{\}$	$\{4\}$	$\{\}$	$\{\}$
3	$\{\}$	$\{\}$	$\{5\}$	$\{\}$
4	$\{\}$	$\{\}$	$\{6\}$	$\{\}$
5	$\{\}$	$\{\}$	$\{7\}$	$\{\}$
6	$\{7\}$	$\{\}$	$\{\}$	$\{\}$
7	$\{7\}$	$\{7\}$	$\{7\}$	$\{7\}$

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 zum jeweiligen Zeitpunkt befinden kann. Der Startzustand wird mit $\{1\}$ bezeichnet, da sich der indeterministische Automat zu Beginn im Zustand 1 (und nur in diesem) befindet. Von diesem Zustand ausgehend erstellen wir zeilenweise die Tabelle für die Übergangsfunktion des deterministischen Automaten.

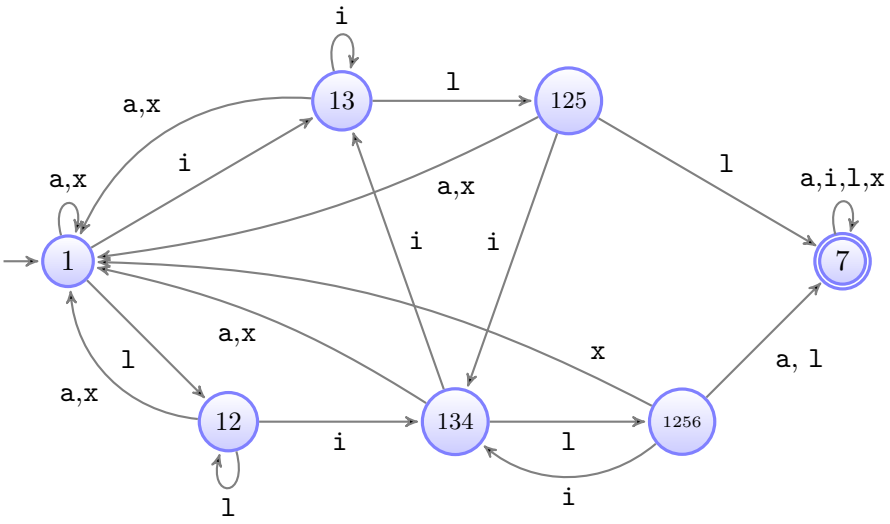
$\hat{\delta}$	a	i	l	x
$\{1\}$	$\{1\}$	$\{1,3\}$	$\{1,2\}$	$\{1\}$
$\{1,2\}$	$\{1\}$	$\{1,3,4\}$	$\{1,2\}$	$\{1\}$
$\{1,3\}$	$\{1\}$	$\{1,3\}$	$\{1,2,5\}$	$\{1\}$
$\{1,3,4\}$	$\{1\}$	$\{1,3\}$	$\{1,2,5,6\}$	$\{1\}$
$\{1,2,5\}$	$\{1\}$	$\{1,3,4\}$	$\{1,2,7\}$	$\{1\}$
$\{1,2,5,6\}$	$\{1,7\}$	$\{1,3,4\}$	$\{1,2,7\}$	$\{1\}$
$\{1,2,7\}$	$\{1,7\}$	$\{1,3,4,7\}$	$\{1,2,7\}$	$\{1,7\}$
$\{1,7\}$	$\{1,7\}$	$\{1,3,7\}$	$\{1,2,7\}$	$\{1,7\}$
$\{1,3,4,7\}$	$\{1,7\}$	$\{1,3,7\}$	$\{1,2,5,6,7\}$	$\{1,7\}$
$\{1,3,7\}$	$\{1,7\}$	$\{1,3,7\}$	$\{1,2,5,7\}$	$\{1,7\}$
$\{1,2,5,6,7\}$	$\{1,7\}$	$\{1,3,4,7\}$	$\{1,2,7\}$	$\{1,7\}$
$\{1,2,5,7\}$	$\{1,7\}$	$\{1,3,4,7\}$	$\{1,2,7\}$	$\{1,7\}$

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 7 enthält. Dieser wird somit durch das Tupel $\langle \widehat{Q}, \{\mathtt{a}, \mathtt{i}, \mathtt{l}, \mathtt{x}\}, \hat{\delta}, \{1\}, \widehat{F}\rangle$ beschrieben, wobei

$$\widehat{F} = \{\{1,2,7\},\{1,7\},\{1,3,4,7\},\{1,3,7\},\{1,2,5,6,7\},\{1,2,5,7\}\}$$
$$\widehat{Q} = \{\{1\},\{1,2\},\{1,3\},\{1,3,4\},\{1,2,5\},\{1,2,5,6\}\} \cup \widehat{F}$$

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 vier Symbole wieder nur Endzustände erreichbar sind. Man kann diese daher zu einem einzigen Endzustand zusammenfassen, ohne die akzeptierte Sprache zu verändern. Der vereinfachte deterministische Automat sieht dann folgendermaßen aus:



Aufgabe 3 (0.3 Punkte)

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

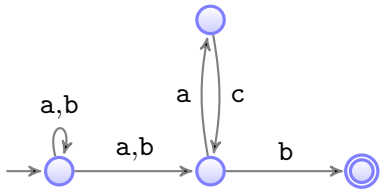
(a) $(\mathtt{a}+\mathtt{b})^+(\mathtt{ac})^*\mathtt{b}$

(b) $(\mathtt{ab}+\mathtt{b})(\mathtt{ab})^+ + \mathtt{a}$

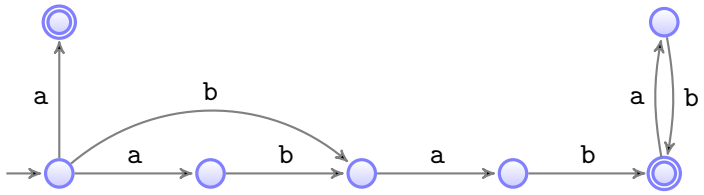
Lösung

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)

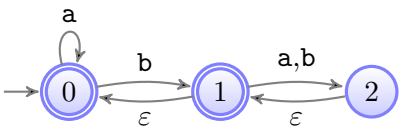


(b)



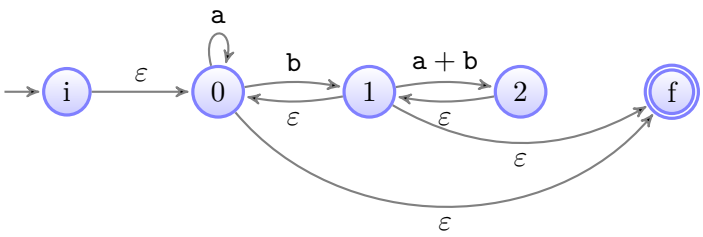
Aufgabe 4 (0.4 Punkte)

Geben Sie einen regulären Ausdruck an, der dieselbe Sprache beschreibt wie der folgende endliche Automat. Verwenden Sie dabei das in der Vorlesung beschriebene Verfahren und geben Sie den Automaten nach jeder Zustandselimination an.



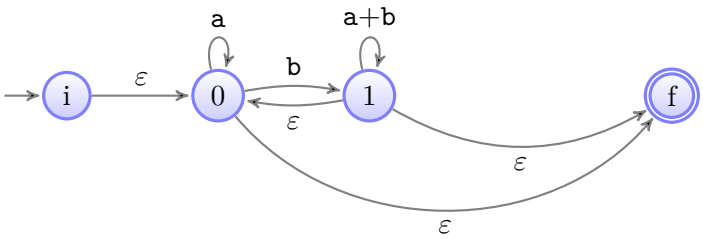
Lösung

Neuer Anfangs- und Endzustand:

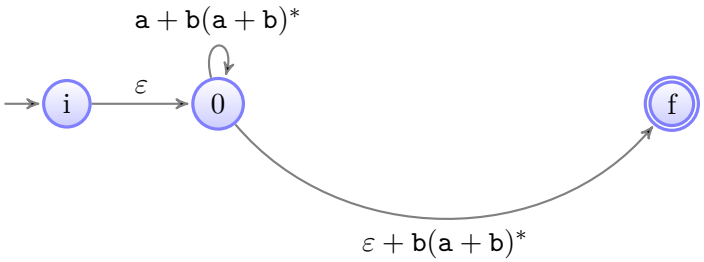


Wir eliminieren die Zustände in der Reihenfolge 2, 1 und 0; andere Reihenfolgen sind ebenfalls möglich.

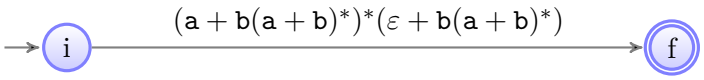
Elimination von Zustand 2:



Elimination von Zustand 1:



Elimination von Zustand 0:



Die Sprache des ursprünglichen Automaten kann daher durch den regulären Ausdruck $(\mathtt{a}+\mathtt{b}(\mathtt{a}+\mathtt{b})^*)^*(\varepsilon+\mathtt{b}(\mathtt{a}+\mathtt{b})^*)$ beschrieben werden.
Fleißaufgabe: Dieser Ausdruck lässt sich noch vereinfachen. Die Sprache $\mathtt{b}(\mathtt{a}+\mathtt{b})^*$ enthält das Wort $\mathtt{b}$, da $(\mathtt{a}+\mathtt{b})^*$ das Leerwort enthält. Somit ist $(\mathtt{a}+\mathtt{b})^*$ jedenfalls eine Teilsprache von $(\mathtt{a}+\mathtt{b}(\mathtt{a}+\mathtt{b})^*)^*$; da $(\mathtt{a}+\mathtt{b})^*$ aber bereits die Menge *aller* Wörter über $\{\mathtt{a},\mathtt{b}\}$ ist, gilt $(\mathtt{a}+\mathtt{b})^* = (\mathtt{a}+\mathtt{b}(\mathtt{a}+\mathtt{b})^*)^*$. Wir erhalten daher:

$$\begin{aligned}(\mathtt{a}+\mathtt{b}(\mathtt{a}+\mathtt{b})^*)^*(\varepsilon+\mathtt{b}(\mathtt{a}+\mathtt{b})^*) &= (\mathtt{a}+\mathtt{b})^*(\varepsilon+\mathtt{b}(\mathtt{a}+\mathtt{b})^*)\\ &= (\mathtt{a}+\mathtt{b})^*\cdot\varepsilon + (\mathtt{a}+\mathtt{b})^*\cdot\mathtt{b}(\mathtt{a}+\mathtt{b})^*\\ &= (\mathtt{a}+\mathtt{b})^* + (\mathtt{a}+\mathtt{b})^*\mathtt{b}(\mathtt{a}+\mathtt{b})^*\end{aligned}$$

Da der erste Ausdruck bereits alle Wörter enthält, trägt der zweite Ausdruck nichts mehr bei und kann gestrichen werden:

$$= (\mathtt{a}+\mathtt{b})^*$$

Aufgabe 5 (0.3 Punkte)

An der TU beschäftigte Personen erhalten Mailadressen der Form

Vorname`.`*Mittelteil*`.`*Zuname*`+`*OrgEinheit*`@tuwien.ac.at`

wobei die Teile „`.`*Mittelteil*“ und „`+`*OrgEinheit*“ optional sind. Die Namensteile bestehen aus Groß- und Kleinbuchstaben, wobei Umlaute und scharfes S durch ae, oe, ue, ss etc. zu ersetzen sind. *OrgEinheit* ist die Kennung des Instituts oder der Serviceeinrichtung, der die Person angehört; sie besteht aus dem Zeichen `E` gefolgt von drei Ziffern und einem optionalen Großbuchstaben. Beispiele:

```
Hikaru.Sulu@tuwien.ac.at
James.Tiberius.Kirk@tuwien.ac.at
Beverly.Crusher+E010C@tuwien.ac.at
Nyota.Penda.Uhura+E006@tuwien.ac.at
```
[1]

[1] Wer von diesen vier Personen passt nicht zu den anderen?

(a) Geben Sie einen regulären Ausdruck in algebraischer Notation an, der die Menge derartiger Emailadressen beschreibt.

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

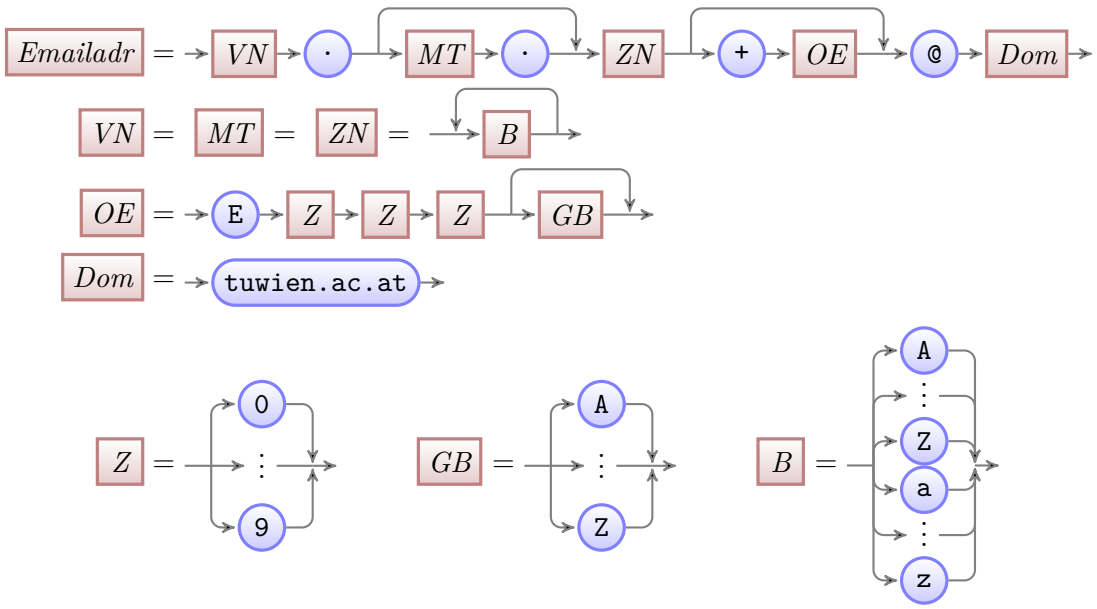
(c) Geben Sie das Syntaxdiagramm an, das Ihrem regulären Ausdruck aus Teilaufgabe a entspricht.

Lösung

(a) $BB^*(\varepsilon + \texttt{.}BB^*)\texttt{.}BB^*(\varepsilon + (\texttt{+}EZZZ(\varepsilon + G)))\texttt{@tuwien.ac.at}$ wobei Z eine Abkürzung für $(\texttt{0} + \cdots + \texttt{9})$, G eine für $(\texttt{A} + \cdots + \texttt{Z})$ und B eine für $(\texttt{A} + \cdots + \texttt{Z} + \texttt{a} + \cdots + \texttt{z})$ ist.

(b) `^[A-Za-z]+(\.[A-Za-z]+)?\.[A-Za-z]+(\+E[0-9]{3}[A-Z]?)?@tuwien\.ac\.at$`

(c)



Aufgabe 6 (0.3 Punkte)

Bertram der Biber friert im Winter regelmäßig, er zittert und bibbert. Seine Mitbiber geben ihm daher Namen wie bibberbibberbibbernderbiber, bibbernderbibberbibberbibberbibberbiber, oder einfach nur bibberbiber. Einem Biberforscher gelingt es, diese

Bibbersprache durch die Grammatik $G = \langle N, T, P, A \rangle$ zu beschreiben, wobei

$$
\begin{aligned}
N &= \{A, B, C, D, E\} \\
T &= \{\texttt{b}, \texttt{d}, \texttt{e}, \texttt{i}, \texttt{n}, \texttt{r}\} \\
P &= \{\, A \to \texttt{bib}\, B \\
&\qquad B \to \texttt{ber}\, C \mid \texttt{er} \\
&\qquad C \to \texttt{bib}\, B \mid \texttt{nder}\, D \\
&\qquad D \to \texttt{bib}\, E \\
&\qquad E \to \texttt{ber}\, D \mid \texttt{er} \,\}
\end{aligned}
$$

(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 Parallelableitung an. Falls nein, argumentieren Sie, warum nicht.

(1) `bibbernderbibberbibber`

(2) `bibberbibbernderbiber`

(3) `bibbernderbibbernderbiber`

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

(1) Jedes Wort endet mit `biber`.

(2) Die Anzahl der `bibber` vor bzw. nach `nder` im Wort sind gleich.

(3) Jedes Wort beginnt mit `bib`.

(c) Konstruieren Sie einen endlichen Automat, der die Sprache $\mathcal{L}(G)$ akzeptiert.

Lösung

(a) (1) Das Wort ist nicht Teil der Sprache $\mathcal{L}(G)$, da jedes Wort dieser Sprache mit `biber` enden muss. Das lässt sich folgendermaßen sehen.

Jede Ableitung endet durch Anwendung der Produktion $B \to \texttt{er}$ oder $E \to \texttt{er}$. Die Nonterminale B und E treten aber immer nur unmittelbar nach der Symbolfolge `bib` auf. Somit endet jede Ableitung auf eine der folgenden Arten:

$$
\begin{aligned}
\cdots A &\Rightarrow \cdots \texttt{bib}\, B \Rightarrow \cdots \texttt{biber} \\
\cdots C &\Rightarrow \cdots \texttt{bib}\, B \Rightarrow \cdots \texttt{biber} \\
\cdots D &\Rightarrow \cdots \texttt{bib}\, E \Rightarrow \cdots \texttt{biber}
\end{aligned}
$$

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

$$\begin{aligned} A &\Rightarrow_P \texttt{bib}\, B \\ &\Rightarrow_P \texttt{bib}\,\texttt{ber}\, C \\ &\Rightarrow_P \texttt{bib}\,\texttt{ber}\,\texttt{bib}\, B \\ &\Rightarrow_P \texttt{bib}\,\texttt{ber}\,\texttt{bib}\,\texttt{ber}\, C \\ &\Rightarrow_P \texttt{bib}\,\texttt{ber}\,\texttt{bib}\,\texttt{ber}\,\texttt{nder}\, D \\ &\Rightarrow_P \texttt{bib}\,\texttt{ber}\,\texttt{bib}\,\texttt{ber}\,\texttt{nder}\,\texttt{bib}\, E \\ &\Rightarrow_P \texttt{bib}\,\texttt{ber}\,\texttt{bib}\,\texttt{ber}\,\texttt{nder}\,\texttt{bib}\,\texttt{er} \end{aligned}$$

(3) Das Wort ist nicht Teil der Sprache $\mathcal{L}(G)$, da es zwei `n` enthält, jedes Wort der Sprache aber höchstens ein `n` besitzt. Das lässt sich folgerndermaßen argumentieren.

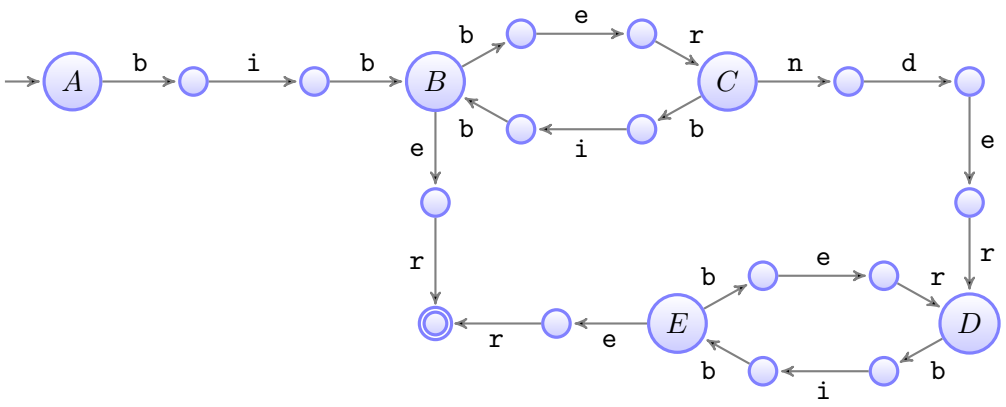
Zunächst gilt, dass `n` nur durch die Produktion $C \to \texttt{nder}\, D$ eingeführt werden kann. Für D gibt es aber nur die Produktion $D \to \texttt{bib}E$, für E nur die Produktionen $E \to \texttt{ber}D$ und $E \to \texttt{er}$; das heißt, dass durch D kein weiteres `n` eingeführt werden kann. Da nach jedem Ableitungsschritt höchstens ein Nonterminal im Ausdruck vorkommt, daher insbesondere höchstens ein C, enthält das erzeugte Wort der Sprache maximal ein `n`.

(b) (1) Richtig. Argumentation siehe die Lösung zu Aufgabe a1.

(2) Falsch. Ein Gegenbeispiel ist das Wort aus Aufgabe a2, das in der Sprache liegt und zwei `bibber` vor, aber kein `bibber` nach `nder` aufweist.

(3) Richtig. Die einzige Produktion für das Startsymbol A ersetzt es durch `bib` B, daher fängt jedes Wort der Sprache mit `bib` an.

(c) Im Wesentlichen werden die Nonterminale zu den Zuständen des Automaten. Weiters müssen wir, da das Alphabet aus einzelnen Buchstaben besteht und Übergänge nur für einzelne Symbole definiert sind, die Silben mit Hilfe von zusätzlichen Zuständen in einzelne Buchstaben zerlegen.



Aufgabe 7 (0.4 Punkte)

In Büchern über formale Sprachen ist folgende Definition zu finden:

> Eine *reguläre Grammatik* wird durch ein 4-Tupel $G = \langle V, T, P, S \rangle$ festgelegt, wobei
>
> - V und T endliche, disjunkte Mengen von Symbolen sind ($V \cap T = \{\}$),
> - S ein Symbol aus V ist ($S \in V$) und
> - $P \subseteq V \times (T \cdot V \cup \{\varepsilon\})$ eine endliche Menge von Paaren ist.
>
> Die Elemente von P werden Produktionen genannt; statt $(x, y) \in P$ wird auch $x \to y$ geschrieben. Die Notation $x \to y_1 \mid \cdots \mid y_n$ ist eine Abkürzung für die Produktionen $x \to y_1$, …, $x \to y_n$.
>
> Das Wort $u\,y\,v$ ist aus dem Wort $u\,x\,v$ in einem Schritt ableitbar, geschrieben $u\,x\,v \Rightarrow u\,y\,v$, wenn $x \to y$ gilt. Die von G generierte Sprache $\mathcal{L}(G)$ ist definiert als die Menge $\{\, w \in T^* \mid S \stackrel{*}{\Rightarrow} w \,\}$, wobei $\stackrel{*}{\Rightarrow}$ den reflexiven und transitiven Abschluss von $\Rightarrow$ bezeichnet.[2]

(a) Geben Sie an, welche der folgenden Tupeln eine reguläre Grammatik gemäß der obigen Definition darstellt. Begründen Sie Ihre Antwort, falls es sich um keine reguläre Grammatik handelt. Entspricht das Tupel der Definition, geben Sie die Sprache an, die durch die Grammatik generiert wird.

(1) $\langle \{\mathtt{X}\}, \{\mathtt{a}, \mathtt{b}\}, \{\mathtt{X} \to \mathtt{aXb} \mid \varepsilon\}, \mathtt{X} \rangle$

(2) $\langle \{\mathtt{X}, \mathtt{Y}\}, \{\mathtt{a}, \mathtt{b}\}, \{\mathtt{X} \to \mathtt{aX} \mid \mathtt{bY} \mid \varepsilon, \mathtt{Y} \to \mathtt{bY}\}, \mathtt{X} \rangle$

(3) $\langle \{\mathtt{a}, \mathtt{b}\}, \{\mathtt{X}, \mathtt{Y}\}, \{\mathtt{a} \to \mathtt{Xa} \mid \mathtt{Yb} \mid \varepsilon, \mathtt{b} \to \mathtt{Yb}\}, \mathtt{b} \rangle$

(4) $\langle \{\mathtt{X}, \mathtt{Y}\}, \{\mathtt{a}, \mathtt{b}\}, \{\mathtt{X} \to \mathtt{Xa} \mid \mathtt{Yb} \mid \varepsilon, \mathtt{Y} \to \mathtt{Yb} \mid \varepsilon\}, \mathtt{X} \rangle$

(b) Beschreiben Sie ein Verfahren, das zu einer regulären Grammatik einen äquivalenten endlichen Automaten liefert.

(c) Beschreiben Sie ein Verfahren, das zu einem endlichen Automaten ohne ε-Übergängen eine äquivalente reguläre Grammatik liefert. Lässt sich Ihr Verfahren auch für Automaten mit ε-Übergängen verwenden?

---

[2] Das heißt, dass $\stackrel{*}{\Rightarrow}$ die kleinste Relation mit folgenden Eigenschaften ist:

- Aus $u \Rightarrow v$ folgt $u \stackrel{*}{\Rightarrow} v$.
- Es gilt $u \stackrel{*}{\Rightarrow} u$ für alle Wörter $u \in T^*$.
- Aus $u \stackrel{*}{\Rightarrow} v$ und $v \stackrel{*}{\Rightarrow} w$ folgt $u \stackrel{*}{\Rightarrow} w$.

Anschaulich gesprochen steht $\stackrel{*}{\Rightarrow}$ für die Ableitbarkeit in beliebig vielen Schritten.

Lösung

(a) i. Ist keine reguläre Grammatik, da die rechte Seite der ersten Produktion, `aXb`, in der Menge $T \cdot V \cdot T$ liegt, nicht aber in der Menge $T \cdot V$. (Offensichtlich ist `aSb` auch nicht das Leerwort.)

ii. Ist eine reguläre Grammatik, die die Sprache $\{\texttt{a}\}^*$ generiert.

iii. Ist eine reguläre Grammatik, die die Sprache $\{\}$ generiert.

iv. Ist keine reguläre Grammatik, da etwa die rechte Seite der ersten Produktion, `Xa`, in der Menge $V \cdot T$ liegt, nicht aber in der Menge $T \cdot V$.

(b) Wir interpretieren die Variablensymbole als Zustände und führen für jede Produktion $A \to sB$ einen Übergang mit dem Symbol s von Zustand A nach Zustand B ein. A ist Endzustand genau dann, wenn es die Produktion $A \to \varepsilon$ gibt.

Formal lässt sich der Automat $\mathcal{A}_G = \langle Q, \Sigma, \delta, q_0, F \rangle$ zu einer regulären Grammatik $G = \langle V, T, P, S \rangle$ folgendermaßen definieren.

$$\begin{aligned} Q &= V \\ \Sigma &= T \\ \delta &= \{\, (A, s, B) \mid (A \to sB) \in P \,\} \\ q_0 &= S \\ F &= \{\, A \mid (A \to \varepsilon) \in P \,\} \end{aligned}$$

(c) Analog lässt sich zu jedem Automaten $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ ohne ε-Übergängen eine reguläre Grammatik $G_\mathcal{A} = \langle V, T, P, S \rangle$ definieren, indem man die Zustände zu Nonterminalen und die Übergänge zu Produktionen macht:

$$\begin{aligned} V &= Q \\ T &= \Sigma \\ P &= \{\, (q \to sq') \mid (q, s, q') \in \delta \,\} \cup \{\, (q \to \varepsilon) \mid q \in F \,\} \\ S &= q_0 \end{aligned}$$

Aufgabe 8 (0.4 Punkte)

Angenommen es sind zwei deterministische Automaten $\mathcal{A}$ und $\mathcal{B}$ gegeben. Geben Sie ein allgemeines Verfahren an um einen endlichen Automaten zu konstruieren, der genau jene Wörter akzeptiert, die von $\mathcal{A}$ aber nicht von $\mathcal{B}$ akzeptiert werden. Welche Eigenschaft regulärer Sprachen lässt sich daraus ablesen?

Lösung

Seien $\mathcal{A} = \langle Q_1, \Sigma, \delta_1, i_1, F_1 \rangle$ und $\mathcal{B} = \langle Q_2, \Sigma, \delta_2, i_2, F_2 \rangle$ zwei beliebige deterministische Automaten über dem Alphabet Σ. Wir konstruieren einen Automaten $\mathcal{C}$, der die beiden Automaten $\mathcal{A}$ und $\mathcal{B}$ gleichzeitig ausführt. Als Zustände für $\mathcal{C}$ verwenden wir Paare

(q_1, q_2), wobei $q_1 \in Q_1$ ein Zustand des ersten Automaten und $q_2 \in Q_2$ ein Zustand des zweiten Automaten ist. Der neue Automat befindet sich bei Eingabe eines Wortes w im Zustand (q_1, q_2), wenn sich der erste Automat bei diesem Wort im Zustand q_1 und der zweite im Zustand q_2 befinden würde. Der Startzustand (i_1, i_2) entspricht der Situation, in der sich die beiden ursprünglichen Automaten im Startzustand befinden. Ein Übergang mit dem Symbol s von (q_1, q_2) nach (q_1', q_2') existiert genau dann, wenn man mit diesem Symbol in $\mathcal{A}$ von q_1 nach $q_1' = \delta_1(q_1, s)$ und in $\mathcal{B}$ von q_2 nach $q_2' = \delta_2(q_2, s)$ gelangt.
Uns interessieren nun alle Wörter, die von $\mathcal{A}$ aber nicht von $\mathcal{B}$ akzeptiert werden. Ein derartiges Wort liegt genau dann vor, wenn der neue Automat damit einen Zustand (q_1, q_2) erreicht, bei dem q_1 ein Endzustand des Automaten $\mathcal{A}$ aber q_2 kein Endzustand des Automaten $\mathcal{B}$ ist. Der gesucht Automat ist somit gegeben durch

$$\mathcal{C} = \langle Q_1 \times Q_2, \Sigma, \delta, (i_1, i_2), F_1 \times (Q_2 \setminus F_2) \rangle \ ,$$

wobei die Übergangsfunktion festgelegt ist durch

$$\delta((q_1, q_2), s) = (\delta_1(q_1, s), \delta_2(q_2, s))$$

für alle $(q_1, q_2) \in Q_1 \times Q_2$ und alle $s \in \Sigma$.
Die Menge der Wörter, die von $\mathcal{A}$ aber nicht von $\mathcal{B}$ akzeptiert werden, ist nichts anderes als die Mengendifferenz $\mathcal{L}(\mathcal{A}) \setminus \mathcal{L}(\mathcal{B})$. Unsere Konstruktion zeigt, dass die Differenz von Sprachen, die von Automaten akzeptiert werden, wieder durch einen Automaten beschrieben werden kann. Die Familie der Sprachen, die von endlichen Automaten akzeptiert werden, ist daher abgeschlossen gegenüber Mengendifferenz. In der Vorlesung haben wir gezeigt, dass diese Sprachfamilie genau den regulären Sprachen entspricht. Daher sind auch die regulären Sprachen abgeschlossen gegenüber Differenzbildung: Die Differenz zweier regulärer Sprachen ist wieder regulär.

Aufgabe 9 (0.3 Punkte)

Die *Planning Domain Definition Language* (PDDL) wird zur Beschreibung von Planungsproblemen verwendet. Das folgende PDDL-Beispiel beschreibt ein Problem namens „`Blocksworld-Instanz-1`“ aus dem Bereich „`Blocksworld`“, in dem es die Objekte „`A`“, „`B`“ und „`C`“ vom Typ „`Block`“ sowie das Objekt „`D`“ vom Typ „`Cube`“ gibt; der „`:init`“-Abschnitt beschreibt die gegebene Ausgangssituation, der „`:goal`“-Abschnitt die gewünschte Zielsituation.

```
(define (problem Blocksworld-Instanz-1)
        (:domain Blocksworld)
        (:objects A B C - Block D - Cube)
        (:init (Clear A) (Clear C) (On C B) (OnTable A) (OnTable B) (HandEmpty) )
        (:goal (OnTable C) (On A B) (On B C) )
)
```

Grundsätzlich bestehen Problembeschreibungen aus ineinander geschachtelten Listen, die jeweils in runden Klammern eingeschlossen sind. Die Listenelemente werden durch Leerzeichen getrennt. Es genügt, wenn Sie einzelne Leerzeichen zur Trennung vorsehen, mehrfache Leerzeichen und Zeilenumbrüche müssen nicht berücksichtigt werden.
Die oberste Liste besteht aus sechs Elementen, nämlich dem Schlüsselwort „`define`“ gefolgt von Listen für die Problembezeichnung, die Bereichsbezeichnung, die Objektdefinitionen, die Ausgangs- und die Zielsituation (wie im Beispiel oben).
„`define`“, „`problem`“, „`:domain`“, „`:objects`“, „`:init`“ und „`:goal`“ sind Schlüsselwörter. Alle anderen Bezeichnungen bestehen aus Buchstaben, Ziffern und Bindestrichen, beginnen aber immer mit einem Buchstaben.
Objekte werden im „`:objects`“-Abschnitt definiert. Der Liste von Objekten folgt getrennt durch einen Bindestrich ihr Typ. Darauf können weitere Objekte mit einer weitere Typangabe (im Beispiel oben „`Cube`“) folgen, und so weiter.
Der „`:init`“- und der „`:goal`“-Abschnitt besteht jeweils aus einer Folge von sogenannten Prädikaten. Jedes Prädikat ist eine Liste, die mit einem Prädikatnamen beginnt, dem Argumente folgen können. Etwa enthält das Prädikat „`(Clear A)`“ den Prädikatnamen „`Clear`“ und das Argument „`A`“. „`(HandEmpty)`“ enthält nur einen Prädikatnamen, aber keine Argumente. Im Prädikat „`(On A B)`“ folgen dem Prädikatnamen zwei Argumente.
Geben Sie eine strukturierte kontextfreie Grammatik in EBNF an, die den Aufbau derartiger Problembeschreibungen spezifiziert.

Lösung

$\langle V, T, P, Pddl \rangle$ mit

$V = \{Pddl, Problem, Domain, \ldots, Char, Letter, Digit\}$,

$T = \{\texttt{"␣"}, \texttt{"("}, \texttt{")"}, \texttt{"-"}, \texttt{":"}, \texttt{"a"}, \ldots, \texttt{"z"}, \texttt{"A"}, \ldots, \texttt{"Z"}, \texttt{"0"}, \ldots, \texttt{"9"}\}$,

$P = \{$

$Pddl \rightarrow \texttt{"(define␣"}\ Problem\ \texttt{"␣"}\ Domain\ \texttt{"␣"}\ Objects\ \texttt{"␣"}\ Init\ \texttt{"␣"}\ Goal\ \texttt{")"}$,

$Problem \rightarrow \texttt{"(problem␣"}\ Name\ \texttt{")"}$,

$Domain \rightarrow \texttt{"(:domain␣"}\ Name\ \texttt{")"}$,

$Objects \rightarrow \texttt{"(:objects␣"}\ ObjDecl\ \{\ \texttt{"␣"}\ ObjDecl\ \}\ \texttt{")"}$,

$ObjDecl \rightarrow Name\ \{\ \texttt{"␣"}\ Name\ \}\ \texttt{"␣-␣"}\ Name$,

$Init \rightarrow \texttt{"(:init␣"}\ PredList\ \texttt{")"}$,

$Goal \rightarrow \texttt{"(:goal␣"}\ PredList\ \texttt{")"}$,

$PredList \rightarrow Pred\ \{\ \texttt{"␣"}\ Pred\ \}$,

$Pred \rightarrow \texttt{"("}\ Name\ \{\ \texttt{"␣"}\ Name\ \}\ \texttt{")"}$,

$Name \rightarrow Letter\ \{\ Char\ \}$,

$Char \rightarrow Letter \mid Digit \mid \texttt{"-"}$,

$Letter \rightarrow \texttt{"a"} \mid \cdots \mid \texttt{"z"} \mid \texttt{"A"} \mid \cdots \mid \texttt{"Z"}$,

$Digit \rightarrow \texttt{"0"} \mid \cdots \mid \texttt{"9"}\ \}$

Aufgabe 10 (0.3 Punkte)

Sei $\mathcal{E}$ die Sprache der Posix Extended Regular Expressions, wobei nur die in der folgenden Tabelle angeführten Sprachelemente zugelassen sind.

regexp	trifft zu auf	*regexp*	trifft zu auf
`\`s	Zeichen s	rr'	r gefolgt von r'
s	s, falls kein Sonderzeichen	r`|`r'	r oder r'
`.`	alle Zeichen	r`*`	≥ 0 Mal r
`[`$s_1 \cdots s_n$`]`	ein Zeichen aus $\{s_1, \ldots, s_n\}$	r`+`	≥ 1 Mal r
`(`r`)`	r	r`?`	≤ 1 Mal r

Weiters ist das Alphabet auf die Zeichen `A`, `9`, `\`, `.`, `[`, `]`, `(`, `)`, `|`, `*`, `+`, `?` und `!` eingeschränkt, es besteht also nur aus den in der Tabelle auftretenden Sondersymbolen sowie aus den drei Symbolen `A`, `9` und `!`.
Beispiel: Das Wort `([A9.]*(!|\?)?)+` liegt in der Sprache $\mathcal{E}$.

Spezifizieren Sie die Sprache der Posix Extended Regular Expressions mit Hilfe einer kontextfreien Grammatik. Verwenden Sie so weit als möglich Ebnf-Notationen, um die Grammatik übersichtlich zu halten und rekursive Regeln zu vermeiden.

Lösung

$\langle V, T, P, \mathit{RegExp} \rangle$, wobei

$$
\begin{aligned}
V = \{&\mathit{RegExp}, \mathit{Quantor}, \mathit{Einzel}, \mathit{Zeichen}, \mathit{SonderZ}, \mathit{NormalZ}\},\\
T = \{&\texttt{"A"}, \texttt{"9"}, \texttt{"\textbackslash"}, \texttt{"."}, \texttt{"["}, \texttt{"]"}, \texttt{"("}, \texttt{")"}, \texttt{"|"}, \texttt{"*"}, \texttt{"+"}, \texttt{"?"}, \texttt{"!"}\},\\
P = \{\ &\mathit{RegExp} \to \mathit{Einzel} \mid \texttt{"("}\mathit{RegExp}\texttt{")"} \mid \mathit{RegExp}\,[\texttt{"|"}]\,\mathit{RegExp} \mid \mathit{RegExp}\,\mathit{Quantor}\ ,\\
&\mathit{Quantor} \to \texttt{"*"} \mid \texttt{"+"} \mid \texttt{"?"}\ ,\\
&\mathit{Einzel} \to \texttt{"\textbackslash"}\mathit{Zeichen} \mid \mathit{NormalZ} \mid \texttt{"."} \mid \texttt{"["}\mathit{Zeichen}\,\{\,\mathit{Zeichen}\,\}\texttt{"]"}\ ,\\
&\mathit{Zeichen} \to \mathit{SonderZ} \mid \mathit{NormalZ}\ ,\\
&\mathit{SonderZ} \to \texttt{"\textbackslash"} \mid \texttt{"."} \mid \texttt{"["} \mid \texttt{"]"} \mid \texttt{"("} \mid \texttt{")"} \mid \texttt{"|"} \mid \mathit{Quantor}\ ,\\
&\mathit{NormalZ} \to \texttt{"A"} \mid \texttt{"9"} \mid \texttt{"!"}\ \ \}
\end{aligned}
$$

Aufgabe 11 (0.4 Punkte)

Zeigen Sie mit Hilfe der algebraischen Gesetze für logische Operatoren und Quantoren, dass die beiden Formeln jeweils äquivalent zueinander sind. Übersetzen Sie jede Formel in natürliche Sprache, wobei die Prädikatensymbole die durch ihren Namen nahegelegten Bedeutungen haben sollen.

(a) $\neg\forall x\,(\mathit{Mann}(x) \supset \exists y\,(\mathit{Frau}(y) \wedge \mathit{Liebt}(x, y)))$ und
$\exists x\,(\mathit{Mann}(x) \wedge \forall y\,(\mathit{Frau}(y) \supset \neg\mathit{Liebt}(x, y)))$

(b) $\forall x\,((\mathit{Frau}(x) \wedge \exists y \mathit{Mann}(y)) \supset \forall z\,\mathit{Liebt}(x, y))$ und
$\forall x\,\neg\mathit{Mann}(x) \vee \forall z\,(\mathit{Frau}(z) \supset \mathit{Liebt}(z, y))$

Lösung

(a)
$$\begin{array}{ll}
\neg\forall x\,(\mathit{Mann}(x) \supset \exists y\,(\mathit{Frau}(y) \wedge \mathit{Liebt}(x,y))) & \neg\forall xF = \exists x\neg F \\
\exists x\,\neg(\mathit{Mann}(x) \supset \exists y\,(\mathit{Frau}(y) \wedge \mathit{Liebt}(x,y))) & F \supset G = \neg F \vee G \\
\exists x\,\neg(\neg\mathit{Mann}(x) \vee \exists y\,(\mathit{Frau}(y) \wedge \mathit{Liebt}(x,y))) & \neg(F \vee G) = \neg F \wedge \neg G \\
\exists x\,(\neg\neg\mathit{Mann}(x) \wedge \neg\exists y\,(\mathit{Frau}(y) \wedge \mathit{Liebt}(x,y))) & \neg\neg F = F \\
\exists x\,(\mathit{Mann}(x) \wedge \neg\exists y\,(\mathit{Frau}(y) \wedge \mathit{Liebt}(x,y))) & \neg\exists xF = \forall x\neg F \\
\exists x\,(\mathit{Mann}(x) \wedge \forall y\,\neg(\mathit{Frau}(y) \wedge \mathit{Liebt}(x,y))) & \neg(F \wedge G) = \neg F \vee \neg G \\
\exists x\,(\mathit{Mann}(x) \wedge \forall y\,(\neg\mathit{Frau}(y) \vee \neg\mathit{Liebt}(x,y))) & \neg F \vee G = F \supset G \\
\exists x\,(\mathit{Mann}(x) \wedge \forall y\,(\mathit{Frau}(y) \supset \neg\mathit{Liebt}(x,y))) &
\end{array}$$

Die erste Formel lässt sich lesen als „Nicht alle Männer lieben eine Frau“, die letzte als „Es gibt Männer, die alle Frauen nicht lieben“. Die Zwischenformel in der fünften Zeile entspricht der Formulierung „Manche Männer lieben keine Frau“.

(b)
$$\begin{array}{ll}
\forall x\,((\mathit{Frau}(x) \wedge \exists y\mathit{Mann}(y)) \supset \forall z\,\mathit{Liebt}(x,y)) & \forall zF = F \text{ wenn } z \text{ nicht in } F \text{ vorkommt} \\
\forall x\,((\mathit{Frau}(x) \wedge \exists y\mathit{Mann}(y)) \supset \mathit{Liebt}(x,y)) & F \supset G = \neg F \vee G \\
\forall x\,(\neg(\mathit{Frau}(x) \wedge \exists y\mathit{Mann}(y)) \vee \mathit{Liebt}(x,y)) & \neg(F \wedge G) = \neg F \vee \neg G \\
\forall x\,(\neg\mathit{Frau}(x) \vee \neg\exists y\mathit{Mann}(y) \vee \mathit{Liebt}(x,y)) & \neg\exists xF = \forall x\neg F \\
\forall x\,(\neg\mathit{Frau}(x) \vee \forall y\neg\mathit{Mann}(y) \vee \mathit{Liebt}(x,y)) & \forall x(F \vee G) = F \vee \forall xG \text{ wenn } x \text{ nicht in } F \text{ vorkommt} \\
\forall y\,\neg\mathit{Mann}(y) \vee \forall x\,(\neg\mathit{Frau}(x) \vee \mathit{Liebt}(x,y)) & \neg F \vee G = F \supset G \\
\forall y\,\neg\mathit{Mann}(y) \vee \forall x\,(\mathit{Frau}(x) \supset \mathit{Liebt}(x,y)) & \forall yF[y] = \forall xF[x] \text{ wenn } x \text{ nicht in } F \text{ vorkommt} \\
\forall x\,\neg\mathit{Mann}(x) \vee \forall x\,(\mathit{Frau}(x) \supset \mathit{Liebt}(x,y)) & \forall xF[x] = \forall zF[z] \text{ wenn } z \text{ nicht in } F \text{ vorkommt} \\
\forall x\,\neg\mathit{Mann}(x) \vee \forall z\,(\mathit{Frau}(z) \supset \mathit{Liebt}(z,y)) &
\end{array}$$

Die erste Formel entspricht der Aussage:

> Für alle x gilt:
> Falls x eine Frau ist und falls es irgendeinen Mann gibt,
> dann gilt für alles, dass x ein Objekt y liebt.

Auffällig an dieser Formel ist, dass es eine freie Variable y für ein Element des Universiums gibt, das von x geliebt wird. Diese Variable hat nichts mit der gleichnamigen Variable in der Teilformel $\exists y\mathit{Mann}(y)$ zu tun. Weiters gibt es noch die quantifizierte Variable z, die aber nirgends vorkommt, wodurch der entsprechende Quantor bedeutungslos ist.

Die letzte Formel ist die „aufgeräumte“ Version der ersten und kann gelesen werden als:

> Alles ist kein Mann (bzw. niemand ist ein Mann),
> oder alle Frauen lieben das durch y bezeichnete Objekt.

Aufgabe 12 (0.5 Punkte)

Seien *Person*/1, *Nass*/1, *Hobby*/1 und *Betreibt*/2 Prädikatensymbole sowie *surfen* und *tauchen* Konstantensymbole mit folgender Bedeutung:

$Person(x)$	... x ist eine Person	$Betreibt(x, y)$	... x betreibt y
$Nass(x)$	... x ist nass	*surfen*	... Surfen
$Hobby(x)$	... x ist ein Hobby	*tauchen*	... Tauchen

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

(a) Alle Personen, die Surfen oder Tauchen betreiben, sind nass.

(b) Es gibt Personen, die alle nassen Hobbies betreiben.

Sei weiters folgende Interpretation I gegeben:

$$\begin{aligned}
\mathcal{U} &= \{\text{Tom, Anna, Lisa, Karin, Schwimmen, Surfen, Segeln, Eislaufen,} \\
&\qquad \text{Tauchen, Radfahren, Lesen, Laufen}\} \\
I(Person) &= \{\text{Tom, Lisa, Karin}\} \\
I(Nass) &= \{\text{Schwimmen, Surfen, Segeln, Eislaufen}\} \\
I(Hobby) &= \{\text{Schwimmen, Surfen, Tauchen, Radfahren, Lesen, Laufen}\} \\
I(Betreibt) &= \{(\text{Tom, Eislaufen}), (\text{Tom, Radfahren}), (\text{Tom, Segeln}), \\
&\qquad (\text{Anna, Radfahren}), (\text{Anna, Tauchen}), (\text{Anna, Eislaufen}), \\
&\qquad (\text{Lisa, Surfen}), (\text{Lisa, Schwimmen}), (\text{Lisa, Segeln}), \\
&\qquad (\text{Karin, Radfahren}), (\text{Karin, Eislaufen})\} \\
I(surfen) &= \text{Surfen} \\
I(radfahren) &= \text{Radfahren}
\end{aligned}$$

Übersetzen Sie die nachfolgenden Formeln in natürliche Sprache. Geben Sie an, ob die Formeln in der Interpretation I wahr oder falsch sind. Begründen Sie Ihre Antwort; es ist keine formale Auswertung erforderlich.

(c) $\exists x\,(Betreibt(x, surfen) \wedge \neg Betreibt(x, radfahren))$

(d) $\forall x\,(Nass(x) \supset (Person(x) \wedge \exists y\, Betreibt(x, y)))$

(e) $\exists x\,(Person(x) \wedge \exists y\,(Hobby(y) \wedge Betreibt(x, y)))$

Lösung

(a) $\forall x\,((Person(x) \wedge (Betreibt(x, surfen) \vee Betreibt(x, tauchen))) \supset Nass(x))$

(b) $\exists x\,(Person(x) \wedge \forall y\,((Hobby(y) \wedge Nass(y)) \supset Betreibt(x, y)))$

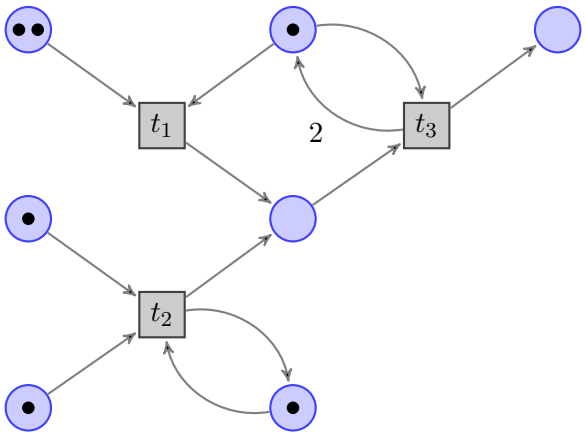
(c) Übersetzung: Es gibt etwas, das surft aber nicht Rad fährt.
Diese Aussage ist richtig, da $(\text{Lisa, Surfen}) \in I(Betreibt)$ und $(\text{Lisa, Radfahren}) \notin I(Betreibt)$

(d) Übersetzung: Alles Nasse ist eine Person und betreibt irgendetwas.
Diese Aussage ist falsch, da $\text{Schwimmen} \in I(Nass)$ aber $\text{Schwimmen} \notin I(Person)$.

(e) Übersetzung: Es gibt eine Person, die ein Hobby betreibt.
Diese Aussage ist wahr, da Tom $\in I(Person)$, Radfahren $\in I(Hobby)$ und (Tom, Radfahren) $\in I(Betreibt)$.

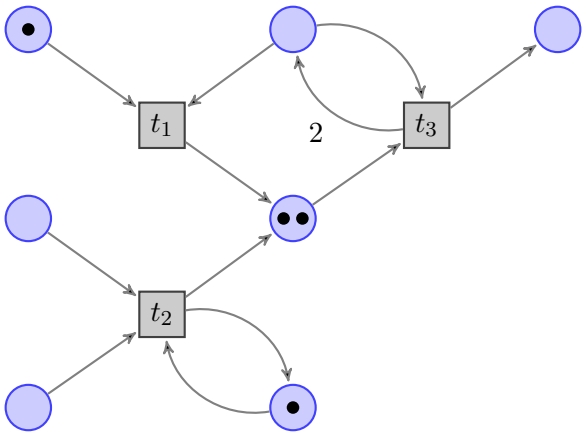
Aufgabe 13 (0.4 Punkte)

Geben Sie für das folgende Petri-Netz mit Anfangsmarkierung alle möglichen Reihenfolgen an, in denen die Transitionen feuern können, sowie die Endmarkierungen, die dadurch erreicht werden. (Endmarkierung bedeutet, dass keine Transition aktiviert ist.)

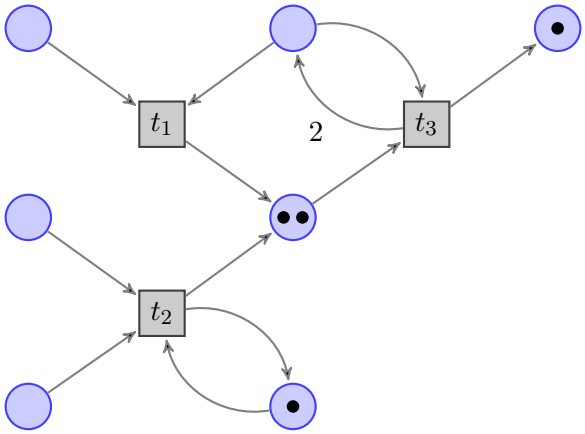


Lösung

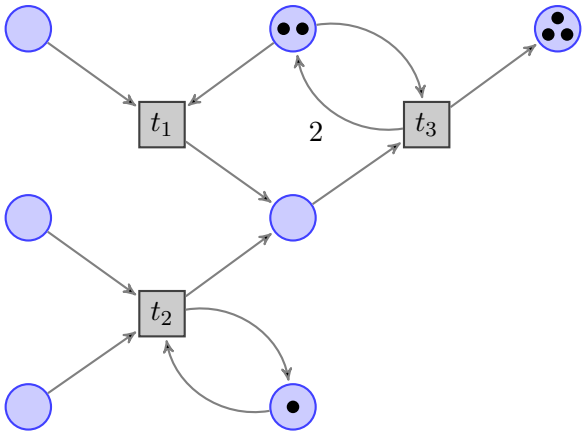
Die Transitionsfolgen t_1–t_2 und t_2–t_1 liefern beide die folgende Endmarkierung.



Feuern die Transitionen in der Reihenfolge t_2–t_3–t_1–t_1, ergibt sich folgende Markierung.



Die letzte Möglichkeit ist die Reihenfolge t_2–t_3–t_1–t_3–t_1–t_3, die mit folgender Markierung endet:



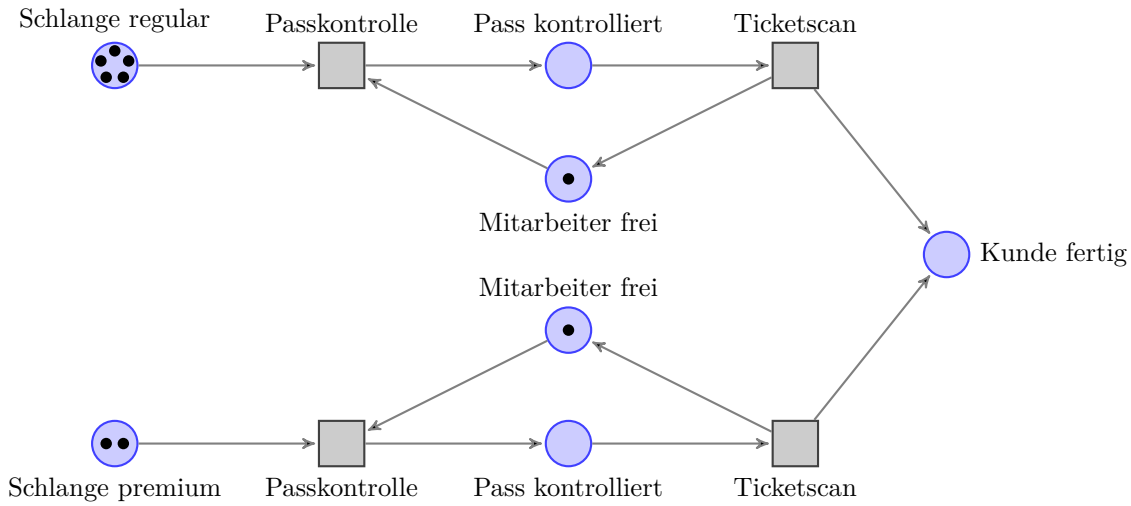
Aufgabe 14 (0.4 Punkte)

Beim Boarding eines Flugzeugs werden zwei Schlangen gebildet, „regular“ und „premium“, wobei die Premiumschlange für Vielflieger gedacht ist. Jede der beiden Schlangen wird von einem Mitarbeiter der Fluglinie betreut. Der Prozess ist bei beiden Schlangen ident: Zuerst wird der Reisepass kontrolliert und dann der Barcode auf dem Flugticket gescannt.

(a) Modellieren Sie dieses System mit Hilfe eines Petri-Netzes. Geben Sie eine geeignete Anfangsmarkierung für fünf reguläre und zwei Vielflieger an.

(b) Da es meist mehr reguläre als Premiumgäste gibt, ist der Premiumbetreuer weniger ausgelastet als der andere. Erweitern Sie Ihr Modell dahingegend, dass der Premiumbetreuer, wenn er nicht gerade einen Passagier abfertigt, Fluggäste von der regulären Schlange in die Premiumschlange umreihen kann.

Lösung



Die Token, die die Kunden repräsentieren, befinden sich zunächst in der „Schlange regular“ oder der „Schlange premium“. Ist der jeweilige Mitarbeiter verfügbar, so kann die Passkontrolle und anschließend der Ticketscan durchgeführt werden. Der Mitarbeiter ist erst wieder für den nächsten Kunden bereit, sobald für den aktuellen Kunden beide Schritte durchgeführt wurden.
Soll der Mitarbeiter, der die Premiummitglieder betreut, zusätzlich einzelne Kunden von der „regular“ in die „premium“ Schlange schicken können, so muss das Netz wie folgt erweitert werden.