2. Übungsblatt (mit Lösungen)

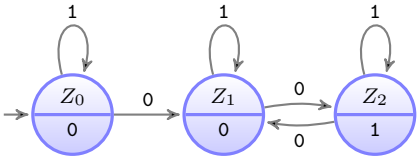
3.0 VU Formale Modellierung

Gernot Salzer, Marion Scholz

30. April 2017

Aufgabe 1 (3 Punkte)

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



(a) Geben Sie die Ausgaben zu folgenden Eingaben an: `10101`, `11110`, `00111`.

(b) Berechnen Sie schrittweise $\delta^*(Z_0, \texttt{01010})$ und $\gamma^*(Z_0, \texttt{01010})$.

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

Lösung

(a)

w:	`10101`	`11110`	`00111`
$[\mathcal{A}](w)$:	`000011`	`000000`	`001111`

(b)
$$\begin{aligned}
\delta^*(Z_0, \texttt{01010}) &= \delta^*(\delta(Z_0, \texttt{0}), \texttt{1010}) &&= \delta^*(Z_1, \texttt{1010}) \\
&= \delta^*(\delta(Z_1, \texttt{1}), \texttt{010}) &&= \delta^*(Z_1, \texttt{010}) \\
&= \delta^*(\delta(Z_1, \texttt{0}), \texttt{10}) &&= \delta^*(Z_2, \texttt{10}) \\
&= \delta^*(\delta(Z_2, \texttt{1}), \texttt{0}) &&= \delta^*(Z_2, \texttt{0}) \\
&= \delta^*(\delta(Z_2, \texttt{0}), \varepsilon) &&= \delta^*(Z_1, \varepsilon) \\
&= Z_1
\end{aligned}$$

$$
\begin{aligned}
\gamma^*(Z_0, \texttt{01010}) &= \gamma(Z_0) \cdot \gamma^*(\delta(Z_0, \texttt{0}), \texttt{1010}) &&= \texttt{0} \cdot \gamma^*(Z_1, \texttt{1010}) \\
&= \texttt{0} \cdot \gamma(Z_1) \cdot \gamma^*(\delta(Z_1, \texttt{1}), \texttt{010}) &&= \texttt{00} \cdot \gamma^*(Z_1, \texttt{010}) \\
&= \texttt{00} \cdot \gamma(Z_1) \cdot \gamma^*(\delta(Z_1, \texttt{0}), \texttt{10}) &&= \texttt{000} \cdot \gamma^*(Z_2, \texttt{10}) \\
&= \texttt{000} \cdot \gamma(Z_2) \cdot \gamma^*(\delta(Z_2, \texttt{1}), \texttt{0}) &&= \texttt{0001} \cdot \gamma^*(Z_2, \texttt{0}) \\
&= \texttt{0001} \cdot \gamma(Z_2) \cdot \gamma^*(\delta(Z_2, \texttt{0}), \varepsilon) &&= \texttt{00011} \cdot \gamma^*(Z_1, \varepsilon) \\
&= \texttt{00011} \cdot \gamma(Z_1) &&= \texttt{000110}
\end{aligned}
$$

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

Aufgabe 2 (3 Punkte)

Sei L die Sprache

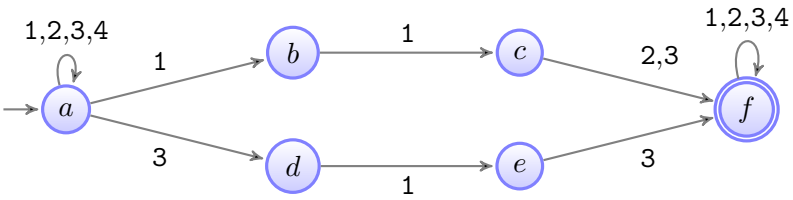
$$\{\, w \in \{\texttt{1}, \texttt{2}, \texttt{3}, \texttt{4}\}^* \mid w \text{ beinhaltet die Ziffernfolge } \texttt{112}, \texttt{113} \text{ oder } \texttt{313} \,\} \ .$$

Geben Sie einen *deterministischen* Automaten für L an. Gehen 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.

Lösung

Wir konstruieren zunächst auf möglichst direktem Weg einen beliebigen Automaten für die gesuchte Sprache. Dieser ist im Zustand a indeterministisch: Für das Symbol `1` und für das Symbol `3` gibt es jeweils zwei mögliche Folgezustände. Zusätzlich zur graphischen Darstellung geben wir die Übergangsfunktion auch als Tabelle an, da diese bei der Determinisierung hilft.



δ^*	1	2	3	4
a	$\{a, b\}$	$\{a\}$	$\{a, d\}$	$\{a\}$
b	$\{c\}$	$\{\}$	$\{\}$	$\{\}$
c	$\{\}$	$\{f\}$	$\{f\}$	$\{\}$
d	$\{e\}$	$\{\}$	$\{\}$	$\{\}$
e	$\{\}$	$\{\}$	$\{f\}$	$\{\}$
f	$\{f\}$	$\{f\}$	$\{f\}$	$\{f\}$

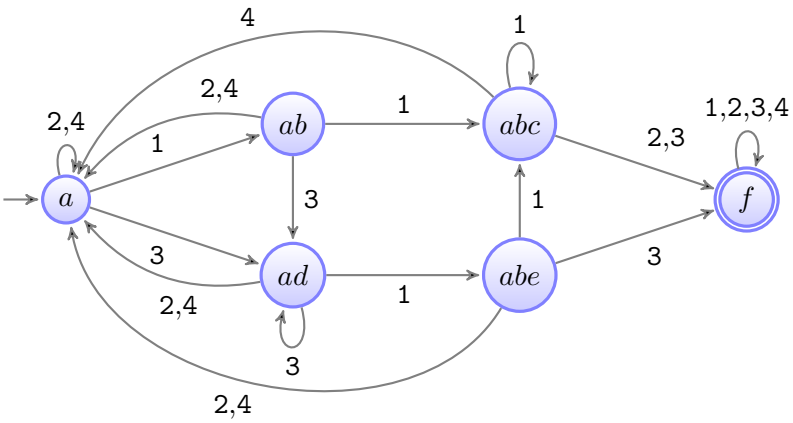
Einen deterministischen Automaten erhalten wir, indem wir den indeterminischen Automaten simulieren. Ein Zustand des deterministischen Automaten repräsentiert dabei jene Zustände des indeterministischen, in denen sich dieser zu diesem Zeitpunkt befinden kann. Der Startzustand wird mit $\{a\}$ bezeichnet, 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.

$\hat{\delta}$	1	2	3	4
$\{a\}$	$\{a, b\}$	$\{a\}$	$\{a, d\}$	$\{a\}$
$\{a, b\}$	$\{a, b, c\}$	$\{a\}$	$\{a, d\}$	$\{a\}$
$\{a, d\}$	$\{a, b, e\}$	$\{a\}$	$\{a, d\}$	$\{a\}$
$\{a, b, c\}$	$\{a, b, c\}$	$\{a, f\}$	$\{a, d, f\}$	$\{a\}$
$\{a, b, e\}$	$\{a, b, c\}$	$\{a\}$	$\{a, d, f\}$	$\{a\}$
$\{a, f\}$	$\{a, b, f\}$	$\{a, f\}$	$\{a, d, f\}$	$\{a, f\}$
$\{a, d, f\}$	$\{a, b, e, f\}$	$\{a, f\}$	$\{a, d, f\}$	$\{a, f\}$
$\{a, b, f\}$	$\{a, b, c, f\}$	$\{a, f\}$	$\{a, d, f\}$	$\{a, f\}$
$\{a, b, e, f\}$	$\{a, b, c, f\}$	$\{a, f\}$	$\{a, d, f\}$	$\{a, f\}$
$\{a, b, c, f\}$	$\{a, b, c, f\}$	$\{a, f\}$	$\{a, d, f\}$	$\{a, f\}$

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 f enthält. Dieser wird somit durch das Tupel $\langle \widehat{Q}, \Sigma, \hat{\delta}, \{a\}, \widehat{F} \rangle$ beschrieben, wobei

$$\Sigma = \{1, 2, 3, 4, 5\}$$
$$\widehat{F} = \{\{a, f\}, \{a, b, f\}, \{a, d, f\}, \{a, b, c, f\}, \{a, b, e, f\}\}$$
$$\widehat{Q} = \{\{a\}, \{a, b\}, \{a, d\}, \{a, b, c\}, \{a, b, e\}\} \cup \widehat{F}$$

Auch ohne Anwendung des allgemeinen Minimierungsverfahrens (das in der Vorlesung nicht besprochen wurde) lässt sich erkennen, dass die Endzustände zu einem einzigen zusammengefasst werden können: Wurde einer der Endzustände erreicht, führt jedes weitere Eingabesymbol wieder zu einem Endzustand, jedes beliebige Wortende wird somit akzeptiert. Das lässt sich auch mit einem einzigen Endzustand erreichen. Hier eine graphische Darstellung des bereits optimierten Automaten, wobei die zusammengefassten Endzustände



Aufgabe 3 (3 Punkte)

Vereinfachen Sie die folgenden Ausdrücke.

(a) $(\{\mathtt{a}, \mathtt{ab}\} \cup \{\}) \cdot \{\} \cup \{\}$

(b) $(\{\mathtt{a}\} \cup \{\varepsilon\}) \cdot \{\mathtt{a}\}^*$

(c) $\{\mathtt{a}\} \cdot \{\mathtt{bca}\}^* \cdot \{\mathtt{b}\} \cdot \{\mathtt{cab}\}^* \cdot \{\mathtt{c}\} \cdot \{\mathtt{abc}\}^*$

Lösung

(a) $(\{\mathtt{a}, \mathtt{ab}\} \cup \{\}) \cdot \{\} \cup \{\} = \{\} \cup \{\} = \{\}$

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

(c)
$$\begin{aligned}
&\{\mathtt{a}\} \cdot \{\mathtt{bca}\}^* \cdot \{\mathtt{b}\} \cdot \{\mathtt{cab}\}^* \cdot \{\mathtt{c}\} \cdot \{\mathtt{abc}\}^* \\
&= \{\mathtt{abc}\}^* \cdot \{\mathtt{ab}\} \cdot \{\mathtt{cab}\}^* \cdot \{\mathtt{c}\} \cdot \{\mathtt{abc}\}^* \\
&= \{\mathtt{abc}\}^* \cdot \{\mathtt{abc}\}^* \cdot \{\mathtt{abc}\} \cdot \{\mathtt{abc}\}^* \\
&= \{\mathtt{abc}\}^* \cdot \{\mathtt{abc}\} \cdot \{\mathtt{abc}\}^* \\
&= \{\mathtt{abc}\}^+ \cdot \{\mathtt{abc}\}^* \\
&= \{\mathtt{abc}\}^+
\end{aligned}$$

Aufgabe 4 (2 Punkte)

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

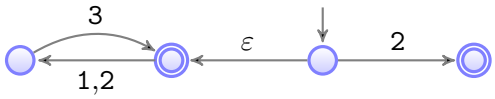
(a) $((1 + 2)3)^* + 2$

(b) $1^*(2 + 3)2^+$

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.

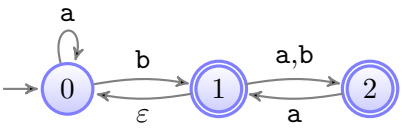
(a)



(b)

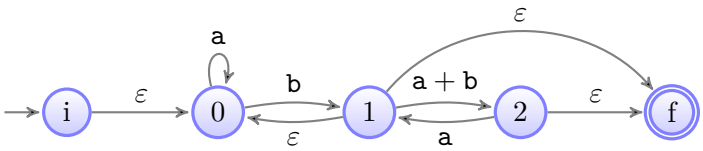
Aufgabe 5 (4 Punkte)

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!



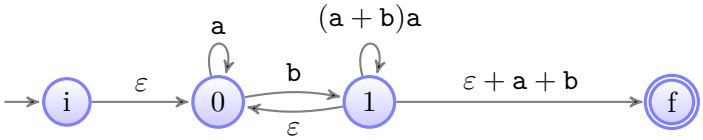
Lösung

Neuer Anfangs- und Endzustand:

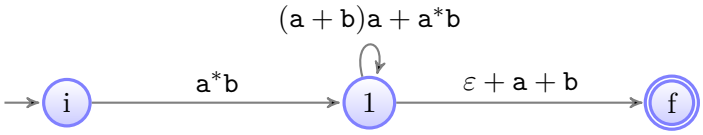


Wir eliminieren die Zustände in der Reihenfolge 2, 0 und 1; die anderen Reihenfolgen sind ebenfalls möglich.

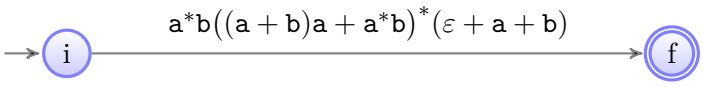
Elimination von Zustand 2:



Elimination von Zustand 0:



Elimination von Zustand 1:



Die Sprache des ursprünglichen Automaten wird also durch den Ausdruck

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

beschrieben.

Aufgabe 6 (3 Punkte)

Die Automaten in den Übungsblättern werden mit Hilfe des LaTeX-Pakets TikZ[1] gezeichnet. Um einen Zustand als Knoten eines Graphen darzustellen, benutzt man (vereinfacht) einen Befehl mit folgendem Aufbau: Der Befehl beginnt mit `\node`, anschließend folgen in eckigen Klammern Optionen (siehe unten); die eckigen Klammern samt Optionen können auch fehlen. Danach folgt in runden Klammern eine interne Bezeichnung für den Knoten. Diese kann beliebig gewählt werden; für diese Aufgabe nehmen wir an, dass die Bezeichnung nur aus ein oder zwei Buchstaben und/oder Ziffern besteht. Nach der Bezeichnung in runden Klammern folgt die Beschriftung des Knotens in geschwungenen Klammern, so wie er in der fertigen Abbildung benannt werden soll. Ein Strichpunkt beendet den Befehl.
Die Optionen (sofern welche vorhanden sind) werden durch Beistriche getrennt. Folgende Optionen können in beliebiger Reihenfolge angegeben werden: `state`, `initial` (für den Startzustand), `accepting` (für einen Endzustand), sowie Positionsangeben. Positionsangaben beginnen mit einem der Wörter `below`, `above`, `left` oder `right`, dem ein Gleichheitszeichen (`=`), das Schlüsselwort `of`, ein Leerzeichen sowie eine interne Knotenbezeichnung folgt.
Beispiele für derartige Befehle (␣ steht für ein Leerzeichen):

```
\node[state,initial,left=of␣0](i){i};
\node[state,initial,accepting](12){1};
\node[state,below=of␣i](2){knotenname};
```

Beschreiben Sie den Aufbau solcher `\node`-Befehle mit den folgenden Methoden. Treffen Sie sinnvolle Annahmen, wenn Ihnen Informationen fehlen.

(a) Geben Sie einen regulären Ausdruck in algebraischer Notation an.

[1] Abkürzung für „TikZ ist kein Zeichenprogramm“; Beispiel für ein rekursives Akronym.

(b) Geben Sie einen regulären Ausdruck in POSIX-Notation an, der alle Zeilen beschreibt, die *ausschließlich* einen derartigen Befehl enthalten.

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

Lösung

(a) Wir schreiben die Symbole des Alphabets unter Anführungszeichen, um sie von algebraischen Symbolen (Metanotation) zu unterscheiden. Der Einfachheit halber berücksichtigen wir nur Kleinbuchstaben.

`"\node"` $(\mathit{Optionen} + \varepsilon)$ `"("`*Name*`"){"`*Inhalt*`"};"`

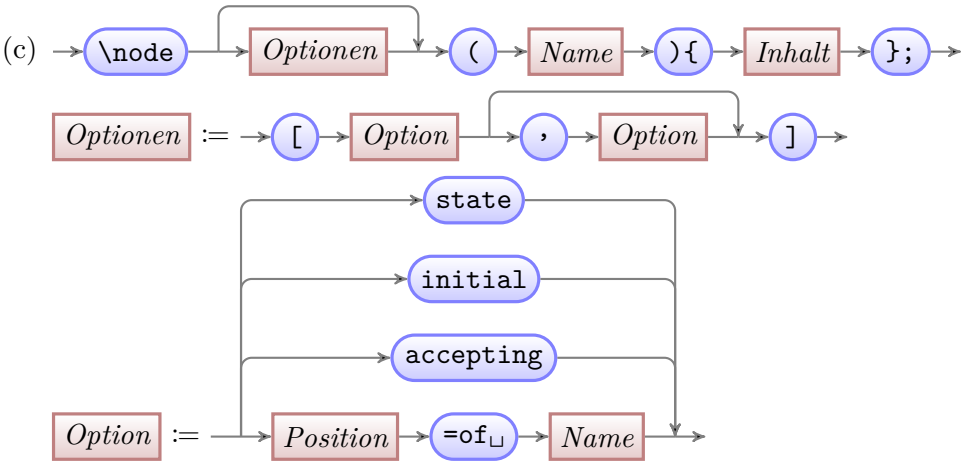
mit den Abkürzungen

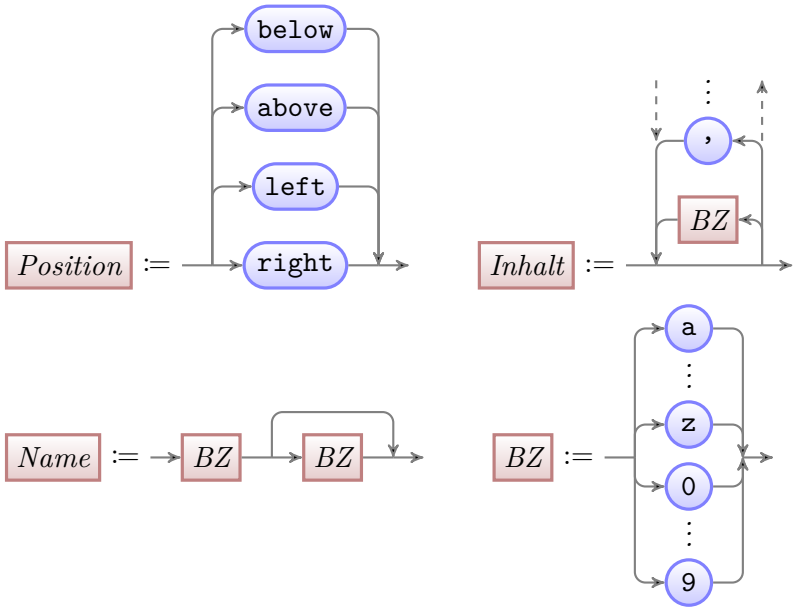
$$
\begin{aligned}
\mathit{Optionen} &:= \texttt{"["}\ \mathit{Option}\,(\ \texttt{","}\ \mathit{Option}\,)^*\ \texttt{"]"} \\
\mathit{Option} &:= \texttt{"state"} + \texttt{"initial"} + \texttt{"accepting"} + \mathit{Position}\ \texttt{"=of␣"}\mathit{Name} \\
\mathit{Position} &:= (\ \texttt{"below"} + \texttt{"above"} + \texttt{"left"} + \texttt{"right"}\ ) \\
\mathit{Name} &:= \mathit{BZ}(\mathit{BZ} + \varepsilon) \\
\mathit{Inhalt} &:= (\mathit{BZ} + \texttt{","} + \cdots)^* \\
\mathit{BZ} &:= (\ \texttt{"a"} + \cdots + \texttt{"z"} + \texttt{"0"} + \cdots + \texttt{"9"}\ )
\end{aligned}
$$

(b) `^\\node(\[`*Option*`(,`*Option*`)*\])?\([a-z0-9]{1,2}\)\{.*\};$`
wobei *Option* eine Abkürzung für den folgenden Ausdruck ist:

`(state|initial|accepting|(below|above|left|right)=of␣[a-z0-9]{1,2})`

(c)





Aufgabe 7 (6 Punkte)

„kikerikiiiiiiiii! kukerikuuuuuu!“ So oder so ähnlich klingt es, wenn Kokolorix[2] ein Liedchen trällert. Die Grammatik $G = \langle V, T, P, A \rangle$ erzeugt solche Hahnenschreie, wobei

$$
\begin{aligned}
V &= \{A, B, C, D\} \\
T &= \{\texttt{e}, \texttt{i}, \texttt{u}, \texttt{k}, \texttt{r}\} \\
P &= \{\, A \to \texttt{"ki"}\; B\; \texttt{"ku"} \;\;, \\
&\qquad B \to C\; B \mid \varepsilon \\
&\qquad C \to \texttt{"ke"}\; D \mid A \;\;, \\
&\qquad D \to \texttt{"ri"}\; D \mid \varepsilon \;\}
\end{aligned}
$$

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.

(a) `kikeriki`

(b) `kikeriku`

(c) `kikekekirikuku`

Welche der folgenden Aussagen über die Sprache der Hahnenschreie sind korrekt? Begründen Sie Ihre Antwort!

(d) Die Anzahl der `ki` und die Anzahl der `ku` in einem Wort sind gleich.

[2] Der Hahn aus einem wohlbekannten gallischen Dorf, das seit jeher den Römern Widerstand leistet.

(e) Man kann beliebig lange Wörter bilden, in denen keine zwei gleichen Silben aufeinander folgen.

(f) Die Sprache $\mathcal{L}(G)$ kann nicht durch einen endlichen Automaten beschrieben werden.

Lösung

(a) Das Wort liegt nicht in der Sprache $\mathcal{L}(G)$. Betrachtet man die Produktion für die Startvariable A, sieht man, dass jedes Wort der Sprache auf `ku` enden muss.

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

$$\begin{aligned} A &\Rightarrow \texttt{"ki"}\, B\, \texttt{"ku"} \Rightarrow \texttt{"ki"}\, C\, B\, \texttt{"ku"} \Rightarrow \texttt{"kike"}\, D\, B\, \texttt{"ku"} \\ &\Rightarrow \texttt{"kikeri"}\, D\, B\, \texttt{"ku"} \Rightarrow \texttt{"kikeri"}\, \varepsilon\, B\, \texttt{"ku"} \Rightarrow \texttt{"kikeri"}\, \varepsilon\, \texttt{"ku"} \\ &\Rightarrow \texttt{"kikeriku"} \end{aligned}$$

(c) Um zu zeigen, dass ein Wort nicht in der Sprache der Grammatik liegt, müssen wir zeigen, dass dieses eine Eigenschaft nicht besitzt, die aber jedes Wort in $\mathcal{L}(G)$ hat. Wir verwenden folgende Eigenschaft: „Auf `ki` kann nur `ke`, `ki` oder `ku` folgen.“ Diese Eigenschaft trifft auf das Wort `kikekekirikuku` nicht zu, da auf `ki` die Silbe `ri` folgt.

Es bleibt zu zeigen, dass tatsächlich jedes Wort in $\mathcal{L}(G)$ diese Eigenschaft besitzt. Die Silbe `ki` kann nur durch Anwenden der ersten Produktion aus A entstehen. Wenn wir nun alle Möglichkeiten betrachten, wie fortgesetzt werden kann, sehen wir, dass nur `ke`, `ki` oder `ku` folgen kann.

$$A \Rightarrow \texttt{"ki"}\, B\, \texttt{"ku"} \Rightarrow \texttt{"kiku"}$$
$$A \Rightarrow \texttt{"ki"}\, B\, \texttt{"ku"} \Rightarrow \texttt{"ki"}\, C\, B\, \texttt{"ku"} \Rightarrow \texttt{"kike"}\, D\, B\, \texttt{"ku"}$$
$$A \Rightarrow \texttt{"ki"}\, B\, \texttt{"ku"} \Rightarrow \texttt{"ki"}\, C\, B\, \texttt{"ku"} \Rightarrow \texttt{"ki"}\, A\, B\, \texttt{"ku"} \Rightarrow \texttt{"kiki"}\, B\, \texttt{"ku"}\, B\, \texttt{"ku"}$$

(d) Ja, sowohl `ki` als auch `ku` können nur durch die Produktion $A \Rightarrow \texttt{"ki"}\, B\, \texttt{"ku"}$ in ein Wort der Sprache gelangen und werden dabei immer paarweise erzeugt. Die Anzahl der beiden Silben ist daher gleich.

(e) Ja. Die Grammatik erlaubt folgende Ableitung aus A:

$$\begin{aligned} A &\Rightarrow \texttt{"ki"}\, B\, \texttt{"ku"} && \Rightarrow \texttt{"kikeri"}\, A\, B\, \texttt{"ku"} \\ &\Rightarrow \texttt{"ki"}\, C\, B\, \texttt{"ku"} && \Rightarrow \texttt{"kikeri"}\, A\, C\, B\, \texttt{"ku"} \\ &\Rightarrow \texttt{"kike"}\, D\, B\, \texttt{"ku"} && \Rightarrow \texttt{"kikeri"}\, A\, \texttt{"ke"}\, D\, B\, \texttt{"ku"} \\ &\Rightarrow \texttt{"kikeri"}\, D\, B\, \texttt{"ku"} && \Rightarrow \texttt{"kikeri"}\, A\, \texttt{"ke"}\, \varepsilon\, B\, \texttt{"ku"} \\ &\Rightarrow \texttt{"kikeri"}\, \varepsilon\, B\, \texttt{"ku"} && \Rightarrow \texttt{"kikeri"}\, A\, \texttt{"ke"}\, \varepsilon\, \texttt{"ku"} \\ &\Rightarrow \texttt{"kikeri"}\, C\, B\, \texttt{"ku"} && \Rightarrow \texttt{"kikeri"}\, A\, \texttt{"keku"} \end{aligned}$$

Weiters kann man aus A auch `kiku` ableiten:

$$A \Rightarrow \texttt{"ki"}\, B\, \texttt{"ku"} \Rightarrow \texttt{"ki"}\, \varepsilon\, \texttt{"ku"} = \texttt{"kiku"}$$

Zusammengesetzt ist daher folgende Ableitung möglich:

$$\begin{aligned} A &\overset{*}{\Rightarrow} \texttt{"kikeri"}\, A\, \texttt{"keku"} \overset{*}{\Rightarrow} (\texttt{"kikeri"})^2\, A\, (\texttt{"keku"})^2 \overset{*}{\Rightarrow} \cdots \\ &\overset{*}{\Rightarrow} (\texttt{"kikeri"})^n\, A\, (\texttt{"keku"})^n \overset{*}{\Rightarrow} (\texttt{"kikeri"})^n\, \texttt{"kiku"}\, (\texttt{"keku"})^n \end{aligned}$$

Somit lassen sich beliebig lange Wörter aus der Startvariablen A ableiten, in denen keine zwei gleichen Silben aufeinander folgen.

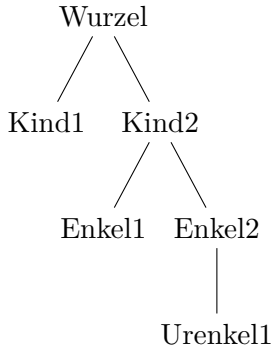
(f) Nein, die Sprache $\mathcal{L}(G)$ kann nicht durch einen endlichen Automaten beschrieben werden. Die Anzahl der `ki`'s und `ku`'s muss gleich groß sein, somit müsste sich der endliche Automat die Zahl der `ki`'s und `ku`'s in seinen Zuständen merken. Da diese Zahl nicht begrenzt werden kann, reicht eine endliche Zahl von Zuständen nicht aus.

(Diese Aufgabe wurde von Unterrichtsmaterialien des Gymnasiums Odenthal inspiriert.[3])

Aufgabe 8 (3 Punkte)

Baumdiagramme können in LaTeX mit dem Paket TikZ erstellt werden. Beispielsweise kann ein einfacher Baum mit vier Ebenen folgendermaßen beschrieben werden.

```
\begin{tikzpicture}
 \node{Wurzel}
   child{node{Kind1}}
   child{node{Kind2}
     child{node{Enkel1}}
     child{node{Enkel2}
       child{node{Urenkel1}}
          }
      };
\end{tikzpicture}
```



Die Beschreibungen sind zwischen `\begin{tikzpicture}` und `\end{tikzpicture}` eingeschlossen. Jeder Knoten beginnt mit dem Schlüsselwort `node`. Danach folgt die Bezeichnung, die aus Buchstaben und Ziffern besteht, in geschwungenen Klammern. Im Anschluss an einen Knoten werden seine Unterbäume aufgelistet. Jeder Unterbaum beginnt mit dem Schlüsselwort `child`, dem in geschwungenen Klammern ein weiterer Knoten mit dessen Unterbäumen folgt. Dem Schlüsselwort `node` des Wurzelknotens geht ein verkehrter Schrägstrich (\) voraus; dem letzten Unterbaum folgt ein Strichpunkt (`;`).

[3] `http://projekte.gymnasium-odenthal.de/informatik/dateien/Informatik/Jahrgangsstufe%20Q/003%20Unterrichtsreihen%20Java/06%20Vertiefung%20Automaten-Sprachen/02%20kontextfreie%20Sprachen/Docs/03%20Arbeitsblatt%20Kontextfreie%20Grammatik%20(Kraehen).pdf`

Sei $\mathcal{L}$ die Menge derartiger einfacher TikZ-Baumbeschreibungen. Spezifizieren Sie die Sprache $\mathcal{L}$ mit Hilfe einer kontextfreien Grammatik. Verwenden Sie EBNF-Notationen, um die Grammatik übersichtlich zu strukturieren.

Handelt es sich bei $\mathcal{L}$ um eine reguläre Sprache, d.h., lässt sich diese Sprache auch durch einen (komplizierten) regulären Ausdruck spezifizieren? Begründen Sie Ihre Antwort.

Lösung

Die Sprache $\mathcal{L}$ wird durch die Grammatik $\langle N, T, P, \mathit{Tikz}\rangle$ beschrieben, wobei

$$
\begin{aligned}
N &= \{\, \mathit{Tikz}, \mathit{Baum}, \mathit{Kinder}, \mathit{Kind}, \mathit{Bezeichnung}, \mathit{Buchstabe}, \mathit{Ziffer}\},\\
T &= \{\cdots \textit{alle Zeichen in den Produktionen zwischen Anführungszeichen} \cdots\},\\
P &= \{\quad \mathit{Tikz} \to \texttt{"\textbackslash begin\{tikzpicture\}"}\ \texttt{"\textbackslash"}\ \mathit{Baum}\ \texttt{";"}\ \texttt{"\textbackslash end\{tikzpicture\}"}\ ,\\
&\qquad \mathit{Baum} \to \texttt{"node"}\ \texttt{"\{"}\ \mathit{Bezeichnung}\ \texttt{"\}"}\ \mathit{Kinder}\ ,\\
&\qquad \mathit{Kinder} \to \{\, \mathit{Kind}\,\}\ ,\\
&\qquad \mathit{Kind} \to \texttt{"child"}\ \texttt{"\{"}\ \mathit{Baum}\ \texttt{"\}"}\ ,\\
&\qquad \mathit{Bezeichnung} \to \{\, \mathit{Buchstabe} \mid \mathit{Ziffer}\,\}\ ,\\
&\qquad \mathit{Buchstabe} \to \texttt{"a"} \mid \cdots \mid \texttt{"z"} \mid \texttt{"A"} \mid \cdots \mid \texttt{"Z"}\ ,\\
&\qquad \mathit{Ziffer} \to \texttt{"0"} \mid \cdots \mid \texttt{"9"}\ \}
\end{aligned}
$$

Die Sprache $\mathcal{L}$ ist *nicht* regulär, sie lässt sich daher nicht durch einen regulären Ausdruck beschreiben. Grund dafür sind die geschachtelten Unterbäume, die jeweils in geschwungenen Klammern eingeschlossen sind. Die Kontrolle über die richtige Anzahl an schließenden Klammern ist in regulären Sprachen nicht möglich.

Aufgabe 9 (3 Punkte)

Wir nennen eine kontextfreie Grammatik $G = \langle V, T, P, S\rangle$ *Automatengrammatik*, falls $P \subseteq V \times (T^* \cdot (V \cup \{\varepsilon\})$ gilt.

(a) Geben Sie zwei Beispiele für Automatengrammatiken an.

(b) Geben Sie zwei Beispiele für kontextfreie Grammatiken an, die keine Automatengrammatiken sind.

(c) Beschreiben Sie ein allgemeines Verfahren, um zu einer gegebenen Automatengrammatik einen endlichen Automaten zu konstruieren, der die Sprache der Grammatik akzeptiert.

Lösung

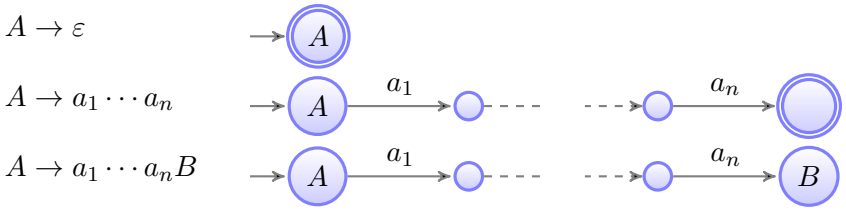
$P \subseteq V \times (T^* \cdot (V \cup \{\varepsilon\})$ bedeutet, dass die rechte Seite jeder Produktion ein Wort aus der Sprache $T^* \cdot (V \cup \{\varepsilon\}$ sein muss. Somit darf die rechte Seite jeder Produktion aus einer beliebigen Anzahl von Terminalsymbolen bestehen, denen optional ein einzelnes Nonterminal folgt.

(a) $\langle \{A\}, \{a\}, \{A \to aaA \mid \varepsilon\}, A \rangle$
$\langle \{A\}, \{a\}, \{A \to aaaa\}, A \rangle$

(b) $\langle \{A\}, \{a, b\}, \{A \to aAb \mid \varepsilon\}, A \rangle$
$\langle \{A\}, \{a\}, \{A \to Aa \mid \varepsilon\}, A \rangle$

(c) Wir ordnen jedem Nonterminal einen entsprechend benannten Zustand zu. Weiters benötigen wir Hilfszustände, die keine Entsprechung in der Grammatik besitzen und die wir in der graphischen Darstellung unbenannt lassen. Als Startzustand wählen wir jenen Zustand, der der Startvariablen S entspricht. Die Menge T der Terminalsymbole bildet das Eingabealphabet des Automaten.

Die Produktionen lassen sich wie folgt in Zustände und Übergänge übersetzen. Dabei stehen A und B für Nonterminale und $a_1, \ldots, a_n$ (mit $n \geq 1$) für Terminalsymbole.



Aufgabe 10 (5 Punkte)

Seien *Frisst*/2, *Vogel*/1, *Futter*/1 und *Groß*/1 Prädikatensymbole sowie *samen* und *beeren* Konstantensymbole mit folgender Bedeutung:

Vogel(x)	... x ist ein Vogel	*Frisst*(x, y)	... x frisst y
Futter(x)	... x ist Futter	*samen*	... Samen
Groß(x)	... x ist groß	*beeren*	... Beeren

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

(a) Manche Vögel fressen Samen aber keine Beeren.

(b) Kein Futter wird von allen großen Vögeln gefressen.

Sei weiters folgende Interpretation I gegeben:

$$\begin{aligned}
\mathcal{U} &= \{\text{Kakadu, Kiwi, Ara, Specht, Spatz, Nüsse, Körner, Samen, Würmer, Beeren,} \\
&\qquad \text{Früchte}\} \\
I(\mathit{Vogel}) &= \{\text{Kakadu, Kiwi, Ara, Specht}\} \\
I(\mathit{Futter}) &= \{\text{Körner, Samen, Würmer, Beeren, Früchte}\} \\
I(\mathit{Groß}) &= \{\text{Kakadu, Specht, Ara, Nüsse}\} \\
I(\mathit{Frisst}) &= \{(\text{Kakadu, Beeren}), (\text{Kakadu, Samen}), \\
&\qquad (\text{Kiwi, Früchte}), (\text{Kiwi, Beeren}), \\
&\qquad (\text{Ara, Würmer}), (\text{Ara, Samen}), \\
&\qquad (\text{Specht, Körner}), (\text{Specht, Samen}), (\text{Specht, Würmer})\} \\
I(\mathit{beeren}) &= \text{Beeren} \\
I(\mathit{samen}) &= \text{Samen}
\end{aligned}$$

Geben Sie an, ob die nachfolgenden Formeln in dieser Interpretation wahr oder falsch sind. Begründen Sie Ihre Antwort.

(c) $\forall x\,((\mathit{Groß}(x) \wedge \mathit{Futter}(x)) \supset \mathit{Frisst}(x, \mathit{samen}))$

(d) $\forall x \exists y\,(\mathit{Vogel}(x) \supset (\mathit{Futter}(y) \wedge \mathit{Frisst}(x, y)))$

(e) $\forall x \exists y\,(\mathit{Futter}(x) \supset (\mathit{Vogel}(y) \wedge \mathit{Groß}(y) \wedge \mathit{Frisst}(y, x)))$

Lösung

(a) $\exists x\,(\mathit{Vogel}(x) \wedge \mathit{Frisst}(x, \mathit{samen}) \wedge \neg \mathit{Frisst}(x, \mathit{beeren}))$

„Manche“ bedeutet eigentlich auch, dass nicht alle betroffen sind. Eine präzisere Lösung wäre daher:

$\exists x\,(\mathit{Vogel}(x) \wedge \mathit{Frisst}(x, \mathit{samen}) \wedge \neg \mathit{Frisst}(x, \mathit{beeren}))$
$\wedge \neg \forall x\,(\mathit{Vogel}(x) \supset (\mathit{Frisst}(x, \mathit{samen}) \wedge \neg \mathit{Frisst}(x, \mathit{beeren})))$

(b) $\neg \exists x\,(\mathit{Futter}(x) \wedge \forall y\,((\mathit{Vogel}(y) \wedge \mathit{Groß}(y)) \supset \mathit{Frisst}(y, x)))$ oder
$\forall x\,(\mathit{Futter}(x) \supset \exists y\,(\mathit{Vogel}(y) \wedge \mathit{Groß}(y) \wedge \neg \mathit{Frisst}(y, x)))$

(c) Übersetzung: Jedes große Futter frisst Samen
Diese Aussage ist wahr in I, da es kein großes Futter gibt, weswegen die Implikation immer wahr liefert.

(d) Übersetzung: Alle Vögel fressen (irgend)ein Futter.
Diese Aussage ist wahr in I. Kakadu, Kiwi, Ara und Specht fressen jeweils etwas aus $I(\mathit{Futter})$.

(e) Übersetzung: Jedes Futter wird von einem großen Vogel gefressen.
Diese Aussage ist falsch in I. Früchte frisst nur der Kiwi, dieser ist zwar ein Vogel aber er ist nicht groß.

Aufgabe 11 (5 Punkte)

Seien *Bekämpft*/2, *Jedi*/1, *Sith*/1 und *Dunkel*/1 Prädikatensymbole sowie *yoda* ein Konstantensymbol mit folgender Bedeutung:

$Jedi(x)$	... x ist ein Jedi-Ritter	$Bekämpft(x, y)$	... x bekämpft y
$Sith(x)$	... x ist ein Sith	*yoda*	... Yoda
$Dunkel(x)$	... x ist dunkel		

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

(a) Es gibt dunkle Sith, die alle Jedi-Ritter bekämpfen.

(b) Kein Jedi-Ritter bekämft alle Sith.

Sei weiters folgende Interpretation I gegeben:

$$\begin{aligned}
\mathcal{U} &= \{\text{Obi-Wan}, \text{Yoda}, \text{Luke}, \text{Anakin}, \text{DarthVader}, \text{DarthBane}, \text{DarthMaul},\\
&\quad \text{DarthTyrannus}, \text{DarthSidious}, \text{Yaddle}\}\\
I(\mathit{Jedi}) &= \{\text{Obi-Wan}, \text{Yoda}, \text{Luke}, \text{DarthVader}, \text{Yaddle}\}\\
I(\mathit{Sith}) &= \{\text{DarthVader}, \text{DarthBane}, \text{DarthMaul}, \text{DarthTyrannus}\}\\
I(\mathit{Dunkel}) &= \{\text{DarthBane}, \text{DarthVader}, \text{Anakin}\}\\
I(\mathit{Bekämpft}) &= \{(\text{Obi-Wan}, \text{DarthTyrannus}), (\text{Obi-Wan}, \text{DarthBane}),\\
&\quad (\text{Obi-Wan}, \text{DarthMaul}),\\
&\quad (\text{Yoda}, \text{DarthMaul}), (\text{Yoda}, \text{DarthTyrannus}),\\
&\quad (\text{Luke}, \text{Anakin}), (\text{Luke}, \text{DarthTyrannus}),\\
&\quad (\text{DarthVader}, \text{Luke}), (\text{DarthVader}, \text{DarthVader}),\\
&\quad (\text{Anakin}, \text{Obi-Wan}), (\text{Anakin}, \text{Luke}), (\text{Anakin}, \text{DarthVader}),\\
&\quad (\text{DarthBane}, \text{Obi-Wan}), (\text{DarthBane}, \text{Luke}), (\text{DarthBane}, \text{Yoda}),\\
&\quad (\text{DarthSidious}, \text{Yaddle})\}\\
I(\mathit{luke}) &= \text{Luke}\\
I(\mathit{darthvader}) &= \text{DarthVader}
\end{aligned}$$

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

(c) $\exists x\,(\mathit{Dunkel}(x) \wedge \mathit{Sith}(x) \wedge (\mathit{Bekämpft}(x, \mathit{luke}) \not\equiv \mathit{Bekämpft}(x, \mathit{darthvader})))$

(d) $\forall x \exists y\,(\mathit{Jedi}(x) \wedge \mathit{Sith}(y) \wedge \mathit{Bekämpft}(x, y))$

(e) $\forall x\,(\mathit{Dunkel}(x) \supset \mathit{Bekämpft}(x, \mathit{luke}))$

Lösung

(a) $\exists x\,(\mathit{Sith}(x) \wedge \mathit{Dunkel}(x) \wedge \forall y\,(\mathit{Jedi}(y) \supset \mathit{Bekämpft}(x, y)))$

(b) $\neg\exists x(\mathit{Jedi}(x) \wedge \forall y(\mathit{Sith}(y) \supset \mathit{Bekämpft}(x, y)))$ oder
$\forall x(\mathit{Jedi}(x) \supset \exists y(\mathit{Sith}(y) \wedge \neg\mathit{Bekämpft}(x, y)))$

(c) Übersetzung: Es gibt einen dunklen Sith, der entweder Luke oder Darth Vader bekämpft.
Diese Aussage ist wahr in I, da Darth Bane Luke, nicht aber Darth Vader bekämpft.

(d) Übersetzung: Alles ist ein Jedi-Ritter und bekämpft einen Sith.
(Die Formel bedeutet nicht: Alle Jedi-Ritter bekämpfen einen Sith! Dafür müsste die erste Konjunktion durch eine Implikation ersetzt werden.)
Diese Aussage ist falsch in I.
Gegenbeispiel: Anakin kommt im Universum vor, ist aber kein Jedi-Ritter (Anakin $\notin I(\mathit{Jedi})$). Damit ist die Konjunktion für diese Wahl von x immer falsch, daher auch die $\exists$-quantifizierte Formel.

(e) Übersetzung: Alles Dunkle bekämpft Luke.
Diese Aussage ist wahr in I, da $I(\mathit{Dunkel}) = \{\text{DarthBane}, \text{DarthVader}, \text{Anakin}\}$ und jeder der drei laut $I(\mathit{Bekämpft})$ Luke bekämpft.

Aufgabe 12 (3 Punkte)

Zeigen Sie, dass die Formeln

$$\forall y\exists x(\neg A(y) \vee (B(x) \wedge \neg C(y, x))) \quad \text{und} \quad \neg\exists x(A(x) \wedge \forall y(B(y) \supset C(x, y)))$$

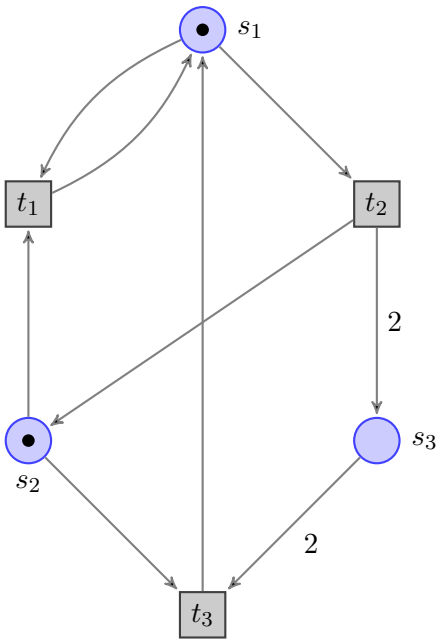
logisch äquivalent sind. Geben Sie an, welches logische Gesetz Sie in jedem Schritt angewendet haben.

Lösung

$\neg\exists x(A(x) \wedge \forall y(B(y) \supset C(x, y)))$	
$= \neg\exists x(\forall y A(x) \wedge \forall y(B(y) \supset C(x, y)))$	$\forall x F = F$ wenn x nicht in F vorkommt
$= \neg\exists x\forall y(A(x) \wedge (B(y) \supset C(x, y)))$	$\forall x(F \wedge G) = \forall x F \wedge \forall x G$
$= \forall x\neg\forall y(A(x) \wedge (B(y) \supset C(x, y)))$	$\neg\exists x F = \forall x\neg F$
$= \forall x\exists y\neg(A(x) \wedge (B(y) \supset C(x, y)))$	$\neg\forall x F = \exists x\neg F$
$= \forall x\exists y\neg(A(x) \wedge (\neg B(y) \vee C(x, y)))$	$F \supset G = \neg F \vee G$
$= \forall x\exists y(\neg A(x) \vee (B(y) \wedge \neg C(x, y)))$	$\neg(F \vee G) = \neg F \wedge \neg G$, $\neg(F \wedge G) = \neg F \vee \neg G$
$= \forall z\exists y(\neg A(z) \vee (B(y) \wedge \neg C(z, y)))$	$\forall x F[x] = \forall y F[y]$ falls y nicht in $F[x]$ auftritt
$= \forall z\exists x(\neg A(z) \vee (B(x) \wedge \neg C(z, x)))$	$\exists x F[x] = \exists y F[y]$ falls y nicht in $F[x]$ auftritt
$= \forall y\exists x(\neg A(y) \vee (B(x) \wedge \neg C(y, x)))$	$\forall x F[x] = \forall y F[y]$ falls y nicht in $F[x]$ auftritt

Aufgabe 13 (3 Punkte)

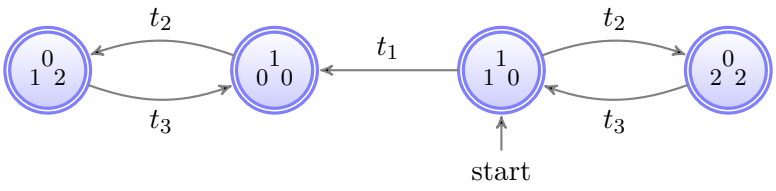
Gegeben sei das folgende Petri-Netz mit Anfangsmarkierung.



Beschreiben Sie alle möglichen Reihenfolgen, in der die Transitionen feuern und die Markierungen auftreten können. Am einfachsten lässt sich das durchführen, wenn Sie die Namen der Transitionen als Alphabet und die Markierungen als Zustände auffassen und einen Automaten bzw. einen regulären Ausdruck verwenden.

Lösung

Die Abläufe im Petrinetz lassen sich durch einen Automaten mit (in diesem Fall) endlich vielen Zuständen beschreiben. Die Markierungen bilden die Zustände, die Transitionen das Alphabet. Erhält man aus einer Markierung m durch Feuern einer Transition t eine Markierung m', dann gibt es einen Übergang beschriftet mit t vom Zustand für m zu jenem für m'.
Wir stellen jede Markierung durch drei Zahlen ${}_{n_2}{}^{n_1}{}_{n_3}$ dar, wobei n_i die Anzahl der Marken in der Stelle s_i angibt. Die endlichen Reihenfolgen, in denen die Transitionen feuern können, entsprechen der Sprache des folgenden Automaten.



Diese Reihenfolgen lassen sich auch durch den folgenden regulären Ausdruck in algebraischer Notation beschreiben:

$$(t_2t_3)^* + (t_2t_3)^*t_2 + (t_2t_3)^*t_1(t_2t_3)^* + (t_2t_3)^*t_1(t_2t_3)^*t_2$$

Verbale Beschreibung: t_2 und t_3 können abwechselnd feuern. Weiters kann t_1 ein einziges Mal entweder ganz zu Beginn oder nach t_3 feuern.

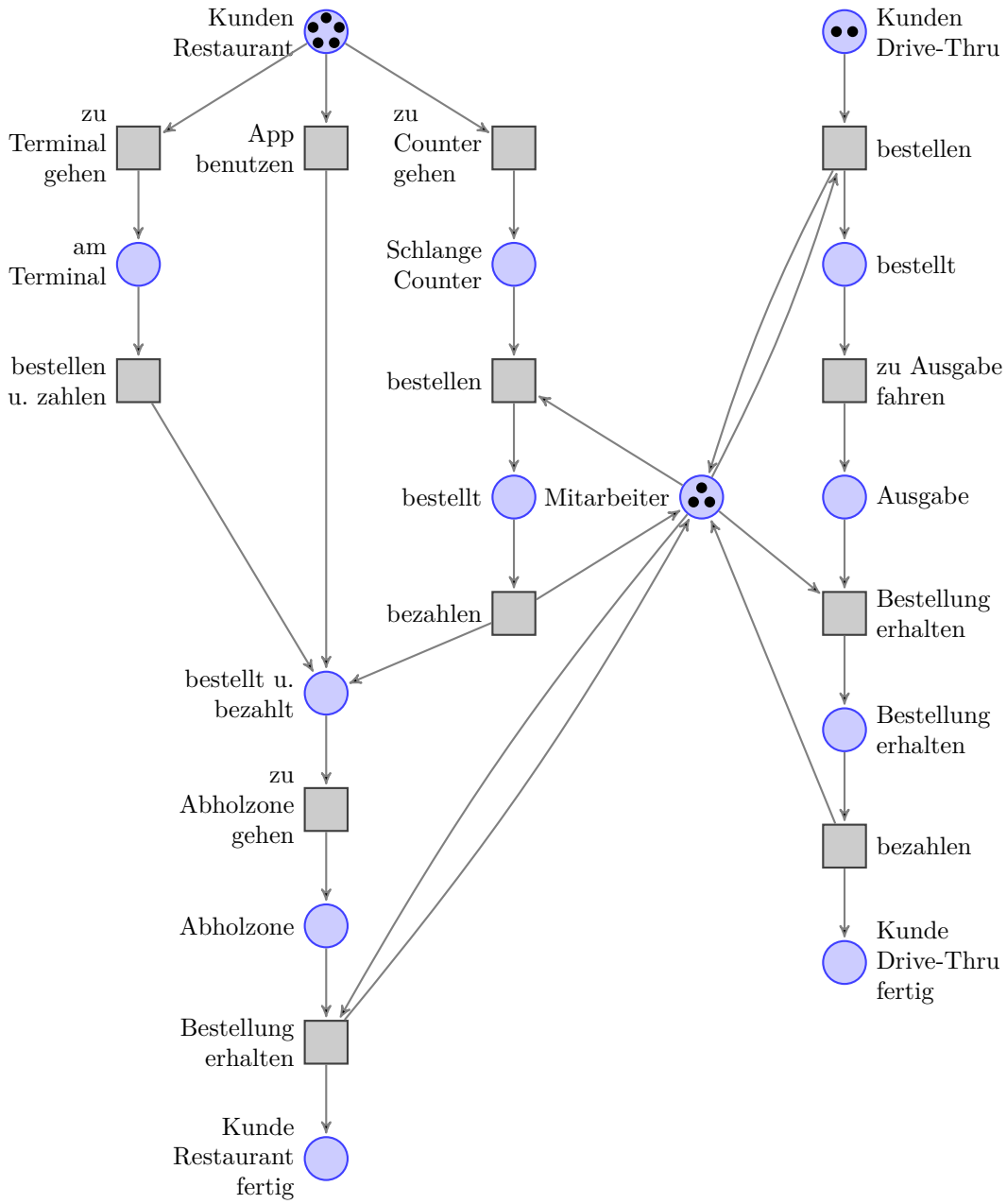
Aufgabe 14 (4 Punkte)

Eine Fastfood-Kette bietet folgende Möglichkeiten um zu bestellen.

- Will man sein Auto nicht verlassen, fährt man zum *Drive-Thru* und gibt dort über eine Gegensprechanlage seine Bestellung bei einem Mitarbeiter auf. Anschließend fährt man zur Ausgabe, wo ein Mitarbeiter zuerst den offenen Betrag kassiert und anschließend die Bestellung aushändigt.
- Im Restaurant selbst gibt es drei weitere Möglichkeiten:
  - Wählt man die klassische Art zu bestellen, geht man zum *Counter* und gibt dort die Bestellung bei einem Mitarbeiter auf. Hat man bezahlt, so erhält man eine Abholnummer und wartet anschließend in der Abholzone auf die fertige Bestellung.
  - Alternativ geht man zu einem *Bestellterminal*, wählt dort am Bildschirm die gewünschten Speisen aus und zahlt anschließend mit Bankomatkarte. Dann erhält man eine Abholnummer und wartet anschließend in der Abholzone auf die fertige Bestellung.
  - Oder man bestellt und bezahlt via *Handy-App*. Auch in diesem Fall erhält man eine Abholnummer für die Abholzone.

Modellieren Sie dieses Bestellkonzept mit Hilfe eines Petri-Netzes mit folgenden Rahmenbedingungen: Es gibt drei Mitarbeiter, die flexibel beim Drive-Thru, am Counter und in der Abholzone eingesetzt werden. Es gibt einen Counter und ein Bestellterminal. Zu Beginn befinden sich zwei Autos am Drive-Thru und fünf Kunden im Eingangsbereich des Restaurants, die sich noch für eine der drei Möglichkeiten, im Restaurant zu bestellen, entscheiden müssen.
Hinweis: Überlegen Sie, wann ein Mitarbeiter in einen Prozess involviert ist und bilden Sie in Ihrem Netz ab, wann ein Mitarbeiter „belegt“ ist.

Lösung


Kunden Restaurant
zu Terminal gehen
App benutzen
zu Counter gehen
am Terminal
Schlange Counter
bestellen u. zahlen
bestellen
bestellt
Mitarbeiter
bezahlen
bestellt u. bezahlt
zu Abholzone gehen
Abholzone
Bestellung erhalten
Kunde Restaurant fertig
Kunden Drive-Thru
bestellen
bestellt
zu Ausgabe fahren
Ausgabe
Bestellung erhalten
Bestellung erhalten
bezahlen
Kunde Drive-Thru fertig