AtoCC - Teil AutoEdit
Aufgabensammlung

Michael Hielscher
Christian Wagenknecht

Einleitung

Diese Aufgabensammlung ist für Schüler/innen und Studierende der Informatik gedacht, ebenso wie für Dozenten. Sie enthält einige Grundaufgaben für die bekannten Automatentypen (DEA, NEA, DKA, NKA, TM, Moore, Mealy) und zugehörige Lösungshinweise. Für jede Aufgabe ist ihr Schwierigkeitsgrad angegeben.

Jedes Kapitel ist einem der o. g. Automatentypen gewidmet. Es beginnt mit der zugehörigen Definition und ein paar Merk-Regeln. Dies kann entsprechende Theorie-Kenntnisse nicht ersetzen, soll vielmehr daran erinnern. Außerdem werden gerade die Automatendefinitionen ausgewählt, die in AutoEdit eingebaut sind.

AutoEdit ist ein zentraler Bestandteil von AtoCC, der Lehr-/Lern-Umgebung für theoretische Informatik und visualisierte Übersetzer-Konstruktion. Zur Bearbeitung der Aufgaben sollen Sie AtoCC auf Ihrem PC installiert haben. Kein Problem: AtoCC kann kostenlos aus dem Internet (http://www.atocc.de/) bezogen werden. Die Installation (als Administrator) erfordert, nur wenige Sekunden.

Auf der genannten Website finden sich viele nützliche Materialien. Zur Einsendung Ihrer Vorschläge und Kritik gibt es ein Web-Formular. Für Anregungen sind wir Ihnen stets sehr dankbar.

Wir hoffen sehr, dass AutoEdit Ihr Begleiter ist, wenn Sie abstrakte Automaten entwerfen, auf verschiedene Eingabewörter anwenden und simulieren, in Grammatiken und andere Automatentypen transformieren, Ihre Lösungen durch maschinellen Vergleich mit Musterlösungen bewerten und verbessern, ...

In Präsentationen von Automaten-Modellen sollten auch die Übergangsgraphen gut aussehen, auf der Folie, im Web oder in gedruckten Materialien (z. B. Vorlesungsskript, auch mit LaTeX). Schöne-Graphen-Zeichnen kann AutoEdit ganz besonders gut, gern auch und in vielen Ausgabeformaten.

Viel Freude! M. Hielscher und Chr. Wagenknecht, Mai 2007

AutoEdit Aufgaben

Die in diesem Kapitel vorgestellten Übungsaufgaben sollen mit AutoEdit bearbeitet werden. Der Schwierigkeitsgrad jeder Aufgabe wird durch ein farbiges Rechteck am Aufgabenbeginn dargestellt (grün = einfach, blau = mittel und schwarz = schwer).

Endliche Automaten

Endliche Automaten werden in zwei Gruppen geteilt:

- Deterministische endliche Automaten (***DEA***)
- Nichtdeterministische endliche Automaten (***NEA***)

Ein ***DEA*** ist ein Quintupel:

$M = (Q,\Sigma,\delta,q_0,E)$, mit

Q ... endliche Menge von Zuständen,
Σ ... Eingabealphabet,
δ ... totale Überführungsfunktion, $Q \times \Sigma \rightarrow Q$,
q_0 ... Anfangszustand ($q_0 \in Q$),
E ... endliche Menge von Endzuständen ($E \subseteq Q$)

Die Sprache L, die durch den Automaten beschrieben wird:

$L(M) = \{w \mid w \in \Sigma^* \text{ und } (q_0,w) \vdash^* (q_e,\varepsilon) \text{ und } q_e \in E\}$

DEA Aufgabe 1:

Bearbeiten Sie den nachfolgend beschriebenen Automaten mit AutoEdit. Machen Sie sich mit dem Programm vertraut und simulieren Sie die Arbeitsweise des Automaten mit einigen Eingabewörtern:

$M = (\{z_0,z_1,z_2,z_3\}, \{a,b\}, \delta, z_0, \{z_3\})$

δ wird in Form einer Übergangstabelle gegeben:

δ	a	b
z_0	z_1	z_3
z_1	z_2	z_0
z_2	z_3	z_1
z_3	z_0	z_2

DEA Aufgabe 2:

Entwickeln Sie einen deterministischen endlichen Automaten, der die Sprache aller Wörter über {0,1}* beschreibt, welche mit dem Teilwort „10“ enden.

Beispieleingaben:

1000111110110	wird akzeptiert
1011101000111	wird nicht akzeptiert

Hinweis:
Betrachten Sie zunächst besondere Wörter wie etwa „0“ oder „“ (leeres Wort) und entscheiden Sie, ob diese Wörter zur beschriebenen Sprache gehören oder nicht.

DEA Aufgabe 3:

Gegeben sei die Sprache S, die aus allen Wörtern der Form a^n besteht, wobei n durch 3 oder durch 4 (oder durch beide) teilbar ist. Geben Sie einen zugehörigen DEA an.

$\Sigma = \{a\}$

Beispieleingaben:

aaa	wird akzeptiert
aaaa	wird akzeptiert
aaaaa	wird nicht akzeptiert
aaaaaa	wird akzeptiert

DEA Aufgabe 4:

Entwickeln Sie einen DEA, der alle Wörter akzeptiert, die eine durch 4 teilbare natürliche Zahl repräsentieren. Dabei gilt, dass alle mehrstelligen Zahlen durch 4 teilbar sind, wenn die durch die letzten beiden Stellen gebildete Zahl durch 4 teilbar ist.

Beispieleingaben:

8	wird akzeptiert (8 durch 4 teilbar)
1342340	wird akzeptiert (40 durch 4 teilbar)
23452373	wird nicht akzeptiert (73 nicht durch 4 teilbar)

Hinweis:
Definieren Sie zunächst ein entsprechendes Eingabealphabet für diesen Automaten. Prüfen Sie auch welche einstelligen Zahlwörter das Teilbarkeitskriterium erfüllen.

AutoEdit Aufgaben: *Endliche Automaten*

Ein ***NEA*** ist ein Quintupel,

$M = (Q,\Sigma,\delta,q_0,E)$, mit

Q ... endliche Menge von Zuständen,
Σ ... Eingabealphabet,
δ ... totale Überführungsfunktion, $Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$,
q_0 ... Anfangszustand ($q_0 \in Q$),
E ... endliche Menge von Endzuständen ($E \subseteq Q$)

Die Sprache L, die durch den Automaten beschrieben wird:

$L(M) = \{w \mid w \in \Sigma^* \text{ und } (q_0,w) \vdash^* (q_e,\varepsilon) \text{ und } q_e \in E\}$

NEA Aufgabe 1:

Gegeben sei die Sprache aller Wörter über {a,b}*, die mit dem Teilwort „aa“ beginnen. Entwickeln Sie einen zugehörigen NEA.

Beispielwörter:

aa	wird akzeptiert
abaa	wird nicht akzeptiert
aabbaabb	wird akzeptiert

Hinweise:
Wählen Sie auf der „Typ“ Seite in AutoEdit den Automatentyp „NEA“ über die erste Schaltfläche.

NEA Aufgabe 2:

Bearbeiten Sie den im Folg. gegebenen NEA mit AutoEdit. Transformieren Sie den Automaten anschließend in einen DEA. Simulieren Sie die Analyse einiger Eingabewörter.

M = ({q_0,q_1,q_2,q_3,q_4}, {a,b}, δ, q_0, {q_2,q_4})
δ wird in Form einer Übergangstabelle gegeben:

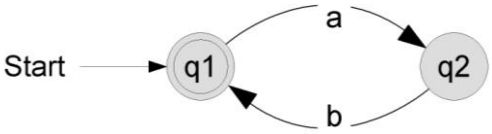
δ	a	b
q_0	{q_0,q_3}	{q_0,q_1}
q_1	{}	{q_2}
q_2	{q_2}	{q_2}
q_3	{q_4}	{}
q_4	{q_4}	{q_4}

Prüfen Sie, ob das Eingabewort „abaab“ vom Automaten akzeptiert wird und protokollieren Sie die Abarbeitung in einer Tabelle.

Hinweis:
Wählen Sie zunächst auf der „Typ“ Seite NEA über die erste Schaltfläche. Kehren Sie später wieder auf die „Typ“ Seite zurück um den entwickelten NEA nun in einen DEA zu transformieren.

NEA Aufgabe 3:

Welche Sprache beschreibt der nachfolgende NEA mit $\Sigma = \{a,b\}$?:



Hinweis:
Verwenden Sie einen regulären Ausdruck, um die Sprache anzugeben.

NEA Aufgabe 4:

Entwickeln Sie einen NEA, der alle Eingabewörter über {a,b}* akzeptiert, die mindestens 2 mal das Teilwort „ab“ enthalten. Die Position von „ab“ innerhalb des Eingabewortes spielt dabei keine Rolle.

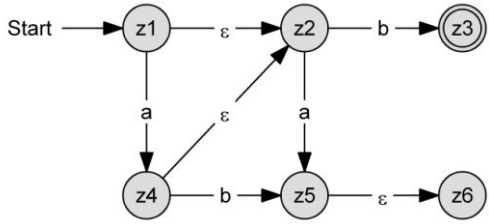
Beispielwörter:

abab	wird akzeptiert
abaa	wird nicht akzeptiert
abba**ab**b	wird akzeptiert

Hinweis:
Sie werden etwa 5 Zustände für diesen Automaten benötigen.

NEA Aufgabe 5:

Gegeben sei ein NEA über dem Alphabet {a,b}*, mit



- ☐ ***Geben Sie die vollständige Definition von M an.***
- ☐ ***Entscheiden Sie, ob M die Wörter „aa11a“, „aaa1a“ und „1a1a1a1a“ akzeptiert.***
- ☐ ***Wenden Sie das Verfahren zur Entwicklung der zu M gehörenden regulären Grammatik G ganz formal an, und testen Sie anschließend die o.g. Wörter.***

Kellerautomaten

Wie bei endlichen Automaten unterscheidet man deterministische (***DKA***) und nichtdeterministische Kellerautomaten (***NKA***). Im Gegensatz zu endlichen Automaten besitzen Kellerautomaten partielle Überführungsfunktionen (auf den Stack können beliebige Wörter geschrieben werden → unendlich viele Möglichkeiten).

DKA sind äquivalente Beschreibungen für LR(k) Sprachen.

Ein ***DKA*** ist ein 7-Tupel,

$M = (Q,\Sigma,\Gamma,\delta,q_0,k_0,E)$, mit

- Q ... endliche Menge von Zuständen,
- Σ ... Eingabealphabet,
- Γ ... Stackalphabet,
- δ ... partielle Überführungsfunkt., $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$,
- q_0 ... Anfangszustand ($q_0 \in Q$),
- k_0 ... Stackvorbelegungszeichen ($k_0 \in \Gamma$ und $k_0 \notin \Sigma$),
- E ... endliche Menge von Endzuständen ($E \subseteq Q$)

Die Sprache L, die durch den Automaten beschrieben wird:

$L(M) = \{w \mid w \in \Sigma^* \text{ und } (q_0,w,k_0) \vdash^* (q_e,\varepsilon,K) \text{ und } q_e \in E \text{ und } K \in \Gamma^*\}$

Merke:

- ☐ ε-Übergänge sind nur dann zulässig, wenn sie nicht parallel zu verbrauchenden Übergängen anwendbar sind.
- ☐ Nach dem Abtasten des Eingabewortes spielt der Kellerinhalt keine Rolle.

DKA Aufgabe 1:

Gegeben sei die Sprache aller Wörter über {a,b}*, welche mehr „a“ als „b“ besitzen. Entwickeln Sie einen entsprechenden DKA.

Beispielwörter:

aab	wird akzeptiert
abba	wird nicht akzeptiert
aabab	wird akzeptiert
bababaa	wird akzeptiert

Hinweis:
Verwenden Sie den Keller um gelesene Zeichen zu „merken“ und löschen Sie ein „a“ mit einem „b“ und umgekehrt.

DKA Aufgabe 2:

Palindrome sind Wörter, die sowohl vorwärts als auch rückwärts gelesen dasselbe Wort ergeben. Zum Beispiel: anna, otto, reliefpfeiler, rentner oder lagerregal.

Vereinfachend soll die Sprache der Palindrome über der Menge $(0+1)^+!(0+1)^+$ betrachtet werden. Das Zeichen „!“ soll also die Wortmitte kennzeichnen. Entwickeln Sie einen entsprechenden DKA.

Beispielwörter:

01!10	wird akzeptiert
01!00	wird nicht akzeptiert
001!100	wird akzeptiert

Hinweise:
Gehen Sie von einer Mindestwortlänge von 3 Zeichen aus. Verwenden Sie den Keller um sich die erste Hälfte des Eingabewortes zu „merken“.

DKA Aufgabe 3:

Geben Sie für die Sprache L der korrekt geklammerten arithmetischen Ausdrücke einen DKA an. L sei durch folgende Grammatik G definiert:

G = ({S}, {x,(,),[,]}, {S → x | SS | (S) | [S] | ε}, S)

Beispielwörter:

(x (x))	wird akzeptiert
[[x))	wird nicht akzeptiert
[x (x) [x (x)]]	wird akzeptiert

DKA Aufgabe 4:

Gegeben ist die kontextfreie Grammatik G = (N, T, P, s) mit N = { B, DT, AT }, T = { begin , end , dec , bez , ; } und P = {

B → begin DT AT end
AT → bez | bez ; AT
DT → DT dec ; | dec ;

}

Konstruieren Sie zu dieser Grammatik einen entsprechenden Kellerautomaten.

Beispielwörter:

begin dec ; bez end	wird akzeptiert
begin end	wird nicht akzeptiert
begin dec; dec; bez; bez end	wird akzeptiert

Hinweis:
AutoEdit „versteht“ bei der Simulation auch Leerzeichen zur besseren Lesbarkeit (obwohl sie nicht zum Eingabealphabet gehören).

NKA sind äquivalente Beschreibungen für Chomsky-Typ-2-Sprachen.

Ein ***NKA*** ist ein 7-Tupel,

$M = (Q,\Sigma,\Gamma,\delta,q_0,k_0,E)$, mit

Q ... endliche Menge von Zuständen,
Σ ... Eingabealphabet,
Γ ... Stackalphabet,
δ ... partielle ÜF., $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow P_{endlich}(Q \times \Gamma^*)$,
q_0 ... Anfangszustand ($q_0 \in Q$),
k_0 ... Stackvorbelegungszeichen ($k_0 \in \Gamma$ und $k_0 \notin \Sigma$),
E ... endliche Menge von Endzuständen ($E \subseteq Q$)

Die Sprache L, die durch den Automaten beschrieben wird:

$L(M) = \{w \mid w \in \Sigma^* \text{ und } (q_0,w,k_0) \vdash^* (q_e,\varepsilon,K) \text{ und } q_e \in E \text{ und } K \in \Gamma^*\}$

Merke:
Es gibt alternative gleichwertige NKA-Definitionen als die hier verwendete. Alle kontextfreien Sprachen (kfS) lassen sich mit NKA beschreiben.

NKA Aufgabe 1:

Zu jedem NKA gibt es einen äquivalenten NKA, der nur genau einen Zustand besitzt.
Entscheiden Sie, ob diese Aussage gilt und begründen Sie ihre Feststellung.

NKA Aufgabe 2:

Entwickeln Sie einen NKA für die Sprache L = {0^n1^{2n} | n>0}.

Beispielwörter:

011	wird akzeptiert
0011	wird nicht akzeptiert
000111111	wird akzeptiert

Hinweis:
Versuchen Sie das Verhältnis von doppelt so vielen 1'en wie 0'en im Keller abzubilden.

NKA Aufgabe 3:

Entwickeln Sie einen NKA für die im Folg. gegebene Sprache: L = {$a^kb^nc^m$ | k,n,m > 0 und k > n+m}.

Beispielwörter:

aaabc	wird akzeptiert, denn n=m=1, k=3
aabbcc	wird nicht akzeptiert, n=m=k=2
aaaabbc	wird akzeptiert, n=2, m=1, k=4

Hinweis:
Der Automat soll sich die vorhandenen „a" zunächst „merken", um dann die nachfolgenden Zeichen davon zu subtrahieren.

NKA Aufgabe 4:

Betrachten Sie die Sprache L = { ww | w $\in \Sigma^*$ }.
Die Sprache L' ist das Komplement der beschriebenen Sprache L. L' über {0,1}* enthält kein Wort der Form ww. Zeigen Sie, dass es sich um eine kontextfreie Sprache handelt.

Turing-Maschinen

Kellerautomaten reichen nicht aus, um kontextsensitive Sprachen (ksS) und Typ-0-Sprachen zu beschreiben. Hierfür werden Turingmaschinen benötigt (***TM***).

Eine deterministische TM ist ein 7-Tupel,

$M = (Q,\Sigma,\Gamma,\delta,q_0,\$,E)$, mit

Q ... endliche Menge von Zuständen,
Σ ... Eingabealphabet,
Γ ... Bandalphabet ($\Sigma \subseteq \Gamma \setminus \{\$\}$),
δ ... partielle ÜF., $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,N,R\}$,
q_0 ... Anfangszustand ($q_0 \in Q$),
$\$$ ... Bandvorbelegungszeichen ($\$ \in \Gamma$ und $\$ \notin \Sigma$),
E ... endliche Menge von Endzuständen ($E \subseteq Q$)

Die Sprache L, die durch den Automaten beschrieben wird:

$L(M) = \{w \mid w \in \Sigma^* \text{ und } (\alpha,q_0,w) \vdash^* (\beta,q_e,\gamma) \text{ und } \alpha \in \{\$\}^* \text{ und } \beta,\gamma \in \Gamma^* \text{ und } q_e \in E\}$

Merke:

- Die hier verwendete Definition erlaubt eine Kopfbewegung nach Links (L), Rechts (R) und auch gar keine (N). Es gibt aber auch andere, gleichberechtigte DTM-Definitionen.
- Eine DTM stoppt per Crash, d.h. $\delta(q_e,x)$ = n.d., im Zustand q_e. Die Kopfposition und der dann aktuelle Bandinhalt spielen keine Rolle.

TM Aufgabe 1:

Gegeben ist die Sprache L = {$a^n b^n c^n$ | n > 0 }. Entwickeln Sie eine TM für diese Sprache.

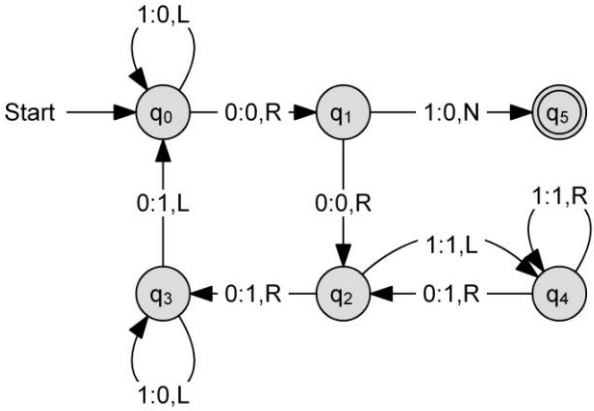
Beispielwörter:

abc	wird akzeptiert
abccc	wird nicht akzeptiert
aabbcc	wird akzeptiert

Hinweis:
Verwenden Sie ein zusätzliches Bandalphabetszeichen, um die Übergänge zwischen a b und b c zu kennzeichnen.

TM Aufgabe 2:

Experimentieren Sie mit so genannten „Busy-Beavern“ und versuchen Sie selbst eine solche TM mit 3 Zuständen zu entwickeln. Es gelten Σ={0,1} und $=0.
Implementieren Sie auch die gezeigte TM mit 6 Zuständen und simulieren Sie die Arbeitsweise mit einem leeren Eingabewort.



Moore-Maschinen

Moore-Maschinen sind DEA's sehr ähnlich, können jedoch mit jedem erreichten Zustand eine Ausgaben erzeugen. Ihr Verhalten entspricht aber nicht dem eines Akzeptors. Man spricht hingegen von endlichen Transducern (Transduktoren).

Die Ausgabe wird bei diesem Typ beim Erreichen eines neuen Zustands erzeugt, unabhängig auf welchem Weg dieser erreicht wurde (Ausgabe ist allein vom Zustand abhängig).

Eine ***MOORE***-Maschine ist ein 6-Tupel:

$M = (Q,\Sigma,\Delta,\delta,\lambda,q_0)$, mit

Q ... endliche Menge von Zuständen,
Σ ... Eingabealphabet,
Δ ... Ausgabealphabet,
δ ... totale Überführungsfunktion, $Q \times \Sigma \rightarrow Q$,
λ ... totale Funktion, $\lambda : Q \rightarrow \Delta$,
q_0 ... Anfangszustand ($q_0 \in Q$),

Merke:

- ☐ Ein Transducer berechnet eine Funktion f: $\Sigma^* \rightarrow \Delta^*$.
- ☐ Das Ausgabealphabet kann ganze Sätze (der deutschen Sprache) als Ausgabezeichen enthalten. Etwa "Bitte werfen Sie noch 2 Euro ein.\n“ (wobei \n für einen Zeilenumbruch in der Ausgabe steht).

MOORE Aufgabe 1:

Entwickeln Sie eine Moore-Maschine, die das Einerkomplement von Binärzahlen berechnet. Das Eingabealphabet ist {0,1}. Aus 0 wird 1 und aus 1 wird 0. Die Ausgabe soll das kommentieren.

Δ = {0, 1, Einerkomplement:}

Beispielwörter:	Ausgaben:
10001001	Einerkomplement: 01110110
1010	Einerkomplement: 0101
011	Einerkomplement: 100

Hinweis:
Die Maschine benötigt 3 Zustände wobei der erste Zustand nur für die allererste Ausgabe verwendet wird.

MOORE Aufgabe 2:

Entwickeln Sie einen Moore-Automaten mit Σ = {0, 1} und Δ = {a, b, c}, der (als letztes Zeichen) „a“ ausgibt, falls die Eingabe auf „010“ endet, „b“, falls die Eingabe auf „001“ endet, und sonst „c“.

Beispielwörter:	Ausgaben (ohne vorangehende Zeichen):
10001001	…b
1010	…a
011	…c

Hinweis:
Es spielt keine Rolle welche und wie viele Zeichen vor dem letzten Zeichen ausgegeben werden.

Mealy-Maschinen

Mealy-Maschinen sind den Moore-Maschinen sehr ähnlich, können jedoch mit jedem gelesenen Eingabezeichen eine Ausgabe erzeugen (Ausgabe ist sowohl vom Zustand als auch vom Eingabezeichen abhängig).

Eine ***MEALY***-Maschine ist ein 6-Tupel:

$M = (Q,\Sigma,\Delta,\delta,\lambda,q_0)$, mit

Q ... endliche Menge von Zuständen,
Σ ... Eingabealphabet,
Δ ... Ausgabealphabet,
δ ... totale Überführungsfunktion, $Q \times \Sigma \rightarrow Q$,
λ ... totale Funktion, $\lambda : Q \times \Sigma \rightarrow \Delta$,
q_0 ... Anfangszustand ($q_0 \in Q$),

Merke:

- ☐ Für jede Mealy-Maschine kann eine äquivalente Moore-Maschine angegeben werden und umgekehrt.
- ☐ Eine Mealy-Maschinen benötigt in der Regel weniger Zustände als eine äquivalente Moore-Maschine.

MEALY Aufgabe 1:

Von einer Mealy-Maschine liegen nachfolgende Tabellen vor. Geben Sie die vollständige Definition der Maschine und den zu den beiden Funktionen gehörenden Graphen an.

δ	a	b	c
q_0	q_1	q_0	q_2
q_1	q_2	q_1	q_3
q_2	q_0	q_0	q_2
q_3	q_2	q_2	q_3

λ	a	b	c
q_0	1	2	1
q_1	1	4	2
q_2	3	1	2
q_3	2	2	3

Hinweis:
Der praktische Verwendungszweck der beschriebenen Maschine steht nicht im Vordergrund.

MEALY Aufgabe 2:

Eine Binärzahl soll um eine Stelle nach rechts verschoben werden (right shift). Dies entspricht einer Division der Zahl mit 2 (die Anzahl der Stellen bleibt gleich). Entwerfen Sie eine Mealy-Maschine, die das jeweils eingegebene Zahlwort dementsprechend verarbeitet.

Beispielwörter:	Ausgaben:
10001111	01000111
1010	0101
1	0

MEALY Aufgabe 3:

Entwickeln Sie einen Getränkeautomaten, welcher 3 verschiedene Geldstücke entgegennimmt (1 Euro, 50 Cent und 20 Cent). Ein Getränk soll genau 1 Euro kosten und zu viel eingeworfenes Geld soll vom Automaten wieder ausgegeben werden.

Hinweis: Beginnen Sie mit der Vorgabe geeigneter Alphabete. Die Ausgaben sollen den Kunden über den verbleibenden Restbetrag informieren.

Anwendungsaufgaben

Die nachfolgenden Aufgaben sind praktischen Anwendungsfällen entlehnt. Die zu verwendenden Automatentypen sind in der Aufgabenstellung angegeben.

Anwendungsaufgabe 1:

Entwickeln Sie einen endlichen Automaten, der eine gültige Uhrzeit im Format 23:56 (hh:mm) akzeptiert. Entwickeln Sie zunächst einen Automaten für die 12h-Anzeige (von 00:00 bis 12:59) und anschließend einen Automaten für die 24h-Anzeige (von 00:00 bis 23:59).

Beispielwörter:

00:23	wird akzeptiert
45:12	wird nicht akzeptiert
:1234	wird nicht akzeptiert

Hinweis:
Der Doppelpunkt ist selbst ein Zeichen des Eingabealphabets.

Anwendungsaufgabe 2:

Ein Getränkeautomat verlangt für ein bestimmtes Getränk 60 Cent. Dieser Betrag muss in Münzen exakt gezahlt werden, da der Automat nicht wechseln kann. Akzeptiert werden 10-, 20- und 50 Cent Münzen. Entwickeln Sie einen endlichen Automaten mit dem Eingabealphabet $\Sigma = \{10, 20, 50\}$.

Beispielwörter:

10 50	wird akzeptiert
50 50 10	wird nicht akzeptiert
10 10 20 20	wird akzeptiert

Anwendungsaufgabe 3:

Ein Logistikunternehmen verwendet intern für Pakete die Angaben für Länge, Breite und Höhe in der Form 200x100x80 (ohne Maßeinheit). Ist ein Paket auf mindestens einer Seite sehr lang, muss es mit einem Extra-Fahrzeug transportiert werden. Ein Aufkleber auf jedem Paket wird von einem Automaten eingescannt. Der Automat bekommt eine Eingabe in der oben angegebenen Form. Entwickeln Sie einen endlichen Automaten so, dass nur die Pakete akzeptiert werden, bei denen alle Kanten kleiner als 200 sind.

Beispielwörter:

50x120x80	wird akzeptiert
210x90x90	wird nicht akzeptiert
10x190x10	wird akzeptiert

Hinweis:
Beachten Sie, dass auch das „x“ ein Zeichen des Eingabealphabets ist.

Anwendungsaufgabe 4:

In der Datenübertragung (zum Beispiel in einem Netzwerk) wird mit verschiedenen Mitteln versucht, Fehler in der Übertragung zu erkennen. Eine einfache Möglichkeit ist der Einsatz des so genannten Paritätsbits. Das Paritätsbit für eine Binärzahl ist 1, wenn die Anzahl der 1er ungerade ist, sonst ist das Paritätsbit 0. Ein Beispiel: 01100101 besitzt vier Einsen. Da 4 eine gerade Zahl ist, ist das Paritätsbit 0.
Entwickeln Sie eine Turingmaschine, welche als Eingabe eine Binärzahl beliebiger Länge entgegennimmt und das entsprechende Paritätsbit an dieses Wort unmittelbar anhängt.

Beispielwörter:	(Vorbelegungszeichen: $)
01	$01**1**$
00100110	$00100110**1**$
111100	$111100**0**$

Hinweis:
Der Kopf der TM soll am Ende auf dem ersten Zeichen des Resultatwortes stehen.

Anwendungsaufgabe 5:

Eine Binärzahl A soll mit einer Binärzahl B verglichen werden. Dafür soll eine Turingmaschine verwendet werden. Die Eingabe hat die Form „A=B“. Die Maschine soll nur dann akzeptieren, wenn die Gleichung stimmt.

Beispielwörter:

0=0	wird akzeptiert
101=010	wird nicht akzeptiert
1100=1100	wird akzeptiert

Hinweis:
Das Gleichheitszeichen „=“ ist ein Zeichen des Eingabealphabets.

Anwendungsaufgabe 6:

Entwerfen Sie eine Moore Maschine, um zwei Binärzahlen zu addieren (mit Übertrag). Die Eingabe der Binärstellen erfolgt paarweise von rechts nach links. Beispiel:

Zahl 1: 01001
Zahl 2: 00011

Eingabewort: 11 01 00 10 00

Die Maschine soll für obiges Zahlenbeispiel: „Summe: 00110“ liefern. Das Ergebnis muss ebenfalls von rechts nach links gelesen werden!

Σ = {00, 01, 10, 11}.
Δ = {0, 1, Summe: }

Hinweis:
Da die Eingabe von rechts nach links geschieht, kann in jedem Schritt der durch die Summation entstehende Übertrag einbezogen werden. Das letzte Eingabepaar soll immer „00“ sein (eine vorangestellte 0).

Anwendungsaufgabe 7:

Entwickeln Sie für die in Aufgabe 6 beschriebene Maschine nun eine äquivalente Mealy-Maschine. Die erste Ausgabe entfällt hier.

Σ = {00, 01, 10, 11}.
Δ = {0, 1}

Hinweis:
Sie benötigen hier nur 2 Zustände um „Übertrag“ oder „kein Übertrag“ abzubilden.