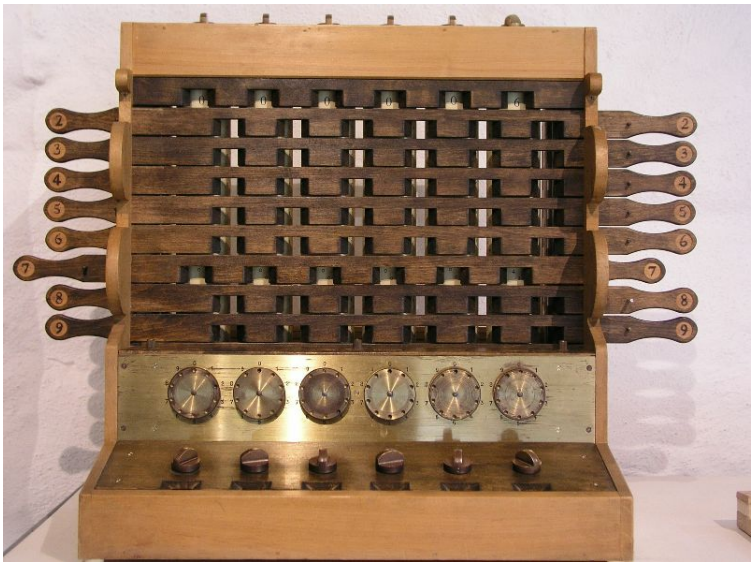
Tagung zur Schulinformatik

GI-FIBBB 2008

Workshop „Endliche Automaten”

Walter Gussmann (Paul-Natorp-Oberschule)



Rechenmaschine von W. Schickard (1592 - 1635)

Inhaltsverzeichnis

1 Endliche Automaten

Unser Alltag ist bestimmt durch zahlreiche „Maschinen”: Auto, Handy, Fahrstuhl, Parkuhr, ... Jede dieser Maschine befindet sich zu einer gewissen Zeit in einem definierten Zustand. Auf Ereignisse reagieren diese Maschinen je nach Zustand: Ein parkendes Auto reagiert auf die Betätigung des Gaspedals überhaupt nicht, während dasselbe Auto während der Fahrt beschleunigt wird.

Eine Beschreibung von Maschinen hängt deshalb vom jeweiligen Zustand ab, in dem sich diese befindet. *Alan Turing* (1912-1954) entwickelte für den Computer eine allgemeine Maschine („*Turing-Maschine*”), mit der zahlreiche Probleme gelöst werden können. Maschinenmodelle nennt man in der Informatik auch Automaten.

Beispiel: Videorekorder

Ein (vereinfachter) Videorekorder oder DVD-Abspielgerät besitzt einen Ein-/ Ausschalter, eine Play-, Stopp und Pausetaste sowie Tasten zum Vor- und Zurückspulen. Die Tasten des Gerätes verhalten sich – je nach Zustand – unterschiedlich. Im Display werden dem Benutzer Informationen über den aktuellen Zustand angezeigt. Gesucht ist eine vollständige Beschreibung des Abspielgerätes für alle möglichen Situationen.

1.1 Definition endlicher Automaten

Das Gerät kann als endlicher Automat beschrieben werden. Er besitzt die sechs Tasten (Eingaben) O (ON/OFF), S (Stopp), P (Play), W (Pause), V (Vorwärts) und R (Rückwärts) sowie fünf mögliche Zustände, wobei vereinfachend das Vor- und Zurückspulen als ein Zustand betrachtet wird. Der endliche Automat kann formal durch folgende Eigenschaften beschrieben werden.:

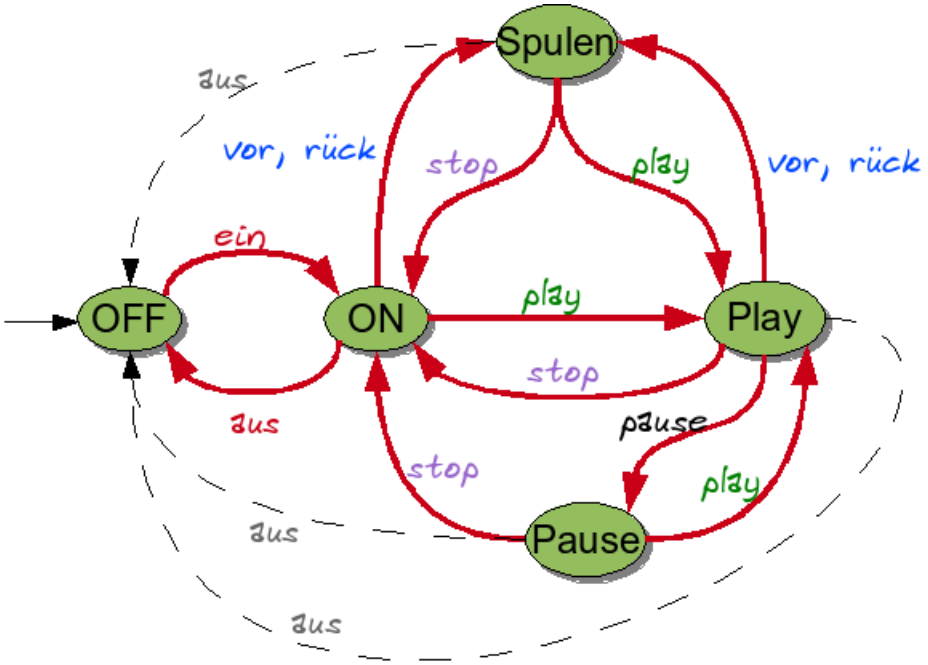
endliche Eingabemenge	E	$= \{O, S, P, V, R, W\}$
endliche Ausgabemenge	A	$= \{ON, OFF, STOP, PLAY, WARTEN, SPULEN\}$
Zustandmenge	Z	$= \{q0, q1, q2, q3, q4\}$
Anfangszustand	z_0	$= q0$
endlich viele Endzustände	Z_E	$= \{q0\}$
eine Übergangsfunktion	u :	$E \times Z \rightarrow Z$
eine Ausgabefunktion	a :	$E \times Z \rightarrow A$

Ein endlicher Automat wird duch das 7-Tupel $(E, A, Z, z_0, Z_E, u, a)$ beschrieben.

Es gibt jedoch auch Automaten, die keine Ausgabe ermöglichen (hier fehlt die Ausgabemenge), andere dagegen besitzen keine definierten Endzustände.

Die beiden Funktionen können entweder durch eine Wertetabelle oder Zustandsdiagramme beschrieben werden.

Zustandsgraph:



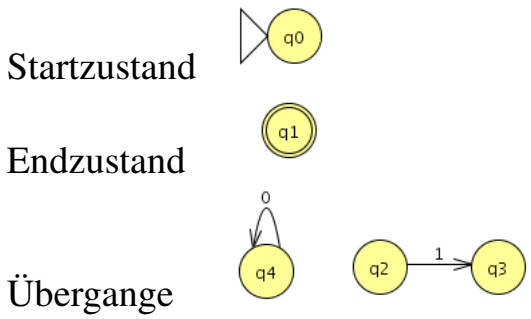
Wertetabelle:

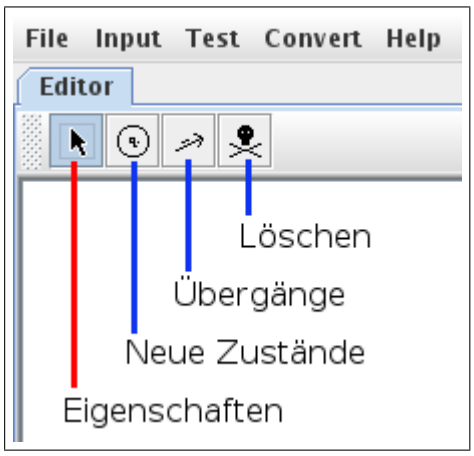
akt. Zustand	Eingabe	Folgezustand	Ausgabe
$q0$	O	$q1$	ON
$q0$	$*$	$q0$	
$q1$	S, W	$q1$	
$q1$	P	$q2$	$PLAY$
$q1$	V, R	$q4$	$SPULEN$
$q1$	O	$q0$	OFF
$q2$	P	$q2$	
	$u.s.w$		

Der Startzustand wird durch einen Pfeil markiert, der Endzustand ist dick oder doppelt umkreist (fehlt beim Videorekorder). Die Übergänge werden durch beschriftete Pfeile markiert.

1.2 Programmierung mit JFLAP

Üblicherweise werden endliche Automaten mit Hilfe spezieller Programme programmiert. An diesem Beispiel wird das Programm **JFLAP** vorgestellt, das neben endlichen Automaten auch die Implementierung von Sprachen und Turingmaschinen ermöglicht. Das Programm kann unter `http://www.jflap.org` heruntergeladen werden. Als Java-Applikation kann das Programm auf unterschiedlichen Betriebssystemen eingesetzt werden.





Menü von JFLAP (Automaten)

Das Programm JFLAP bietet drei unterschiedliche Typen endlicher Automaten an:

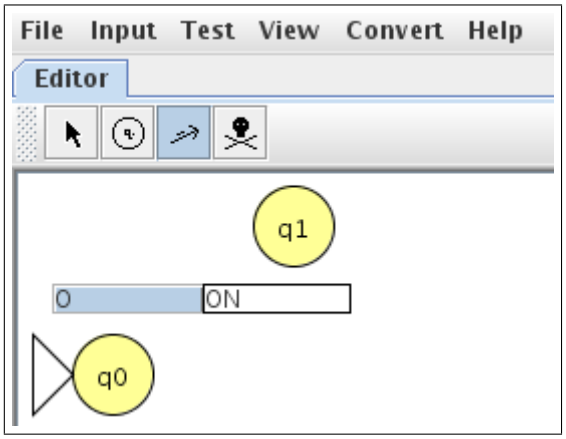
Finite Automaton: Hierbei handelt es sich um eine endliche Maschine ohne Ausgabefunktion. Sie wird immer dann eingesetzt, wenn es nur darum geht zu überprüfen, ob eine Eingabe den Automaten in einen gültigen Endzustand überführt. Man spricht dann auch von Akzeptoren.

Mealy Machine: Dieser Automat erzeugt zu jedem Übergang eine Ausgabe. Endzustände gibt es nicht.

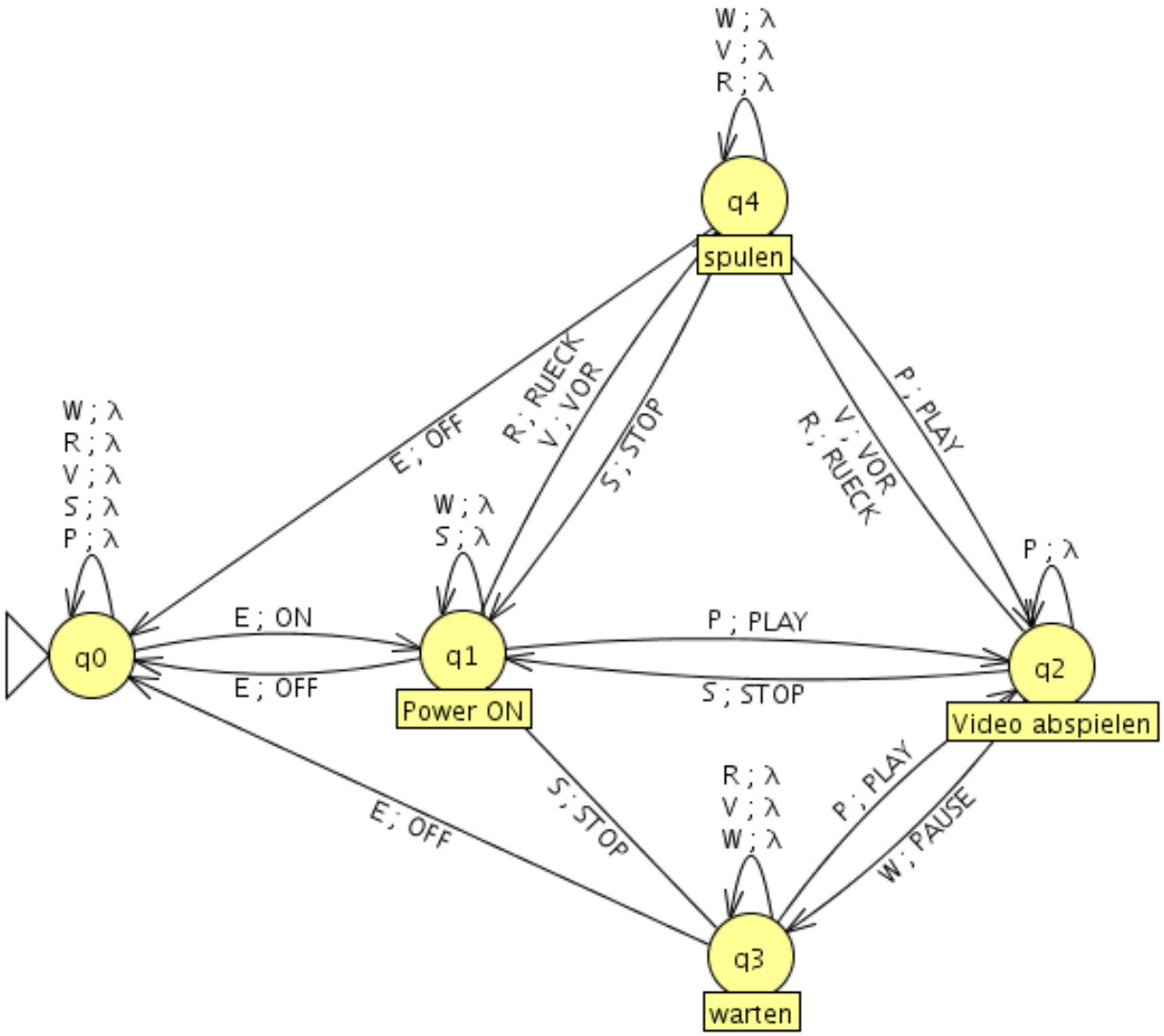
Moore Machine: Ein endlicher Automat, der für jeden Zustand eine Ausgabe erzeugt. Endzustände gibt es nicht.

Auswahlmenü von JFLAP

Für die Implementierung des Video-Automaten mit JFLAP können wir einen Mealy-Automaten verwenden. Den Zustandsgraphen erzeugt man durch Wahl der fünf Zustände, dem Hinzufügen eines Anfangszustandes sowie den entsprechenden Übergängen. Bei der Beschriftung eines Übergangs trägt man zunächst das Zeichen, dann den zugehörigen Ausgabewert ein. Die einzelnen Zustände können zusätzlich beschriftet werden. Ein übersichtliches Layout erhält man durch Verschieben der Zustände.



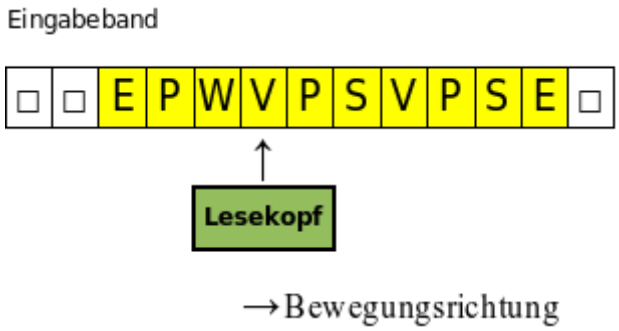
Programmentwicklung mit JFLAP - Erstellung von Übergängen



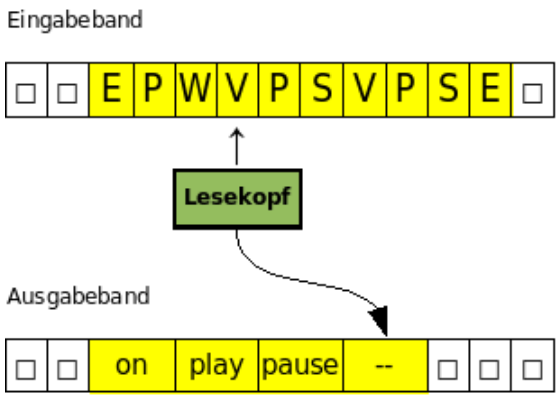
Zustandsgraph für den Video-Automaten (λ steht für keine Ausgabe)

1.3 Funktionsweise eines endlichen Automaten

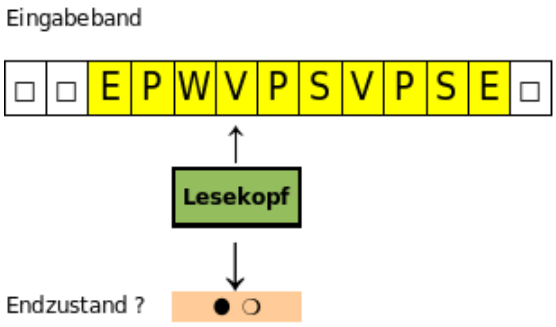
Ein endlicher Automat besteht aus einem Eingabeband, auf dem die Eingaben von links nach rechts abgelegt sind. Der Automat besitzt einen Lesekopf, der sich immer über genau einem Zeichen befindet. Zu Beginn ist dieser Kopf über dem Startzeichen. Beim Lesen eines Zeichens bewegt sich der Kopf um eine Stelle nach rechts.



Bei einem Mealy- und Moore-Automaten wird gleichzeitig die zugehörige Ausgabe auf einem Ausgabeband erzeugt. Akzeptoren haben stattdessen nur einen Ausgang, der einen Wahrheitswert liefert. Ist dieser auf True, dann wird die bisher erfolgte Eingabe akzeptiert.



Transduktor
Automat, der für jeden Übergang eine Ausgabe erzeugt (JFLAP: *Meleay-Automat*).



Akzeptor
Automat ohne Ausgabefunktion. Im Endzustand liefert er True (erkennender Automat).

Weitere Beispiele

Als beliebtes Beispiel eines endlichen Automaten stellt der Parkscheinautomat dar. Eine Darstellung hierzu finden Sie unter `http://www.pns-berlin.de/lk-ti-automat.html`.

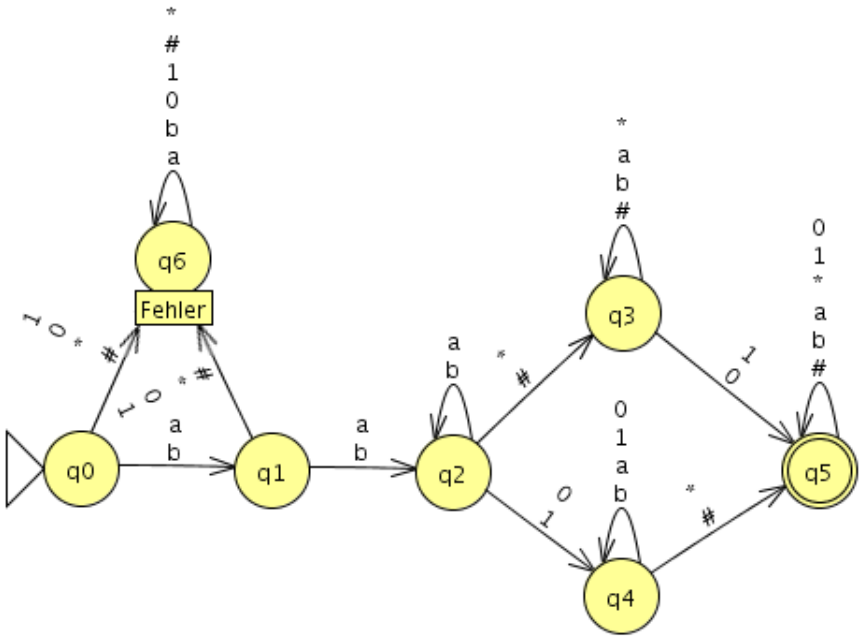
1.4 Beispiel: Passwortakzeptor

*Passwörter müssen in der Praxis gewissen Anforderungen genügen. So legt eine Firma etwa fest, dass die Passwörter mit mindestens zwei Buchstaben beginnen müssen und wenigstens ein Sonderzeichen und eine Ziffer enthalten. Um die Implementierung zu vereinfachen sollen nur die Buchstaben a und b, die Sonderzeichen # und * sowie die Ziffern 0 und 1 berücksichtigt werden.*

Formale Beschreibung mit Zustandsgraph

endliche Eingabemenge	E	$= \{a, b, \#, *, 0, 1\}$
endliche Zustandmenge	Z	$= \{q0, q1, q2, q3, q4, q5, q6\}$
Anfangszustand	z_0	$= q0$
endlich viele Endzustände	Z_E	$= \{q5\}$
eine Übergangsfunktion	u :	$E \times Z \rightarrow Z$
eine Ausgabefunktion	a :	$E \times Z \rightarrow A$

Hinweis: Übergänge, die auf einen Fehlerzustand führen, werden der Übersichtlichkeit wegen häufig weggelassen. In diesem Beispiel würde dann der Zustand $q6$ und alle Übergänge zu diesem Zustand fehlen.

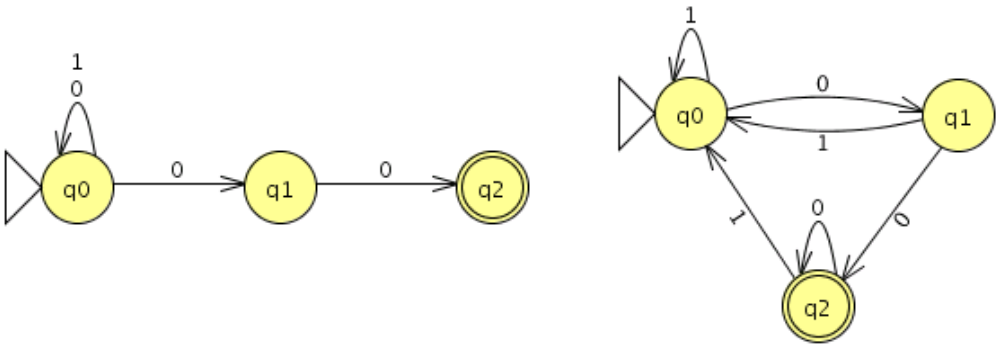


Zustandsgraph für den Passwortakzeptor

1.5 Deterministische und nichtdeterministische Automaten

Manchmal findet man auch Zustandsgraphen, bei denen von einem Zustand mehrere Übergänge für die gleiche Eingabe existieren. Es bleibt also zunächst unklar, welchen Folgezustand der Automat haben wird. Solche Automaten nennt man **nichtdeterministische endliche Automaten** (NEA). Gibt es dagegen zu jedem Zustand genau einen Nachfolgezustand, so spricht man von **deterministischen endlichen Automaten** (DEA).

In vielen Fällen ist die Implementierung eines NEA einfacher. Man kann zeigen, dass jeder nichtdeterministische endliche Automat in einen äquivalenten deterministischen Automaten überführt werden kann. JFLAP kann die Umwandlung automatisch erzeugen.



Beide Automaten akzeptieren ein binäres Wort, das mit der Ziffernfolge "00" endet.

1.6 Programmierung endlicher Automaten

Eine zustandsorientierte Programmierung endlicher Automaten beruht im Wesentlichen auf der Implementierung der Übergangs- und Ausgangsfunktion. Im einfachsten Fall ist es eine Folge geschachtelter Abfragen, die den aktuellen Zustand sowie einen Eingabewert auswerten.

```
if (zustand==0)
{ switch (eingabe)
  {
   case 'O': ausgabe="ON";
             zustand=1;
             break;
   default:  ausgabe="--";
  }
```

```
} ...
```

In funktionalen Sprachen kann die Übergangsfunktion übersichtlich durch „*pattern matching*" dargestellt werden.

```
type Eingabe = Char
data Zustand = Q0 | Q1 | Q2 | Q3 | Q4 | Q5
u :: Zustand -> Eingabe -> Zustand
u Q0 'O' = Q1
u Q0  _  = Q0  -- Taste ohne Funktion
...
```

1.7 Übungsaufgaben

Übung 1 Auf einem Parkplatz kostet das Parken 1,50 Euro. Ein Parkscheinautomat akzeptiert 50 Cent, 1 Euro und 2 Euro-Münzen. Nach Einwurf der korrekten Geldsumme liefert er das Ticket und das Restgeld. Er besitzt keine Abbruchtaste.

Übung 2 Von beliebigen binären Wörtern soll ein Automat erkennen, ob es sich um ein Wort mit gerader Parität (Anzahl der Einsen ist geradzahlig) handelt.

Beschreiben Sie formal den Akzeptor, skizzieren Sie den entsprechenden Zustandsgraphen und implementieren Sie diesen Automaten.

Beispiel:

```
0101101101    --> True
011001        --> False
```

Aufgabe 1 Gesucht ist ein Akzeptor, der überprüft, ob eine Zahl durch 5 oder 2 teilbar ist.

Aufgabe 2 Ein Automat zur Mustererkennung von Gen-Schnipseln soll überprüfen, ob ein Schnipsel die Zeichenkette ”GAT” oder ”ATA” enthält. Als Eingabemenge kommen die Symbole `A C G T` in Frage.

Aufgabe 3 Ein elektronisches Zahlenschloss besitzt die Kombination ”4711”. Gesucht ist ein Automat, der bei Eingabe dieser Kombination das Schloss öffnet, bei jeder anderen Eingabe einen Fehlercode ausgibt. Als Antwort auf eine Eingabe wird ein Sternchen ausgegeben. Eine Fehlermeldung darf erst nach Eingabe von vier Ziffern erfolgen.

Aufgabe 4 Ein Getränkeautomat bietet Kaffee für 1 Euro an. Der Automat akzeptiert 50 Cent- und 1 Euro-Stücke. Er besitzt außerdem zwei Tasten: Eine Taste **K** dient dazu, den Kaffee auszugeben. Mit der Taste **A** wird der Vorgang abgebrochen. In diesem Fall erhält der Kunde sein Geld zurück.

Beschreiben Sie diesen Getränkeautomaten mit den Begriffen eines endlichen Automaten und zeichnen Sie das Zustandsdiagramm.

Aufgabe 5 Mails enthalten oft „*smileys*" in Textform.

```
:-)    ☺
:-(    ☹
```

Gesucht ist ein Automat, der einen Text wiedergibt und dabei die Smileys in Textform durch die grafischen Symbole ersetzt. Es genügt, wenn die beiden angegebenen Smileys berücksichtigt werden und für ein beliebiges Zeichen 'x' verwendet wird.

Beispiel:

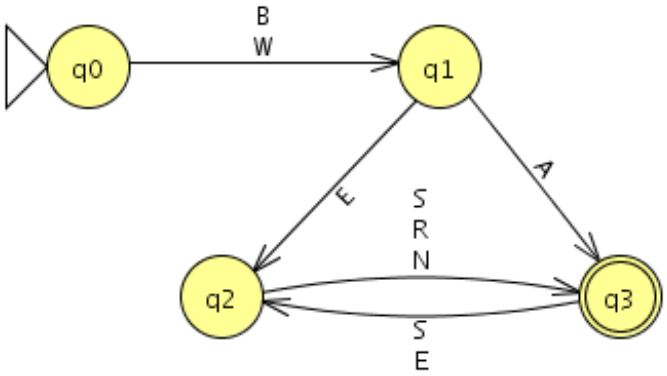
```
Text:    xxx:-):-xx:-(
Ausgabe: xxx☺:-xx☹
```

Aufgabe 6 Gesucht ist ein Akzeptor, der eine vorzeichenbehaftete, gerade Ganzzahl erkennt.

```
-123456   --> richtig
0123      --> falsch
+123.4    --> falsch
1248      --> richtig
```

Aufgabe 7 Beispiel für einen nichtdeterministischen Automaten (NEA). Gesucht ist der Zustandsgraph eines endlichen Automaten, der genau dann True liefert, wenn die Eingabe mit '1' beginnt und mit der Ziffernfolge '100' endet. Lösen Sie das Problem mit einem nichtdeterministischen und einem deterministischen Automaten.

Aufgabe 8 Beschreiben Sie den Automaten, der zu diesem Zustandsgraph passt und geben Sie einige gültige Wörter unterschiedlicher Länge an.



Aufgabe 9 Geben Sie einen Akzeptor an, der überprüft, ob eine Eingabe eine gültige reelle Zahl darstellt. Dabei wird folgendes Format akzeptiert (die in Klammern gesetzte Werte müssen nicht vorhanden sein):

```
reelle Zahl: [vz] zahl ["." zahl ["E" [vz] zahl]]
vz         : + oder -
zahl       : eine oder mehrere Ziffern 0..9
```

Hinweis: Solche Kurzschreibweisen sind durch die „*Erweiterte Backus-Naur-Form*" (kurz *EBNF*) vereinheitlicht worden und bilden eine formale Metasprache zur Spezifizierung. Weitere Informationen hierzu findet man im Internet.

2 Zustandsorientierte Programmierung mit Kara

Die Lernumgebung Kara wird an der ETH Zürich entwickelt und bietet inzwischen eine Fülle unterschiedlicher Varianten. Allen Versionen ist der Marienkäfer Kara gemeinsam, der über eine Fläche bewegt werden kann, auf der sich Kleeblätter, Pilze und Baumstümpfe befinden. Eine typische Aufgabe besteht darin, dass der Marienkäfer einen Wald umlaufen soll.

Dazu bietet Kara eine „Weltansicht", in der die unterschiedlichen Gegenstände mit der Maus auf einer $m \times n$ großen Fläche plaziert werden können. Die Steuerung des Käfers erfolgt im Programmeditor. Java-Kara wird mit Hilfe eines Javaprogramms bewegt; beim zustandsorientierten Programmieren erstellt man einen Zustandsgraphen. Dazu besitzt der Käfer unterschiedliche „Sensoren" wie etwa den Sensor: „rechts neben Kara ist ein Baumstumpf" oder „Kara befindet sich auf einem Feld mit einem Kleeblatt".

Beim Herunterladen des Javaprogramms müssen Sie auf die passende Java-Version achten.
Download: `http://www.educeth.ch/informatik/karatojava/`

2.1 Beispiel: Mustererkennung

Auf einer Wiese befinden sich Kleeblätter. Kara bewegt sich solange in Vorwärtsrichtung, bis er auf die Folge

trifft oder vor einem Baumstumpf steht. Findet Kara die gwünschte Folge, so bleibt er auf dem letzten Blatt stehen. Interpretiert man ein Kleeblatt als '1' und ein freies Feld als '0', dann entspricht dieses Problem der Suche nach der Ziffernfolge '101' in einem binären String.

Startzustand: 

Endzustand: 

Beschreibung des Suchautomaten

Im Programmeditor werden zunächst vier Zustände erzeugt. Dabei muss angegeben werden, welche Sensoren beachtet werden sollen. Da unser Kara nur nach

Kleeblättern sucht und vor einem Baumstumpf stehen bleibt, benötigen wir die Sensoren „*Kara steht auf Kleeblatt*" und „*Vor Kara befindet sich ein Baumstumpf*".

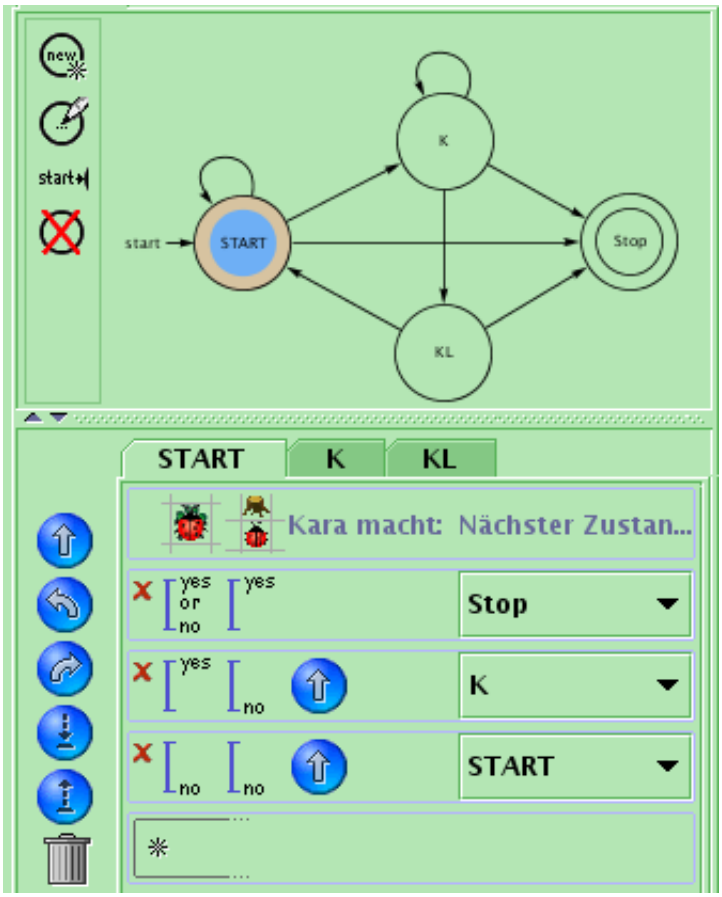
Der Algorithmus kann intuitiv entwickelt werden:

- Kara befindet sich zunächst im Anfangszustand START.
- Trifft er beim Vorwärtsgehen auf ein Kleeblatt, dann hat er ein akzeptiertes Zeichen (Kleeblatt) gelesen und wechselt in den Zustand K.
- Trifft er beim Weitergehen wieder auf ein Kleeblatt, so verharrt er in diesem Zustand K.
- Trifft er irgendwann auf ein leeres Feld, so hat er zwei aufeinanderfolgende gültige Zeichen gelesen (Zustand KL).
- Ist das nächste Zeichen ein Kleeblatt, dann hält Kara an (Endzustand STOP). Andernfalls fällt er in den Startzustand START zurück.
- Steht Kara in irgendeinem Zustand vor einem Baumstumpf, dann hält Kara an (Endzustand STOP).
- Für die Implementierung werden also 4 Zustände benötigt.

Damit kann der zugrundeliegende endliche Automat (Akzeptor) wie folgt beschrieben werden:

- Eingabemenge E = { Klee, Leer, Baum }
- Zustandsmenge Z = {$START$, K, KL, $STOP$}
- Anfangszustand $START$
- Endzustände Z_E = { $STOP$ }
- Übergangsfunktion: $u : E \times Z \rightarrow Z$
- Ausgabefunktion: $a : E \times Z \rightarrow A$

Automat: (E, Z, $z0$, Z_E, u, a)

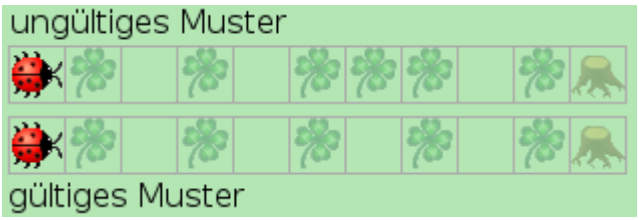


Zustandsdiagramm mit Übergangstabellen für den Startzustand

2.2 Übungsaufgaben

Implementieren Sie die Beispiele mit Kara. Weitere Übungsaufgaben unterschiedlicher Schwierigkeit (mit Lösungen) finden Sie im Karamodell unter Aufgaben.

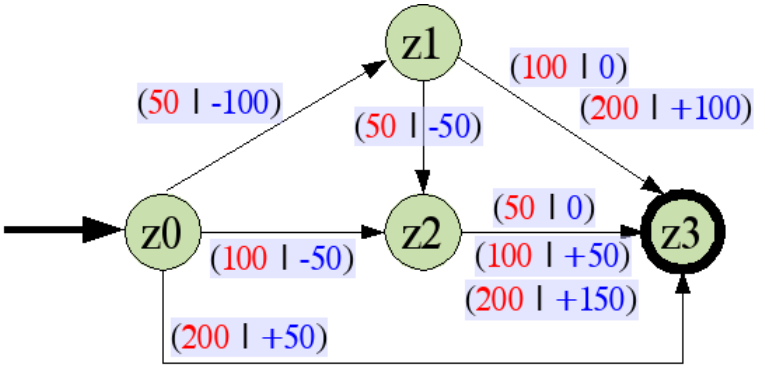
Übung 1: Kara läuft geradeaus und prüft, ob Kleeblätter und leere Felder abwechselnd aufeinander folgen. Sobald diese Reihenfolge nicht mehr gilt, bleibt Kara stehen. Trifft Kara auf einen Baum, so ist die Suche beendet.



Übung 2: Interpretiert man ein Feld mit Kleeblatt als ’1’, ein leeres Feld mit ’0’, dann kann man sich eine Folge von Kleeblättern als binäre Zahl vorstellen. Das Ende der „Zahl” wird durch einen Baumstumpf markiert. Lassen Sie Kara prüfen, ob eine solche „Kleeblattzahl” durch 4 teilbar ist. Im Erfolgsfall bewegt sich Kara um ein Feld nach unten.

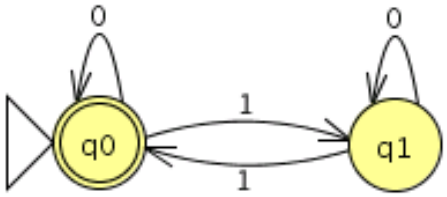
3 Lösungen

Übung 1: Parkautomat

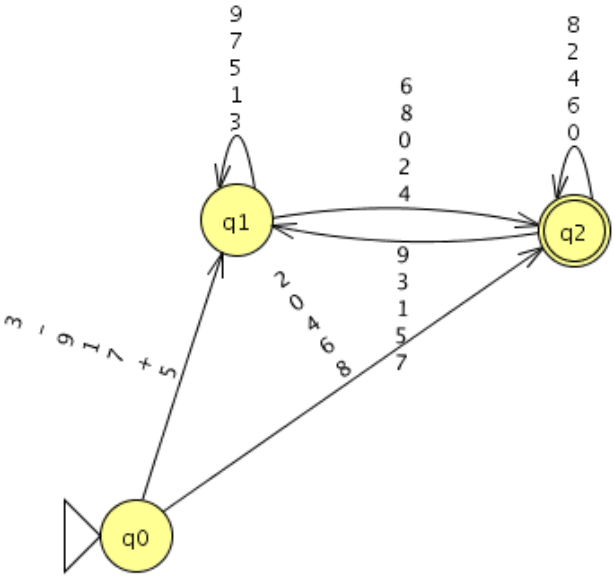


Übung 2: formale Beschreibung:

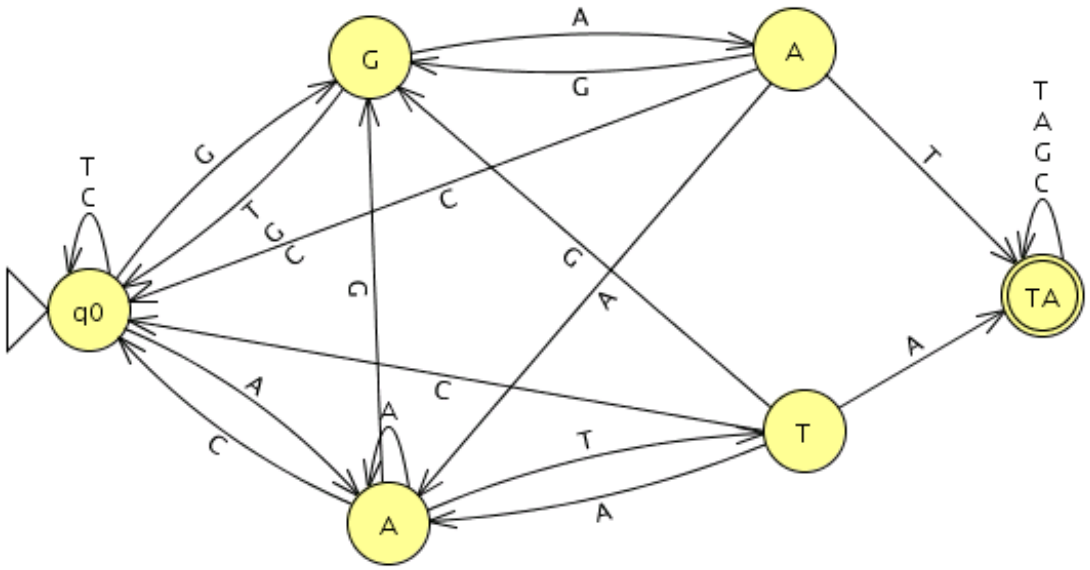
```
E  = {0, 1}
Z  = {q0, q1 }
z0 = q0
ZE = {q0}
```



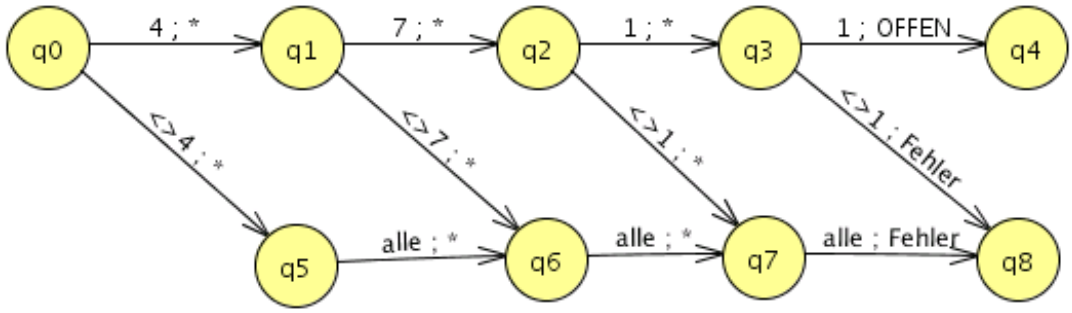
Aufgabe 1: Teilbarkeitsautomat



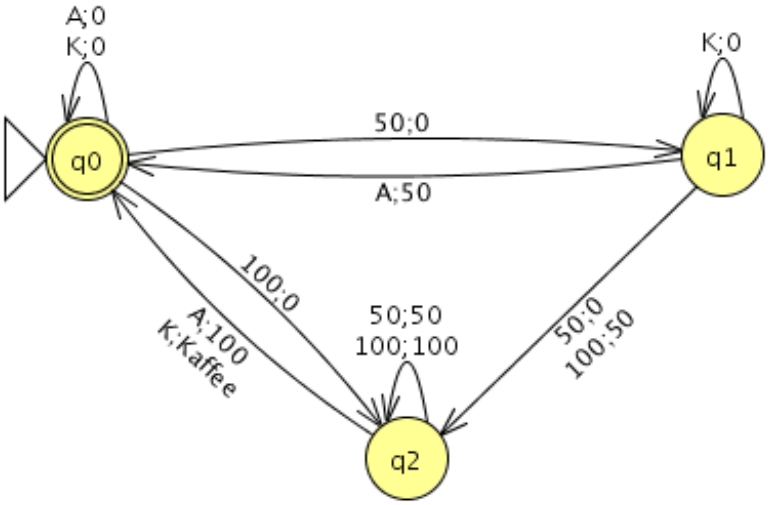
Aufgabe 2: Gen-Schnipsel



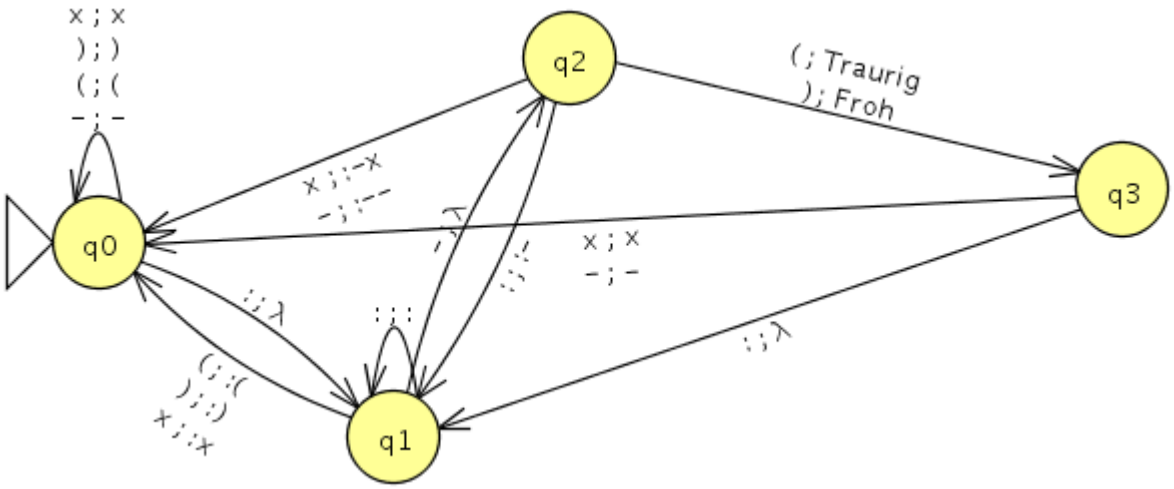
Aufgabe 3: Zahlenschloss „4711”



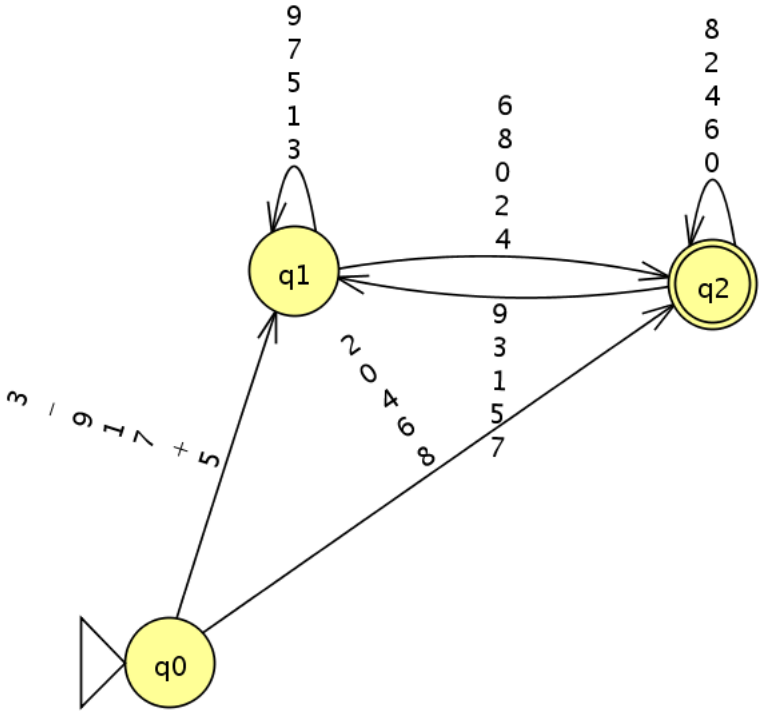
Aufgabe 4: Kaffeeautomat



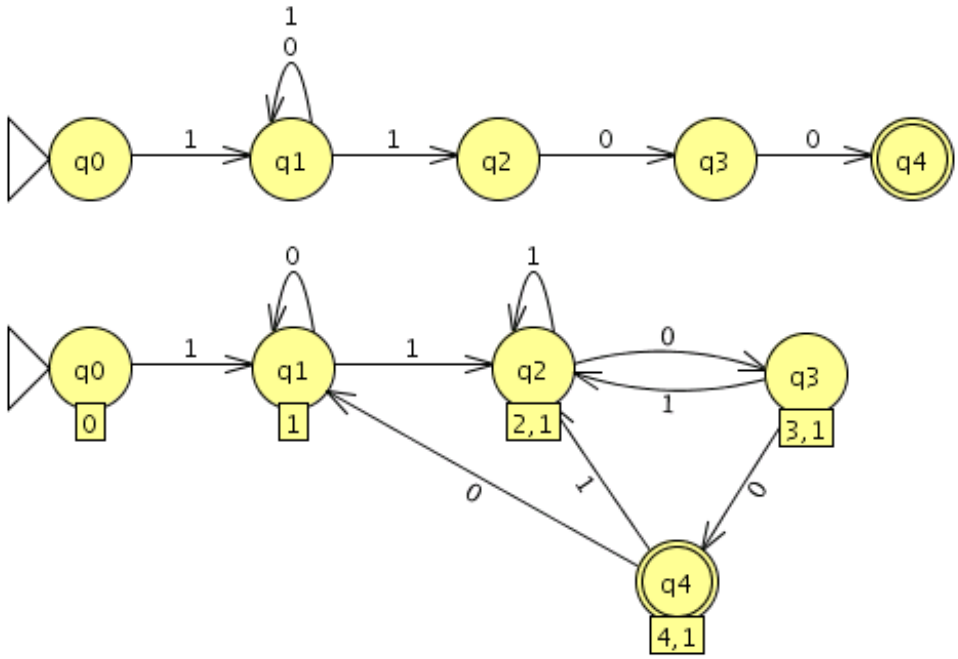
Aufgabe 5: Smiley



Aufgabe 6: Ganzzahl-Akzeptor



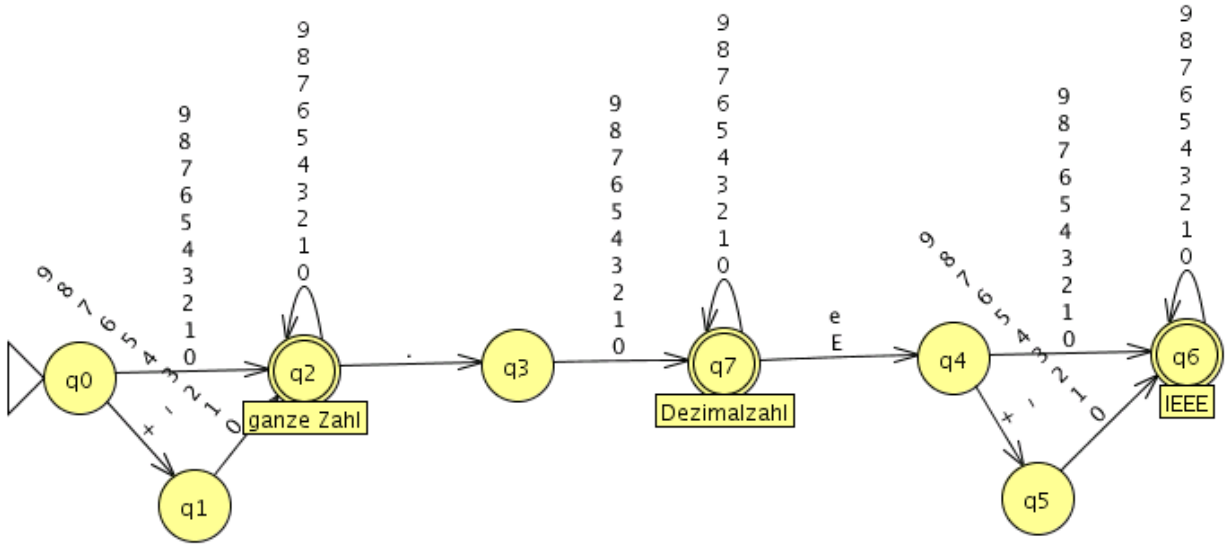
Aufgabe 7: NEA – DEA



Aufgabe 8: Beschreibung:

```
E  = {"A", "B", "E", "N", "R", "S",  "W" }
Z  = {q0, q1, q2, q3}
z0 = q0
ZE = {q3}
Beispiele:  WASSER, BESEN, BAER
```

Aufgabe 9: Reelle Zahlen



4 Literatur und Links

Literatur

[MOD] Modrow, E.: Theoretische Informatik mit Delphi. emu-online Scheden 2005.

Ein Schulbuch mit einer vollständigen Darstellung schulrelevanter Themen (Automaten, Turingmaschinen, Sprachen). Als Programmiersprache wird Delphi verwendet. (`www.emu-online.de`)

[REI] Reichert, Nievergelt, Hartmann: Programmieren mit Kara. Springer-Verlag, Berlin Heidelberg, 2004,2005 (ISBN 3-540-23819-0)

Die Autoren entwickelten das Kara-System an der ETH Zürich. „*Mit dem programmierbaren Marienkäfer Kara kann sich der Leser in spielerischer Weise grundlegende Algorithmen und komplexe Aufgaben der Informatik erarbeiten.*"

[ENG] Engelmann, L.: Informatik bis zum Abitur. Paetec, Berlin, 2002 (ISBN 3-89818-600-8)

Das Unterrichtsbuch enthält eine ausführliche Darstellung der theoretischen Informatik mit einigen Beispielaufgaben. Neben formalen Sprachen und Automaten werden die Berechenbar-keitstheorie behandelt und effiziente Algorithmen mit Komplexitätsbetrachtungen vorgestellt.

[HOF] Douglas R. Hofstadter: Gödel, Escher, Bach ein endloses geflochtenes Band. Ernst Klett Verlag, Stuttgart 1985 (ISBN 3423300175)

Ein absolutes Kultbuch, das in zentrale Aspekte der theoretischen Informatik einführt. Zu jedem Kapitel gibt es eine einführende Kurzgeschichte mit den Hauptfiguren Achilles und Theo Schildkröte.

Eine umfassendere Literaturliste finden Sie auf den Seiten des Berliner Bildungsservers.

Internet

http://www.bebis.de/themen/faecher/informatik

Unter Vertiefungsgebiete/theoretische Informatik finden Sie ein kommentierte Literatur- und Linklisten.

http://www.jflap.org/

Homepage des vorgestellten Javaprogramms zur Modellierung endlicher Automaten, Kellerautomaten, Sprachen, Turing- und Registermaschinen.

http://www.swisseduc.ch/informatik/karatojava/

Mit dem Kara-Modell können Automaten und Turingmaschinen direkt simuliert werden. Zusätzlich enthalten diese Seiten viele Unterrichtsmaterialien zum Thema.

http://www.oberstufeninformatik.de/theorie

Startseite von H. Gierhardt zur theoretischen Informatik mit zahlreiche Verweisen.

http://www.pns-berlin.de/lk-ti-automat.html

Erkennende Automaten und formale Sprachen werden hier unter dem Aspekt einer Implementierung in Haskell und/oder Java dargestellt. Die Informationen beruhen auf Unterrichtserfahrungen aus unterschiedlichen Informatikkursen an der Paul-Natorp-Oberschule.

http://www.pns-berlin.de/fortbildung.html

Auf diesen Seiten finden Sie Materialien zu unterschiedlichen Lehrerfortbildungen, die ich im Rahmen früherer IBBB-Tagungen oder LISUM - Veranstaltungen abgehalten habe.

Für Rückfragen, Anregungen und Kritik können Sie sich gerne an mich wenden. Die Materialien dieses Workshops und früherer Veranstaltungen finden Sie unter der angegebenen Internetadresse.

Walter Gussmann Paul-Natorp-Oberschule,
Internet: `http://www.pns-berlin.de`
Email: `wagu1@web.de`