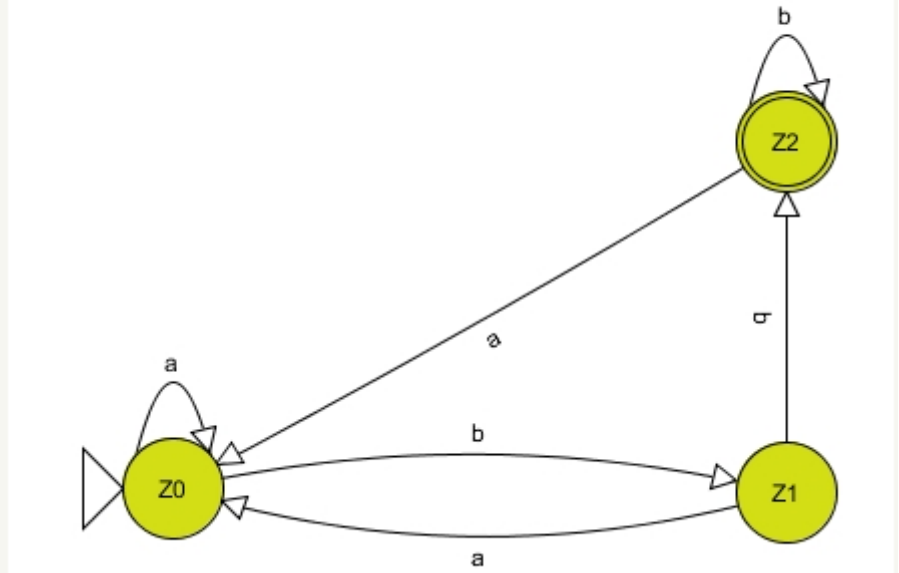
Endliche Automaten

Ein endlicher Automat ist ein spezielles Zustandsdiagramm mit endlich vielen Zuständen.

Für bestimmte formale Sprachen (den sogenannten **regulären Sprachen**) kann man mit einem endlichen Automaten prüfen, ob ein Wort zu dieser Sprache gehört.

Automat für Wörter einer Sprache über dem Alphabet A = {a, b} :



Der Startzustand Z0 ist mit einem Pfeil gekennzeichnet. Der Endzustand Z2 erhält eine doppelte Umrandung.

Der Automat arbeitet der Reihe nach die Buchstaben eines Wortes ab und wechselt abhängig vom Buchstaben den Zustand. Dies geschieht so lange, bis das Restwort leer ist. Ist der Automat dann in einem Endzustand (es muss genau einen Startzustand geben, kann aber mehrere Endzustände geben), gehört das Wort zu der Sprache.

Übung:

Gehört das Wort aababbabb zur Sprache?

Welche Wörter enthält die Sprache?

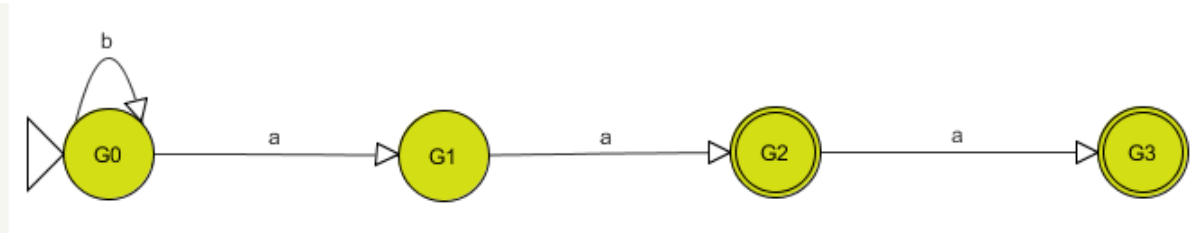
Formuliere eine Grammatik dieser Sprache!

Automaten mit Fehlerzuständen

Ein Automat soll alle Wörter über dem Alphabet A = {a,b} erkennen, die mit beliebig vielen Zeichen b beginnen und auf zwei oder drei a enden. Weitere Zeichen a oder b sollen in einem Wort nicht vorkommen.

In EBNF (vereinfachte Schreibweise ohne Hochkommas):

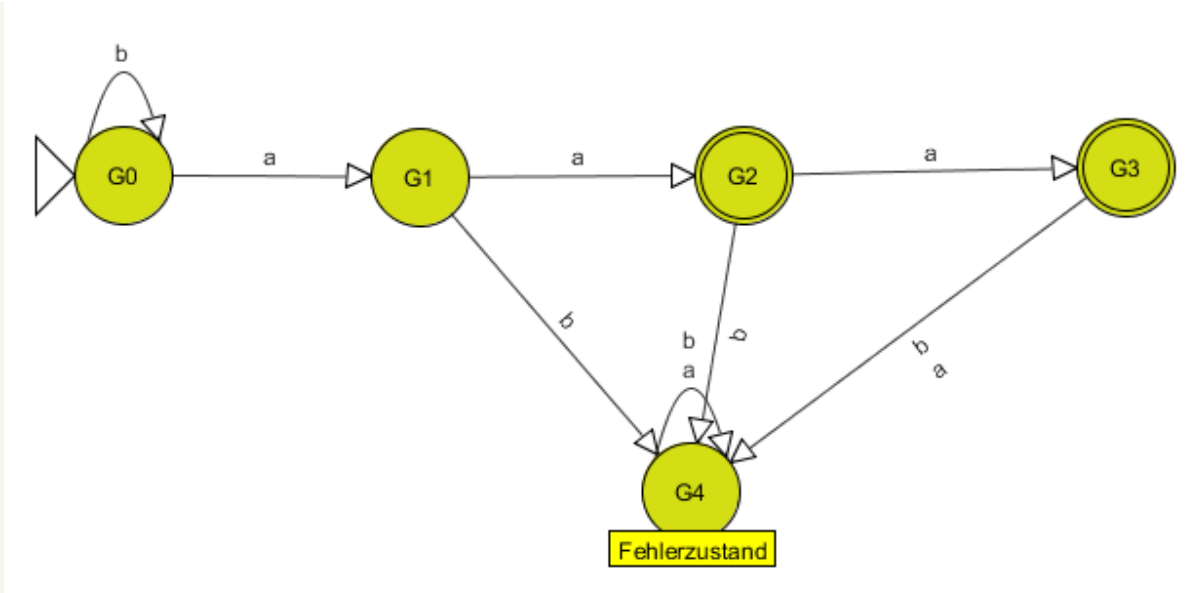
wort = {b}aa[a]



Prüft man mit diesem Automaten das Wort bbaaaa, so wird dieses nicht akzeptiert, weil sich der Automat vor dem letzten a im Endzustand G3 befindet, von dem aus kein Übergang für das Zeichen a existiert. Das Wort ist also nicht vollständig abgearbeitet.

Ebenso erkennt er das Wort bbbaab nicht, weil er sich nach dem zweiten a im Endzustand G2 befindet, von dem aus kein Übergang für das noch fehlende Zeichen b existiert.

Man kann in solchen Fällen einen sogenannten Fang- oder Fehlerzustand einführen:



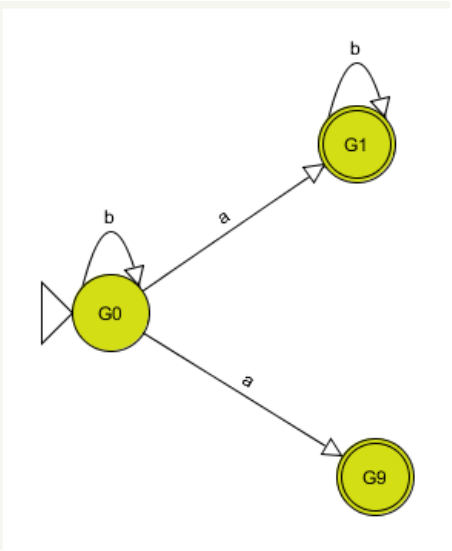
Wörter wie bbaaaa oder bbaabaaa werden ebenfalls nicht akzeptiert aber nun vollständig abgearbeitet.

Dieser Automat hat darüber hinaus die Eigenschaft, dass es zu jedem Zustand bei jedem Eingabezeichen einen Übergang gibt.

Der Automat heißt in so einem Fall **vollständig.**

Gibt es von jedem Zustand für ein bestimmtes Zeichen höchstens einen Übergang, heißt der Automat **deterministisch**.

Beispiel: für einen nichtdeterministischen Automat:



Jeder nichtdeterministische Automat kann durch Zusammenführen entsprechender Zustände in einen deterministischen Automat, der dieselbe Sprache akzeptiert, umgewandelt werden.

Ein vollständiger deterministischer endlicher Automat (DEA) ist festgelegt durch:

(1) eine endliche Menge Z von Zuständen Z = {Z0, Z1, Z2, ... }

(2) ein endliches Eingabealphabet A = {z1, z2, z3, ...}

(3) einen Startzustand S ($S \in Z$)

(4) eine Menge E von Endzuständen ($E \subseteq Z$)

(5) eine Übergangsfunktion f, die jedem möglichen Paar aus Zustand und Eingabezeichen einen Folgezustand zuordnet.

Die Übergänge kann man mit der Übergangsfunktion so schreiben:

f(Zi, zj) = Zk

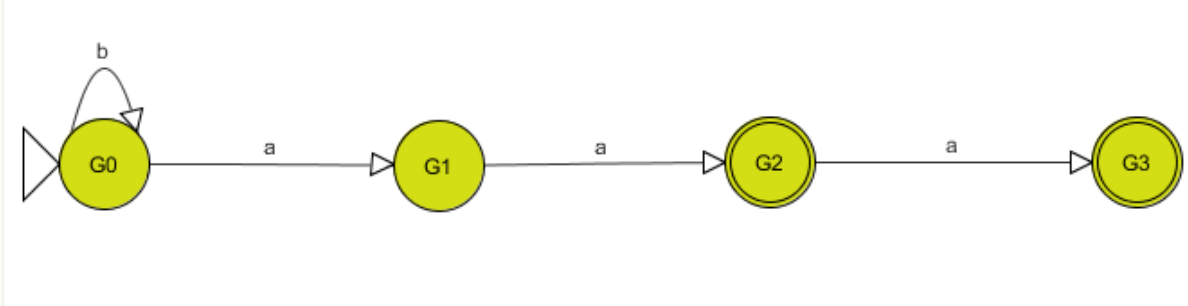
In dem Beispiel oben:

f(G0, a) = G1
f(G0, b) = G0
f(G1, a) = G2, usw.

Bestimmen der Grammatik aus einem endlichen Automaten

Zu jedem DEA lässt sich eine Grammatik ermitteln, welche genau die Sprache erzeugt, die vom Automaten akzeptiert wird.

Beispiel



A = {a,b}

V = {G0, G1, G2, G3}

S = G0

P:

R1 <G0> → b <G0>
R2 <G0> → a <G1>

R3 <G1> → a <G2>

R4 <G2> → a <G3>

R5 <G2> → ε (Leeres Wort)

R6 <G3> → ε

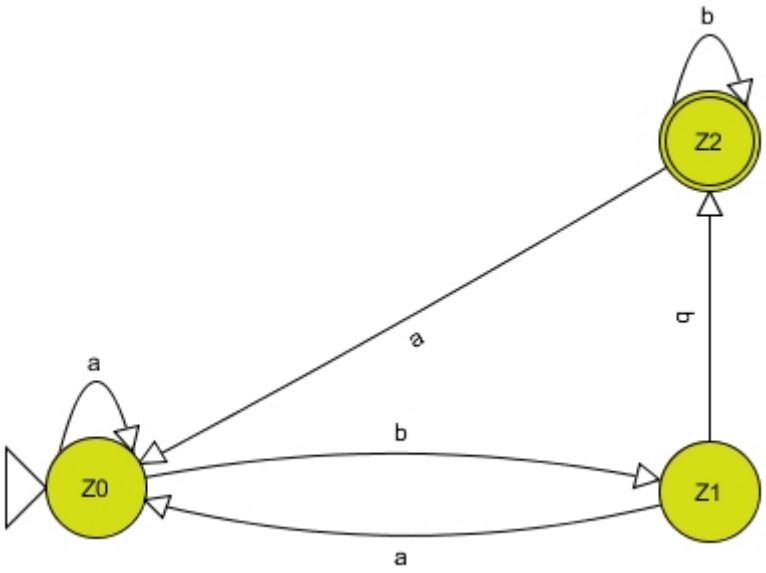
Ableitung von bbaaa:

<G0> → b <G0> → b b <G0> → b b a <G1> → b b a a <G2> → b b a a a <G3> → b b a a a

Natürlich ist es auch möglich, den Fehlerzustand mit in die Grammatik einzubauen.

Übung:

Erstelle eine Grammatik für die Sprache, die von folgendem Automaten akzeptiert wird:



Aus einem vollständigen deterministischen endlichen Automaten (DEA) kann eine Grammatik konstruiert werden, die genau die Sprache erzeugt, die der Automat akzeptiert:

(1) die Menge der Zustände wird zur Menge der Nichtterminalzeichen

(2) das Eingabealphabet wird zur Menge der Terminalzeichen

(3) der Startzustand wird zur Startvariablen

(4) der Übergang f (Zi, zj) = Zk wird zur Produktionsregel <Zi> → zj <Zk>

(5) jeder Endzustand Ze liefert die Produktionsregel <Ze> → ε

Ohne Verwendung von ε :

(5) Für alle Endzustände Ze ersetzt man <Zi> →zj <Ze> durch <Zi> →zj <Ze> | zj

Die Produktionsregeln haben alle die Form

<V> →t <V2> oder
<V> →t oder
<V> → ε

Die Sprachen, die durch solche Grammatiken beschrieben werden nennt man **reguläre Sprachen**.

In der theoretischen Informatik gibt es auch den Begriff **reguläre Ausdrücke.** Darunter versteht man einem der EBNF ähnlichen Formalismus zur Erzeugung von Sprachen.

Beispiel:

b (a | b)* a a beschreibt alle Wörter, die mit b beginnen und mit zwei a enden.

Die Menge der durch reguläre Ausdrücke beschreibbaren Sprachen ist genau die Menge der regulären Sprachen.

Ein wichtiger Satz aus der theoretischen Informatik lautet:

Eine Sprache ist genau dann regulär, wenn sie von einem endlichen Automaten akzeptiert wird.

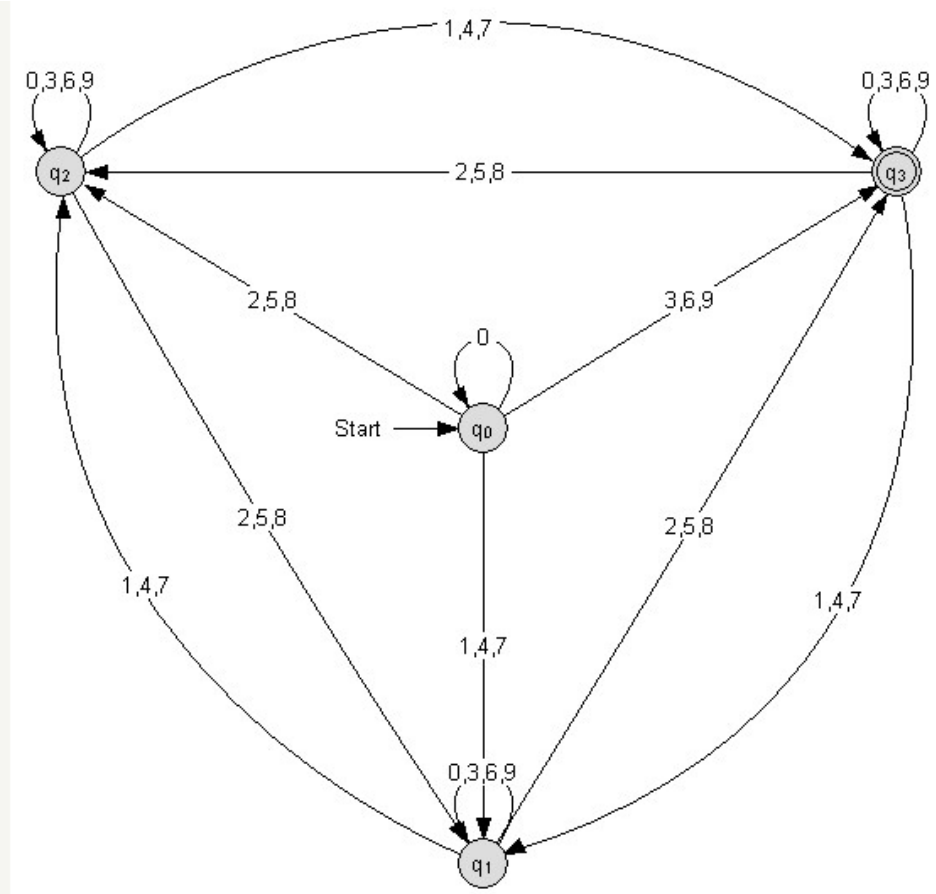
Im Gegensatz dazu kann nicht jede beliebige formale Sprache durch einen endlichen Automaten beschrieben werden.

Beispiel:

Die Sprache der korrekten arithmetischen Terme mit beliebig vielen verschachtelten Klammern.

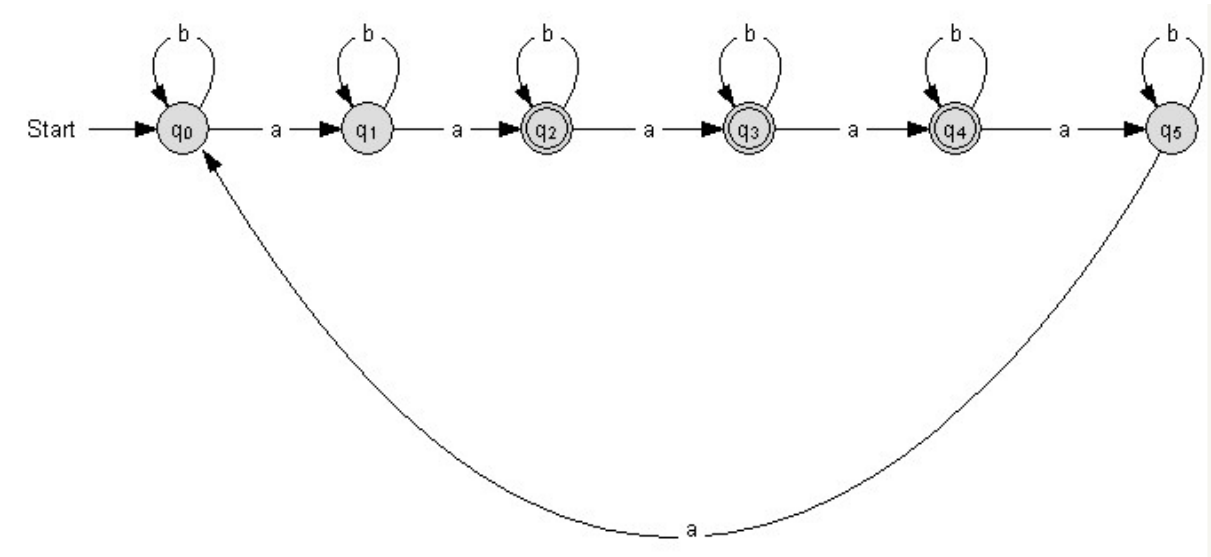
Die Grammatik dazu findest du im Buch im Kapitel 4*. Man kann zwar einen Automaten für eine bestimmte maximale Verschachtelungstiefe angeben, für den allgemeinen Fall findet man jedoch keinen Automaten.

Übung:



Gib eine formale Beschreibung des Automaten und eine dazu gehörige Grammatik an.

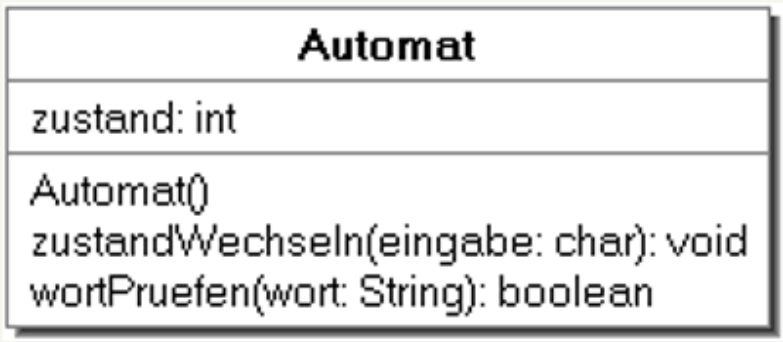
Bearbeite die Aufgaben auch für diesen Automaten:



Implementieren eines endlichen Automaten:

Ein DEA kann in Java sehr einfach mit Hilfe der Mehrfachauswahl implementiert werden.

Das folgende Klassendiagramm modelliert einen Automaten, dessen Sprache einfache Zeichen (char) als Alphabet besitzt.



Klasse Automat
Methode void zustandWechseln(char eingabe)

Mit einer Mehrfachauswahl werden die möglichen Werte für das Attribut zustand erfasst.

Für jeden möglichen Wert von zustand gibt es dann wieder eine Mehrfachauswahl, die die möglichen Werte des zu prüfenden Zeichen erfasst.

```
wenn zustand = 0, dann
   wenn eingabe = 'a', dann zustand = neuer Wert
   wenn eingabe = 'b', dann zustand = neuer Wert
   usw.
wenn zustand = 1, dann
   wenn eingabe = 'a', dann zustand = neuer Wert
   wenn eingabe = 'b', dann zustand = neuer Wert
   usw.
usw.
```

Klasse Automat
Methode boolean wortPruefen(String wort)

Die Zeichenkette wird beginnend mit dem ersten Zeichen abgearbeitet es wird jeweils die Methode zustandWechseln für das aktuelle Zeichen aufgerufen.
Ist das Wort vollständig abgearbeitet, wird geprüft, ob der erreichte Zustand akzeptierend ist und entsprechend true oder false zurückgegeben.

Akzeptor

Definition: Akzeptor

Ein erkennender, endlicher Automat (Akzeptor) ist ein 5-Tupel A: = (Z,X,δ,z_o,Z_E), wobei gilt:

- X ist eine nichtleere, endliche Menge, das **Eingabealphabet**, wobei gilt: $x \in X$
- Z ist eine nichtleere, endliche Menge, die **Zustandsmenge**, wobei gilt: $z \in Z$
- δ ist die **Überführungsfunktion**, welche jedem Paar (Eingabezeichen, Zustand) einen Folgezustand zuordnet: $\delta : X \times Z \rightarrow z$
- $z_0 \in Z$ ist der **Anfangszustand**
- $Z_E \subseteq Z$ ist eine Teilmenge von Z, die **Endzustandsmenge**

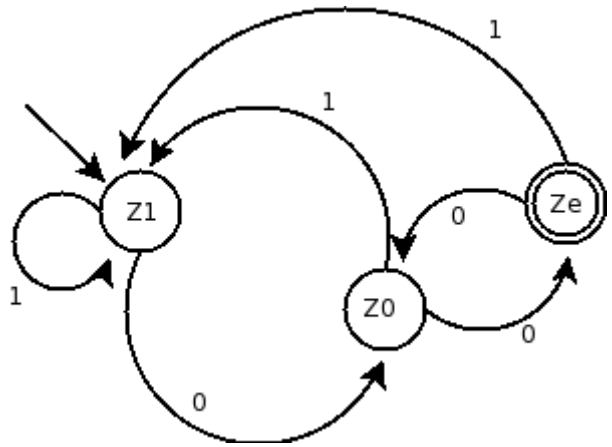
Funktionsweise und Erklärung

Die Unterschiede zwischen Akzeptor und Transduktor liegen darin, dass ein Akzeptor keine Ausgaben besitzt, aber dafür ein oder mehrere Endzustände. Ein Akzeptor akzeptiert und erkennt ein Eingabewort und signalisiert durch seinen Endzustand das Ergebnis nach außen.

Darüber hinaus sind Endzustände nicht als eine Kombination von Sackgasse und Einbahnstraße zu verstehen. Ist der Automat nach dem kompletten Einlesen des Eingabewortes in einem Endzustand, so akzeptiert er. Ist er nicht in einem Endzustand bei Ende des Wortes, so verwirft er es. Sollte ein Wort noch nicht zu Ende sein und ein Automat erreicht einen Endzustand, wird dieser als "normaler" Zustand behandelt.

Beispiel

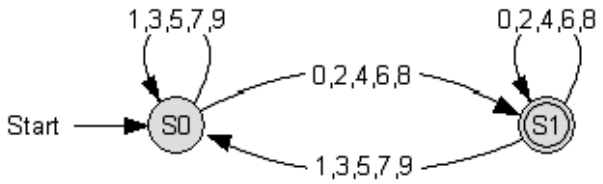
Als einfaches Beispiel stelle man sich vor, dass ein Akzeptor überprüfen soll, ob ein Eingabewort aus Nullen und Einsen mit einer geradzahligen Anzahl von Nullen endet.



Endliche Automaten: Akzeptoren und Transduktoren

Akzeptor

Der folgende Automat besitzt keine Ausgaben, dafür aber einen besonders gekennzeichneten sog. ***Endzustand***.



Wenn man hier eine gerade, natürliche Dezimalzahl eingibt, landet man im Endzustand S1. Das Problem der führenden Nullen ist hier unwichtig.

Endzustände erkennt man an einem Doppelkreis.

Die Eingaben 1434 und 20010 enden beide mit dem Endzustand. Man sagt, diese Zahlen werden vom Automaten **akzeptiert** und den zugehörigen Automaten nennt man ***Akzeptor***.

Die Menge aller Eingabefolgen, die von einem Akzeptor akzeptiert werden, nennt man die ***Sprache L(A)*** des Automaten. Dabei steht ***L*** für *language* und ***A*** für Akzeptor.

Merke:

- ✓ Automaten ohne Ausgabe nennt man **Akzeptoren**
- ✓ Automaten mit Ausgabe heißen **Transduktoren**

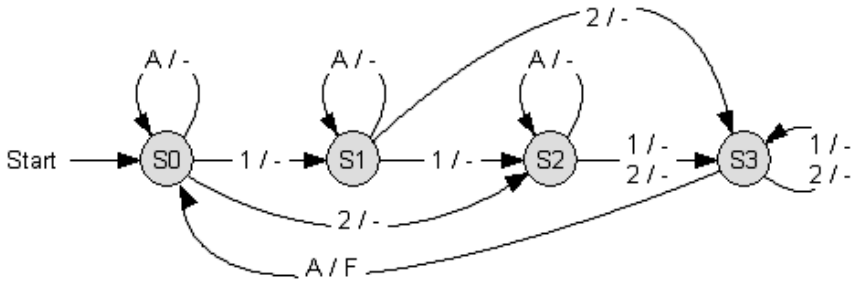
Transduktor

Ein Fahrscheinautomat verkauft Fahrkarten im Wert von 3€. Der Automat sei sehr einfach konstruiert. Es muss exakt der Wert von 3€ eingeworfen werden, denn zu viel eingeworfenes Geld wird einbehalten. Man kann nur 1€- oder 2€-Münzen einwerfen. Nach Drücken des Ausgabeknopfes erhält man den Fahrschein.

Eingabealphabet E = {1, 2, A} entsprechend den Münzen bzw. dem Ausgabeknopf.
Ausgabealphabet A = {F, -} entspricht der Fahrkarte bzw. der leeren Ausgabe. Die leere Ausgabe wird oft auch mit dem Leerstring ‘‘ oder mit dem Buchstaben n *(nothing)* gekennzeichnet.

Es gibt vier innere ***Zustände*** S_0, S_1, S_2, S_3 des Automaten:
S_0: ***Anfangszustand***,
S_1: 1€ wurde bereits eingeworfen,
S_2: 2€ wurden bereits eingeworfen,
S_3: 3€ oder mehr wurden eingeworfen. Betätigen der Ausgabetaste wird erwartet.

Zustandsmenge S = {S_0, S_1, S_2, S_3}



Jeder Zustand wird benannt, durch einen Kreis symbolisiert und bildet so einen Knoten des ***Zustandsdiagramms***. Der Anfangszustand wird durch einen zusätzlichen Eingangspfeil gekennzeichnet.

Von jedem Zustand aus führen Pfeile (die sog. Kanten) zu den möglichen Folgezuständen. Diese Kanten werden mit den Eingabezeichen beschriftet, die den entsprechenden Übergang auslösen. Falls mehrere Eingabezeichen das System in denselben Folgezustand überführen, werden diese Zeichen oft mit dem Oder-Zeichen ∨ verknüpft.

Produziert der Automat auch Ausgabezeichen, so müssen diese ebenfalls an der Kante, durch „/“ von den Eingabezeichen getrennt, angezeigt werden.

Automaten-Definitionen

- Ein Automat heißt ***endlich***, wenn das Eingabealphabet E, das Ausgabealphabet A und die Zustandsmenge S (nichtleer und) endlich sind.

- Ein Automat heißt ***deterministisch***, wenn es für jede Kombination von Eingabezeichen und Zustand nur eine mögliche Kombination von Ausgabezeichen und Folgezustand gibt.

- Ein ***Mealy-Automat*** ist ein Transduktor, dessen Ausgabe von seinem Zustand **und** seiner Eingabe abhängt. Anschaulich bedeutet das, dass jeder Kante im Zustandsdiagramm ein Ausgabewert zugeordnet wird. Der Name geht auf *George H. Mealy (1927 – 2010)* zurück.

- Ein ***Moore-Automat*** ist ein Transduktor, dessen Ausgabe nur von seinem Zustand abhängt. Anschaulich bedeutet das, dass im Zustandsdiagramm an den Kanten keine Ausgabewerte stehen. Der Name geht auf *Edward F. Moore (1925-2003)* zurück.

Unter einem ***Mealy-Automaten*** versteht man ein 6-Tupel
$M = (E, A, S, s_0, u, g)$, welches beschrieben wird durch:

$E = \{e_1, e_2, \ldots, e_r\}$	das endliche Eingabealphabet
$A = \{a_1, a_2, \ldots, a_m\}$	das endliche Ausgabealphabet
$S = \{s_1, s_2, \ldots, s_n\}$	die endliche Zustandsmenge
$s_0 \in S$	den Anfangszustand
$u: E \times S \rightarrow S$	die Übergangsfunktion
$g: E \times S \rightarrow A^*$	die Ausgabefunktion

Dabei versteht man unter A^* die Menge aller Worte, die mit dem Ausgangsalphabet A erzeugt werden können, einschließlich des leeren Wortes.

Alle diese Angaben sind schon im Zustandsdiagramm enthalten und müssen eigentlich nicht mehr extra herausgeschrieben werden. Wir wollen deshalb diese Formalitäten nicht allzu wichtig nehmen. Allerdings sollten sie z.B. für das Zentralabitur bekannt sein.

Unter einem ***Moore-Automaten*** versteht man ein 6-Tupel
$M = (E, A, S, s_0, u, g)$, welches beschrieben wird durch:

$E = \{e_1, e_2, \ldots, e_r\}$	das endliche Eingabealphabet
$A = \{a_1, a_2, \ldots, a_m\}$	das endliche Ausgabealphabet
$S = \{s_1, s_2, \ldots, s_n\}$	die endliche Zustandsmenge
$s_0 \in S$	den Anfangszustand
$u: E \times S \rightarrow S$	die Übergangsfunktion
$g: S \rightarrow A^*$	die Ausgabefunktion

Unter einem ***Akzeptor-Automaten*** versteht man ein 5-Tupel
$M = (E, S, s_0, u, F)$, welches beschrieben wird durch:

$E = \{e_1, e_2, \ldots, e_r\}$	das endliche Eingabealphabet
$S = \{s_1, s_2, \ldots, s_n\}$	die endliche Zustandsmenge
$s_0 \in S$	den Anfangszustand
$u: E \times S \rightarrow S$	die Übergangsfunktion
$F \subset S$	eine Menge von Endzuständen