

Kapitel 2

Das Modell der endlichen Automaten

Wenn man ein Berechnungsmodell beschreiben will, muss man folgende Fragen beantworten (siehe auch das Modul „Geschichte der Informatik und Begriffsbildung“ in den Büchern [2, 3]):

1. Welche elementaren Operationen stehen zur Verfügung, aus denen man die Programme zusammenstellen kann?
2. Wie viel Speicher steht zur Verfügung und wie geht man mit dem Speicher um?
3. Wie wird die Eingabe eingegeben?
4. Wie wird die Ausgabe bestimmt und ausgegeben?

Bei endlichen Automaten hat man keinen Speicher zur Verfügung. Man hat lediglich den Speicher, in dem das Programm gespeichert wird, und den Zeiger, der auf die aktuell ausgeführte Zeile des Programms zeigt. Somit darf das Programm keine Variablen benutzen. Das mag überraschend sein, weil man sich fragt, wie man ohne Variablen überhaupt rechnen kann. Der Grundgedanke dabei ist, dass der Inhalt des Zeigers, also die Nummer der aktuellen Programmzeile, die einzige wechselnde Information ist. Mit dieser Pseudovariablen muss man daher auskommen.

Hinweis für die Lehrperson Wenn die Schülerinnen und Schüler den Befehl `goto` noch nicht kennen, muss man ihn zuerst vorstellen. Der Befehl `goto i` entspricht genau dem Befehl `JUMP i`, den wir im Modul „Geschichte der Informatik und Begriffsbildung“ in [2, 3] vorgestellt haben.

Wenn $\Sigma = \{a_1, a_2, \dots, a_k\}$ das Alphabet ist, über dem die Eingaben dargestellt sind, dann darf der endliche Automat nur den folgenden Operationstyp benutzen:

```

select input =  $a_1$  goto  $i_1$ 
      input =  $a_2$  goto  $i_2$ 
      ⋮
      input =  $a_k$  goto  $i_k$ 

```

Die Bedeutung dieser Operation (dieses Befehls) ist, das nächste Eingabesymbol zu lesen und mit $a_1, a_2, \dots, a_k$ zu vergleichen. Wenn das Eingabesymbol gleich a_j ist, setzt das Programm die Arbeit in der Zeile i_j fort. Die Anweisung

goto l

bedeutet wörtlich „Gehe in die Zeile l und setze dort die Arbeit fort“. Die Umsetzung dieses Befehls bedeutet automatisch die Löschung des gelesenen Symbols und das Weiterlesen in der Zeile i_j beim nächsten Symbol. Jede Zeile des Programms enthält genau einen Befehl in der oben angegebenen Form. Wir nummerieren die Zeilen mit natürlichen Zahlen $0, 1, 2, 3, \dots$ und die Arbeit des Programms beginnt immer in der Zeile 0.

Aufgabe 2.1 Wir betrachten das Programm:

```

0:      select input =  $a$  goto 1
          input =  $b$  goto 1
          input =  $c$  goto 0
1:      select input =  $a$  goto 0
          input =  $b$  goto 0
          input =  $c$  goto 1

```

In welche Zeile geht das Programm, wenn es in der 0-ten Zeile das Symbol c gelesen hat? Wohin geht es, wenn es in der Zeile 1 das Symbol b liest?

Wenn Σ nur aus zwei Symbolen (zum Beispiel 1 und 0) besteht, kann man statt des Befehls **select** den folgenden Befehl **if ... then ... else** verwenden:

if input = 1 then goto i else goto j .

Solche Programme benutzt man, um Entscheidungsprobleme zu lösen. Die Antwort ist durch die Zeilennummer bestimmt, in der sich das Programm befindet, nachdem es die gesamte Eingabe gelesen hat. Wenn ein Programm aus m Zeilen besteht, wählt man eine Teilmenge F von $\{0, 1, 2, \dots, m-1\}$. Wenn nach dem Lesen der gesamten Eingabe das Programm in der j -ten Zeile endet und $j \in F$, dann akzeptiert das Programm die Eingabe. Wenn $j \in \{0, 1, 2, \dots, m-1\} - F$, dann akzeptiert das Programm die Eingabe nicht. Die Menge aller akzeptierten Wörter ist die von dem Programm akzeptierte (erkannte) Sprache.

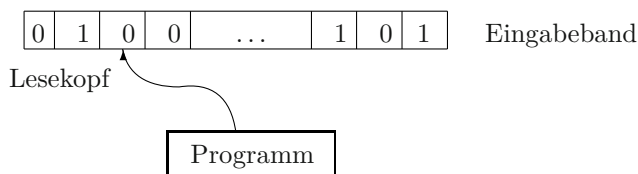


Abbildung 2.1 Schematischer Aufbau eines endlichen Automaten

Betrachten wir als Beispiel folgendes Programm A , das Eingaben über dem Alphabet Σ_{Bool} bearbeitet.

```

0:      if input = 1 then goto 1 else goto 2
1:      if input = 1 then goto 0 else goto 3
2:      if input = 0 then goto 0 else goto 3
3:      if input = 0 then goto 1 else goto 2

```

Wählen wir $F = \{0, 3\}$. Das Programm A arbeitet auf einer Eingabe 1011 wie folgt: Es startet in der Zeile 0 und geht in die Zeile 1, nachdem es eine 1 gelesen hat. Es liest eine 0 in der ersten Zeile und geht in die dritte Zeile. In der dritten Zeile liest es eine 1 und geht in die zweite Zeile, um in die dritte Zeile nach dem Lesen einer weiteren 1 zurückzukehren. Die Berechnung ist beendet, und weil $3 \in F$ gilt, wird das Wort 1011 akzeptiert. Somit gehört 1011 zu der von A akzeptierten Sprache.

Aufgabe 2.2 Wir betrachten das Programm aus Aufgabe 2.1. Sei $F = \{0\}$. Welche der Wörter $a, c, ccc, acbca, aabbcc, accbac$ werden akzeptiert? Kannst du eine unendliche Menge von Wörtern bestimmen, die dieses Programm mit Sicherheit nicht akzeptieren kann? Findest du auch eine unendliche Menge von Wörtern, die dieses Programm akzeptiert?

Mit endlichen Automaten verbindet man oft die schematische Darstellung aus Abbildung 2.1. In dieser Abbildung sehen wir die drei Hauptkomponenten des Modells: ein gespeichertes **Programm**, ein **Band** mit dem Eingabewort und einen **Lesekopf**, der sich auf dem Band nur von links nach rechts bewegen kann.¹ Das Band (auch Eingabeband genannt) betrachtet man als einen linearen Speicher für die Eingabe. Das Band besteht aus Feldern (Zellen). Ein Feld ist eine elementare Speichereinheit, die ein Symbol aus dem betrachteten Alphabet enthalten kann.

¹Die komponentenartige Darstellung von allgemeinen Berechnungsmodellen enthält außerdem noch einen Speicher, Schreib- und Lesezugriffsmöglichkeiten auf diesen Speicher und eventuell ein Ausgabemedium.

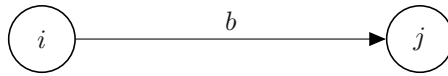


Abbildung 2.2 Graphische Darstellung eines Übergangs von Zeile i zu Zeile j

Ein Arbeitsschritt des endlichen Automaten (des Programms) besteht aus der Ausführung des Befehls der Zeile, in der sich das Programm befindet. Der Automat liest das Symbol in dem Feld, auf dem sich sein Lesekopf befindet. Abhängig vom Symbol geht er in die nächste Programmzeile, die bearbeitet werden soll. Der Lesekopf rückt dabei automatisch um ein Feld nach rechts.

Die oben beschriebene Klasse von Programmen benutzt man heute fast gar nicht mehr, um endliche Automaten zu beschreiben, weil diese Programme wegen des `goto`-Befehls keine schöne Struktur haben. Daher ist diese Modellierungsart nicht sehr anschaulich und für die meisten Zwecke auch sehr unpraktisch. Die Idee einer umgangsfreundlicheren Definition endlicher Automaten basiert auf folgender visueller Darstellung unseres Programms. Wir ordnen jedem Programm A einen gerichteten, markierten Graphen $G(A)$ zu. $G(A)$ hat genauso viele Knoten wie das Programm A Zeilen hat. Jeder Zeile von A ist genau ein Knoten zugeordnet, der durch die Nummer der Zeile markiert ist. Die Knoten werden als kleine Kreise dargestellt. Der Name des Knotens wird in den Kreis geschrieben. Falls das Programm A aus einer Zeile i in die Zeile j beim Lesen eines Symbols b übergeht, dann enthält $G(A)$ eine gerichtete Kante (i, j) mit der Markierung b (Abbildung 2.2).

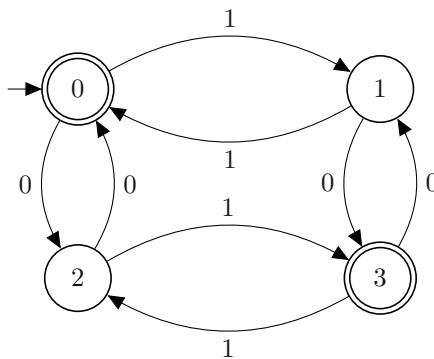


Abbildung 2.3 Graphische Darstellung des Programms A als Automat

Weil unsere Programme ohne Variablen für jedes $a \in \Sigma$ in jeder Zeile einen **select**-Befehl haben², hat jeder Knoten von $G(A)$ genau den Ausgangsgrad³ $|\Sigma|$.

Abbildung 2.3 enthält den Graphen $G(A)$ für das oben beschriebene vierzeilige Programm A . Die Zeilen aus F sind durch Doppelkreise als besondere Knoten von $G(A)$ gekennzeichnet. Der Knoten, der der Zeile 0 entspricht, wird durch einen zusätzlichen Pfeil (Abbildung 2.3) als Anfangsknoten aller Berechnungen gekennzeichnet.

Aufgabe 2.3 Stelle den endlichen Automaten über $\{a, b, c\}$, der durch folgendes Programm angegeben wird, graphisch dar.

```

0:      select input = a goto 1
        input = b goto 1
        input = c goto 0
1:      select input = a goto 2
        input = b goto 2
        input = c goto 1
2:      select input = a goto 2
        input = b goto 2
        input = c goto 2

```

Sei $F = \{2\}$. Welche der Wörter *acc*, *abba*, *ccbac*, *cbcbc*, *caccc*, *cacbc* werden akzeptiert? Kannst du die Sprache bestimmen, die dieser endliche Automat akzeptiert?

Aufgabe 2.4 Schreibe das Programm, das dem Automaten über $\{0, 1\}$ in Abbildung 2.4 entspricht. Bestimme die vier kürzesten Wörter, die er akzeptiert.

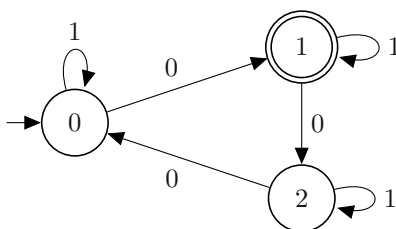


Abbildung 2.4 Der Automat aus Aufgabe 2.4

²Jede Zeile ist ein **select**-Befehl über alle Symbole des Alphabets.

³Der Ausgangsgrad eines Knotens ist die Anzahl der gerichteten Kanten, die den Knoten verlassen.

Beispiel 2.1 Wir betrachten die folgende Aufgabe, die kein Entscheidungsproblem darstellt. Wir wollen einen einfachen Obstautomaten entwerfen, der abgepackte Äpfel, Birnen und Orangen einzeln verkauft. Jedes Stück kostet 1 Euro, und man darf nur mit Münzen der Größen 50 Cent und 1 Euro bezahlen. Wir fordern, dass man den genauen Wert von 1 Euro einwerfen und danach einen der drei Knöpfe für Apfel, Birne oder Orange drücken muss. Wenn dies passiert, akzeptiert der Obstautomat die Folge von Signalen und gibt das gewünschte Obststück aus.

Die Signale von außen sind für den Obstautomaten Münzeinwürfe und das Drücken des Auswahlknopfes. Die Münzen stellen wir durch die Symbole $M_{0.50}$ und $M_{1.00}$ dar. Für die Auswahlknöpfe führen wir die Bezeichnungen A, B und O ein. Somit ist $\Sigma = \{M_{0.50}, M_{1.00}, A, B, O\}$ das benötigte Alphabet. Ein Automat, der diese Aufgabe erfüllt, ist in Abbildung 2.5 dargestellt.

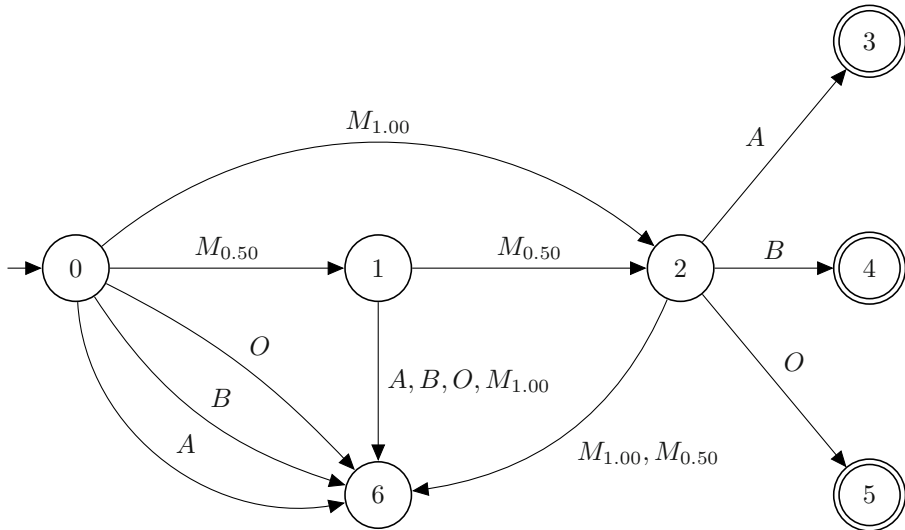


Abbildung 2.5 Graphische Darstellung des Obstautomaten

Wenn dieser Automat den Knoten 2 erreicht hat, wurde genau der Betrag von 1 Euro bezahlt. Durch die Wahl des entsprechenden Knopfes A , B oder C gelangt man dann zu einem der Knoten 3, 4 oder 5, in denen das gewünschte Objekt ausgegeben wird. Wenn man in den Knoten 0 oder 1 eine der Wahltasten A , B oder O betätigt, geht man in den Knoten 6 über. Weil die Summe nicht stimmt, wird hierbei das eingezahlte Geld zurückgegeben und der Vorgang abgebrochen. Der Verlauf ist ähnlich, wenn man im Knoten 1 und 2 versucht, mehr Geld als benötigt einzuwerfen.

Wie man in Abbildung 2.5 sieht, wird das Zeichnen von zu vielen Kanten vermieden, indem man zulässt, die Situation aus Abbildung 2.6 durch Abbildung 2.7 darzustellen. Man darf also alle gerichteten Kanten aus einem Knoten p in einen anderen Knoten q durch eine Kante darstellen. Dazu schreibt man die entsprechenden Symbole all dieser Kanten auf eine einzige Kante.

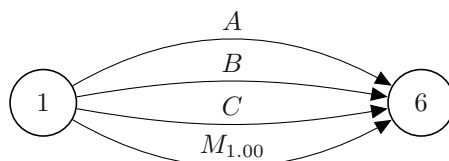


Abbildung 2.6 Mehrfache Kanten in einem Automaten



Abbildung 2.7 Darstellung von mehrfachen Kanten durch eine Kante mit Mehrfachbeschriftung

So wie der Automat entworfen wurde, funktioniert er nur für einen einzigen Versuch. Wir können ihn verbessern, indem ein zusätzliches Symbol S eingeführt wird. Dieses soll automatisch in den Knoten 3, 4, 5, und 6 auftreten, sobald Obst oder eingeworfenes Geld ausgegeben wurde. Visuell kann man dann Pfeile mit der Bezeichnung S aus den Knoten 3, 4, 5, und 6 in den Knoten 0 einfügen. Dadurch ist der Automat im Knoten 0 zu einem neuen Verkaufsversuch bereit. $\diamond$

Aufgabe 2.5 Entwirf einen Verkaufsautomaten, der drei unterschiedliche Produkte verkauft. Mindestens zwei Produkte sollen unterschiedliche Preise haben. Der Verkaufsautomat muss so gesteuert werden, dass man zuerst die gewünschte Ware wählen muss und erst danach korrekt einbezahlt wird.

Hinweis für die Lehrperson Die folgende mathematische Beschreibung eines endlichen Automaten kann man umgehen oder geschickt erleichtern. Im Prinzip reicht es aus, die Notation der Übergangsfunktion $\delta(q, a) = p$ einzuführen. Für das minimale Programm dieses Moduls, das sich auf den Automatenentwurf beschränkt, wird nicht mehr gebraucht.

Aus dieser graphischen Darstellung entwickeln wir jetzt die standardisierte formale Definition von endlichen Automaten. Die graphische Darstellung werden

wir aber weiterhin verwenden, weil sie eine sehr anschauliche Beschreibung von endlichen Automaten bietet. Die folgende formale Definition ist wiederum besser für das Studium der Eigenschaften endlicher Automaten und für die formale Beweisführung geeignet. Hierfür ändern wir teilweise die Terminologie. Was bisher als Zeile des Programms oder als Knoten des Graphen bezeichnet wurde, wird im Weiteren als **Zustand** des endlichen Automaten bezeichnet. Die Kanten des Graphen, die den **goto**-Befehlen des Programms entsprechen, werden durch die sogenannte **Übergangsfunktion** beschrieben.

Man beachte, dass der folgenden Definition ein allgemeines Schema zugrunde liegt, das man bei der Definition aller Rechnermodelle anwenden kann. Zuerst definiert man eine Struktur, welche die exakte Beschreibung jedes Objekts der Modellklasse ermöglicht.

Dann beschreibt man die Bedeutung (Semantik) dieser Struktur. Dies geschieht in folgender Reihenfolge: Zuerst definiert man den Begriff der Konfiguration. Eine **Konfiguration** ist die vollständige Beschreibung einer Situation (eines allgemeinen Zustands), in der sich das Objekt der Modellklasse (also z. B. der endliche Automat) befindet. Dann definiert man einen **Schritt** als einen Übergang aus einer Konfiguration in eine andere Konfiguration, wobei dieser Übergang durch eine elementare Aktion (Durchführung eines Befehls) des Rechnermodells realisierbar sein muss. Eine **Berechnung** kann dann als eine Folge solcher Schritte angesehen werden. Wenn man eine Berechnung definiert hat, kann man jeder Eingabe das Arbeitsergebnis des Objekts (z. B. des Automaten) als Ausgabe zuordnen.

Definition 2.1 *Ein endlicher Automat (EA) ist ein Quintupel $M = (Q, \Sigma, \delta, q_0, F)$, wobei*

- (i) Q eine endliche Menge von **Zuständen** ist,
 $\left\{ \begin{array}{l} \text{vorher: Die Menge der Programmzeilen oder die Menge der} \\ \text{Knoten in graphischer Darstellung.} \end{array} \right\}$
- (ii) Σ ein Alphabet, genannt **Eingabealphabet**, ist,
 $\{ \text{Alle zulässigen Eingaben müssen Wörter über } \Sigma \text{ sein.} \}$
- (iii) $q_0 \in Q$ der **Anfangszustand** ist,
 $\left\{ \begin{array}{l} \text{vorher: Die Zeile 0 des Programms ohne Variablen, aus der} \\ \text{die Berechnung immer startet.} \end{array} \right\}$
- (iv) $F \subseteq Q$ die **Menge der akzeptierenden Zustände**⁴ ist, und

⁴In der deutschsprachigen Literatur auch Endzustände genannt. Der Begriff „Endzustand“ kann aber auch zu Missverständnissen führen, weil die Berechnungen in einem beliebigen Zustand enden können. Außerdem entspricht der Begriff „akzeptierender Zustand“ der wahren

(v) δ eine Funktion von $Q \times \Sigma$ nach Q ist, die **Übergangsfunktion** genannt wird.

$\left\{ \begin{array}{l} \delta(q, a) = p \text{ bedeutet, dass } M \text{ in den Zustand } p \text{ übergeht, falls } M \\ \text{im Zustand } q \text{ das Symbol } a \text{ gelesen hat (Abbildung 2.9).} \end{array} \right\}$

Wir verwenden noch einmal das Programm A , um die gegebene Definition der endlichen Automaten zu illustrieren. Wir haben das Programm A in Abbildung 2.3 graphisch dargestellt. Ein zum Programm A in Abbildung 2.3 äquivalenter EA ist $M = (Q, \Sigma, \delta, q_0, F)$ mit

$$\begin{aligned} Q &= \{q_0, q_1, q_2, q_3\}, & \Sigma &= \{0, 1\}, & F &= \{q_0, q_3\}, \\ &\{ \text{der Zustand } q_i \text{ entspricht der Zeile } i \text{ von } A \text{ für } i = 0, 1, 2, 3 \} \\ \delta(q_0, 0) &= q_2, & \delta(q_0, 1) &= q_1, & \delta(q_1, 0) &= q_3, & \delta(q_1, 1) &= q_0, \\ \delta(q_2, 0) &= q_0, & \delta(q_2, 1) &= q_3, & \delta(q_3, 0) &= q_1, & \delta(q_3, 1) &= q_2. \end{aligned}$$

Anschaulicher kann man die Übergangsfunktion δ durch die folgende Tabelle beschreiben:

Zustand	Eingabe	
	0	1
q_0	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Die prägnantere Darstellung eines EA ist aber die schon angesprochene graphische Form wie in Abbildung 2.3, in die der EA umgewandelt werden kann wie in Abbildung 2.8 gezeigt. Im Unterschied zu Abbildung 2.3 haben wir hier die Zustände als q_0 bis q_3 benannt, um in der formalen Beschreibung des Automaten einfacher zwischen den Zuständen und den Alphabetsymbolen unterscheiden zu können.

Die graphische Darstellung von $\delta(p, a) = q$ entspricht der Kante in Abbildung 2.9. Wenn $\delta(p, a) = q$ gilt, sprechen wir auch von einem **Zustandsübergang** oder einer **Transition** von p nach q mit dem Buchstaben a . Eine Transition entspricht also einer beschrifteten Kante in der graphischen Darstellung des Automaten.

Bedeutung dieses Begriffs und der Bezeichnung bei anderen Berechnungsmodellen, wie etwa bei Turingmaschinen.

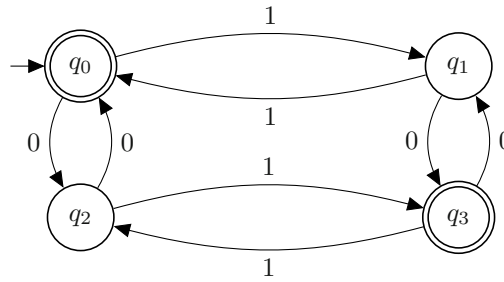


Abbildung 2.8 Ein endlicher Automat

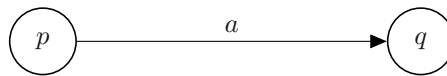


Abbildung 2.9 Graphische Darstellung eines Zustandsübergangs in einem EA

Aufgabe 2.6 Gib die formale Beschreibung als Quintupel für die endlichen Automaten aus Aufgabe 2.3 und Aufgabe 2.4 an.

Aufgabe 2.7 Zeichne die graphische Darstellung des endlichen Automaten $M = (Q, \Sigma, \delta, q_0, F)$ für $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{a, b\}$, $F = \{q_0\}$ und

$$\begin{aligned}
 \delta(q_0, a) &= q_1, & \delta(q_0, b) &= q_2, \\
 \delta(q_1, a) &= q_2, & \delta(q_1, b) &= q_3, \\
 \delta(q_2, a) &= q_3, & \delta(q_2, b) &= q_0, \\
 \delta(q_3, a) &= q_0, & \delta(q_3, b) &= q_1.
 \end{aligned}$$

Unter dem Begriff der Konfiguration in einem Rechnermodell verstehen wir eine vollständige Beschreibung der allgemeinen Situation (allgemeiner Zustand), in dem sich das Objekt der Modellklasse (zum Beispiel der Automat) befindet. Diese Beschreibung muss mindestens alle Informationen enthalten, die noch Einfluss auf eine Berechnung aus dieser Konfiguration haben dürfen. Bei endlichen Automaten gehört der Zustand des Automaten sowie der noch nicht gelesene Rest der Eingabe (der nicht gelesene Puffer des Eingabebandes) dazu. Ein Beispiel für eine Konfiguration eines EA ist in Abbildung 2.10 gezeigt.

Aufgabe 2.8 Nehmen wir an, ein endlicher Automat M hat angefangen, auf dem Wort *abbabba* zu arbeiten, und hat die Konfiguration $(q, abba)$ erreicht. Stelle die entsprechende Situation ähnlich wie in Abbildung 2.10 dar.

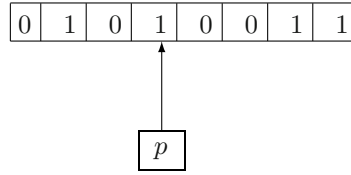


Abbildung 2.10 Ein Automat in der Konfiguration $(p, 10011)$

Definition 2.2 Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein endlicher Automat. Eine **Konfiguration** von M ist ein Element aus $Q \times \Sigma^*$.

$\left\{ \begin{array}{l} \text{Wenn } M \text{ sich in einer Konfiguration } (q, w) \in Q \times \Sigma^* \text{ befindet,} \\ \text{bedeutet dies, dass } M \text{ im Zustand } q \text{ ist und noch das Suffix } w \\ \text{eines Eingabewortes lesen soll (siehe Abbildung 2.10).} \end{array} \right\}$

Eine Konfiguration $(q_0, x) \in \{q_0\} \times \Sigma^*$ nennen wir **Startkonfiguration** von M auf x .

$\left\{ \begin{array}{l} \text{Die Arbeit (Berechnung) von } M \text{ auf } x \text{ muss in der Startkonfigu-} \\ \text{ration } (q_0, x) \text{ von } x \text{ beginnen.} \end{array} \right\}$

Jede Konfiguration aus $Q \times \{\lambda\}$ nennt man eine **Endkonfiguration**, weil alle Symbole der Eingabe schon gelesen worden sind und somit die Arbeit des Automaten beendet ist.

Ein **Schritt** von M ist eine Relation (auf Konfigurationen)

$$\vdash_M \subseteq (Q \times \Sigma^*) \times (Q \times \Sigma^*),$$

definiert durch

$$(q, w) \vdash_M (p, x) \text{ genau dann, wenn } w = ax, a \in \Sigma \text{ und } \delta(q, a) = p.$$

Ein Schritt entspricht einer Anwendung der Übergangsfunktion auf die aktuelle Konfiguration, in der sich der Automat in einem Zustand q befindet und ein Eingabesymbol a liest. Der Schritt $(q, ax) \vdash_M (p, x)$ ist in Abbildung 2.11 veranschaulicht.

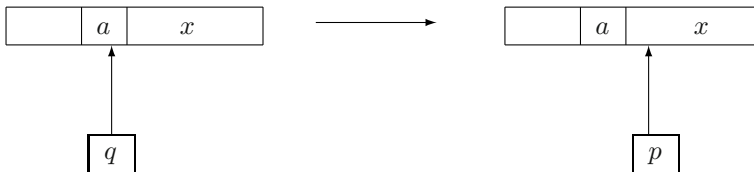


Abbildung 2.11 Ein Schritt eines EA

Aufgabe 2.9 Ist für jeden endlichen Automaten die Relation „Schritt“ reflexiv? Ist diese Relation immer symmetrisch oder transitiv? Falls du eine dieser Fragen bejahst, begründe dies allgemein für jeden möglichen EA. Falls deine Antwort „nein“ ist, konstruiere einen konkreten endlichen Automaten, dessen Relation „Schritt“ die betrachtete Eigenschaft nicht hat.

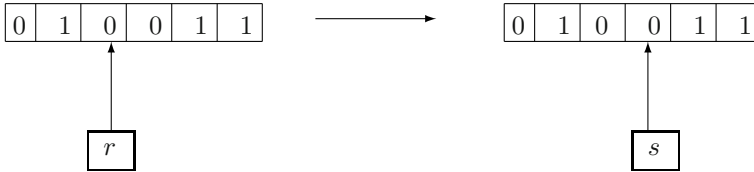


Abbildung 2.12 Ein Schritt eines EA aufgrund der Transition $\delta(r, 0) = s$

Den Schritt in Abbildung 2.12 verdanken wir dem durch $\delta(r, 0) = s$ gegebenen Übergang.

Aufgabe 2.10 Zeichne zwischen zwei Konfigurationen, wie in Abbildung 2.11 oder in Abbildung 2.12 dargestellt, die Übergänge ein, die den folgenden Schritten eines EA M entsprechen:

- (a) $(q_0, abbab) \xrightarrow{M} (q_1, bbab)$
- (b) $(q_0, aaaa) \xrightarrow{M} (q_7, aaa)$
- (c) $(p, 0110) \xrightarrow{M} (r, 110)$

Bei allen diesen Schritten ist das schon gelesene Präfix der Eingabe ohne Bedeutung. Du kannst es in deinem Bild einfach mit y bezeichnen.

Jetzt definieren wir die Berechnung als Folge von Berechnungsschritten.

Definition 2.3 Eine **Berechnung** $\mathcal{B}$ von M ist eine endliche Folge $\mathcal{B} = C_0, C_1, C_2, \dots, C_n$ von Konfigurationen, so dass $C_i \xrightarrow{M} C_{i+1}$ für alle $0 \leq i \leq n-1$ gilt. Um Berechnungen wiederzugeben, verwenden wir vorzugsweise die Darstellung

$$C_0 \xrightarrow{M} C_1 \xrightarrow{M} C_2 \xrightarrow{M} C_3 \dots \xrightarrow{M} C_n.$$

$\mathcal{B}$ ist die **Berechnung von M auf einer Eingabe $x \in \Sigma^*$** , falls $C_0 = (q_0, x)$ und $C_n \in Q \times \{\lambda\}$ eine Endkonfiguration ist.

Somit kann man eine Berechnung auf x auch als eine Folge von $|x|$ vielen Schritten aus der Konfiguration (q_0, x) über $|x| - 1$ Konfigurationen in die entsprechende Endkonfiguration betrachten.

Definition 2.4 Falls $C_n \in F \times \{\lambda\}$ gilt, sagen wir, dass $\mathcal{B}$ eine **akzeptierende Berechnung** von M auf x ist und M **das Wort x akzeptiert**. Falls $C_n \in (Q - F) \times \{\lambda\}$ gilt, sagen wir, dass $\mathcal{B}$ eine **verwerfende Berechnung** von M auf x ist und M **das Wort x verwirft (nicht akzeptiert)**.

Wir bemerken, dass M für jede Eingabe $x \in \Sigma^*$ genau eine Berechnung hat. In der Informatik nennen wir Berechnungsmodelle mit dieser Eigenschaft **deterministisch**.

Die Berechnung von M aus Abbildung 2.8 auf der Eingabe 1011 ist

$$(q_0, 1011) \mid_M (q_1, 011) \mid_M (q_3, 11) \mid_M (q_2, 1) \mid_M (q_3, \lambda).$$

Weil $q_3 \in F$ ist, wird 1011 von M akzeptiert.

Aufgabe 2.11 Schreibe die Berechnungen des endlichen Automaten aus Abbildung 2.8 für die folgenden Wörter auf:

- (a) 0000
- (b) 010101
- (c) 0011011

Welche der Wörter werden akzeptiert und welche verworfen?

Wenn $C_1 \mid_M C_2 \mid_M C_3 \mid_M \dots \mid_M C_m$ eine Berechnung des endlichen Automaten M ist, dann sagen wir, dass die Konfiguration C_m in einer Berechnung aus C_1 **erreichbar** ist und schreiben

$$C_1 \mid_M^* C_m.$$

Die Bezeichnung $C \mid_M^* D$ bedeutet, dass eine Folge von Berechnungsschritten existiert, in der der Automat aus der Konfiguration C in die Konfiguration D übergeht. Somit gilt zum Beispiel $(q_1, 011) \mid_M^* (q_2, 1)$ oder $(q_0, 1011) \mid_M^* (q_3, \lambda)$ für den endlichen Automaten M aus Abbildung 2.8. Die Anzahl der Schritte in dieser Folge kann immer auch 0 sein und deswegen gilt $H \mid_M^* H$ für jede Konfiguration H . Somit ist $\mid_M^*$ für jeden endlichen Automaten M eine reflexive Relation auf den Konfigurationen des Automaten. Mathematisch ausgedrückt ist $\mid_M^*$ eine Relation auf Konfigurationen, welche die transitive und reflexive Hülle der Relation $\mid_M$

ist. Die transitive und reflexive Hülle (auch transitiver und reflexiver Abschluss genannt) bedeutet, dass man zu der Relation $\vdash_M^*$ die kleinste mögliche Menge von Paaren zufügt, so dass man eine transitive und reflexive Relation erhält.

Definition 2.5 Die von M akzeptierte Sprache $L(M)$ ist definiert als

$$L(M) := \{w \in \Sigma^* \mid \text{die Berechnung von } M \text{ auf } w \text{ endet in einer} \\ \text{Endkonfiguration } (q, \lambda) \text{ mit } q \in F\}.$$

Mit $\mathcal{L}(\mathbf{EA}) = \{L(M) \mid M \text{ ist ein EA}\}$ bezeichnen wir die Klasse der Sprachen, die von endlichen Automaten akzeptiert werden. $\mathcal{L}(\mathbf{EA})$ bezeichnet man auch als die **Klasse der regulären Sprachen**, und jede Sprache L aus $\mathcal{L}(\mathbf{EA})$ wird **regulär** genannt.

Beispiel 2.2 Wir betrachten den EA M aus Abbildung 2.8. Die Frage ist, ob $(q_1, 1101) \vdash_M^* (q_3, 1)$ gilt. Um dies festzustellen, starten wir die Berechnung aus der Konfiguration $(q_1, 1101)$ und schauen, ob wir dabei die Konfiguration $(q_3, 1)$ erreichen. Es gilt

$$(q_1, 1101) \vdash_M (q_0, 101) \vdash_M (q_1, 01) \vdash_M (q_3, 1).$$

Wir haben die Konfiguration $(q_3, 1)$ erreicht. Somit ist die Antwort positiv.

Wir untersuchen jetzt die Frage, ob $(q_0, 0011101) \vdash_M^* (q_0, 01)$ gilt. Es gilt

$$(q_0, 0011101) \vdash_M (q_2, 011101) \vdash_M (q_0, 11101) \vdash_M (q_1, 1101) \vdash_M \\ (q_0, 101) \vdash_M (q_1, 01) \vdash_M (q_3, 1) \vdash_M (q_2, \lambda).$$

Diese Berechnung aus der Konfiguration $(q_0, 0011101)$ enthält alle Konfigurationen, die aus $(q_0, 0011101)$ erreichbar sind. Die Konfiguration $(q_0, 01)$ ist nicht dabei, deswegen ist unsere Antwort negativ. Wir beobachten aber, dass es nicht notwendig war, die ganze Berechnung durchzuführen, um festzustellen, dass $(q_0, 01)$ aus $(q_0, 0011101)$ nicht erreichbar ist. Das Wort 0011101 ist um fünf Buchstaben länger als das Wort 01, deswegen kann $(q_0, 01)$ aus $(q_0, 0011101)$ entweder in fünf Berechnungsschritten erreicht werden oder gar nicht. $\diamond$

Aufgabe 2.12 Kannst du feststellen, ob $C_1 \vdash_M^* C_2$ für M aus Abbildung 2.8 für folgende C_1 und C_2 gilt, ohne die Berechnung aufzuschreiben?

(a) $C_1 = (q_0, 110110), C_2 = (q_3, 1101)$

(b) $C_1 = (q_2, 0110), C_2 = (q_3, 0011)$

Begründe deine Antwort.

Aufgabe 2.13 Für welche der folgenden Paare von Konfigurationen (C_1, C_2) ist C_2 aus C_1 durch eine Berechnung des EA M (Abbildung 2.8) erreichbar?

- (a) $C_1 = (q_2, 0101001), C_2 = (q_2, 1)$
- (b) $C_1 = (q_1, 110110), C_2 = (q_3, 10)$
- (c) $C_1 = (q_0, 010100100), C_2 = (q_3, 100)$

Zusammenfassung

Endliche Automaten sind Programme mit nummerierten Zeilen, die keine Variablen, sondern nur die Verzweigungsbefehle **select** und **if ... then ... else** verwenden. Die Zeile, in der das Programm seine Arbeit beendet, bestimmt das Resultat. Endliche Automaten lösen Entscheidungsprobleme. Die Zeilen sind in zwei Gruppen aufgeteilt. Ein Gruppe bedeutet Akzeptanz, die andere Nichtakzeptanz (Verwerfen).

Bei der Visualisierung von endlichen Automaten ziehen wir eine graphische Darstellung vor. Die Zeilen bezeichnen wir als Knoten eines gerichteten Graphen. Einzelne **goto**-Anweisungen sind durch Kanten (Pfeile) dargestellt.

Ein Berechnungsschritt eines endlichen Automaten entspricht der Ausführung des Befehls einer Zeile. Wir verstehen darunter einen Übergang aus einer Konfiguration in eine andere. Mit dem Begriff der Konfiguration bezeichnen wir eine vollständige Beschreibung der Situation (des Gesamtzustands) eines Rechnermodells. Eine Berechnung kann man dann als eine Folge von Berechnungsschritten ansehen oder als eine Folge von Konfigurationen darstellen, wobei man aus jeder Konfiguration der Folge die nachfolgende Konfiguration in einem Schritt erreichen kann.

Zur mathematischen Darstellung eines endlichen Automaten verwenden wir den Begriff Zustand anstelle von Programmzeile.

Kontrollfragen

1. Was sind die Grundkomponenten eines Rechners? Welche fehlen bei endlichen Automaten?
2. Wie kann ein endlicher Automat rechnen, wenn er keine Variablen benutzen darf?
3. Welche sind die einzigen Befehle eines endlichen Automaten?
4. Wie löst ein endlicher Automat ein Entscheidungsproblem?
5. Welche drei Modelle von endlichen Automaten wurden vorgestellt? Wie wechseln die Namen der Grundbegriffe bei dem Übergang von Modell zu Modell?
6. Was ist eine Konfiguration eines endlichen Automaten?

7. Was ist ein Berechnungsschritt eines endlichen Automaten? Was ist eine Berechnung eines endlichen Automaten?
8. Wann betrachten wir eine Berechnung als eine Berechnung auf einem Wort? Wieviele Berechnungen gibt es auf einem Wort?
9. Wann akzeptiert ein endlicher Automat ein Wort? Wie stellen wir fest, dass ein EA ein gegebenes Wort verwirft (nicht akzeptiert)?

Kontrollaufgaben

1. Schreibe einen endlichen Automaten als ein zweizeiliges Programm, das
 - (a) über dem Alphabet $\{a, b, c, d\}$ arbeitet,
 - (b) mit der nullten Zeile akzeptiert und
 - (c) in jeder Zeile nach dem Lesen jedes Buchstabens die Zeile wechselt.
2. Stelle den EA aus Kontrollaufgabe 1 in den anderen Formen dar.
3. Bestimme die drei kürzesten Wörter über $\{0, 1\}$, die der EA in Abbildung 2.3 akzeptiert. Bestimme die vier kürzesten Wörter über $\{0, 1\}$, die der Automat nicht akzeptiert.
4. Betrachte den endlichen Automaten in Abbildung 2.4. Schreibe die Berechnungen des Automaten auf den Wörtern $0^6, 0^9, 0^5, 0101010, 0011001$ auf.
5. Betrachte den EA aus Abbildung 2.8. Finde zwei Konfigurationen C und D so, dass D aus C nicht erreichbar ist. Ist die Konfiguration $(q_0, 111)$ aus der Konfiguration $(q_1, 00010111)$ erreichbar?
6. Entwirf graphisch einen endlichen Automaten zum Verkauf von Briefmarken vom Wert 50c und 80c. Zum Kauf darf man nur die Münzen 10c, 20c und 50c verwenden. Der Verkaufsautomat gibt kein Geld zurück, also muss die Einkaufssumme genau einbezahlt werden.

Lösungen zu ausgewählten Aufgaben

Aufgabe 2.1 Wenn das Programm in der 0-ten Zeile das Symbol c liest, wird der Befehl `goto 0` ausgeführt. Somit bleibt das Programm in der 0-ten Zeile.

Wenn das Programm in der ersten Zeile das Symbol b liest, setzt die Ausführung des Programms wegen „`input = b goto 0`“ die Arbeit in der 0-ten Zeile fort.

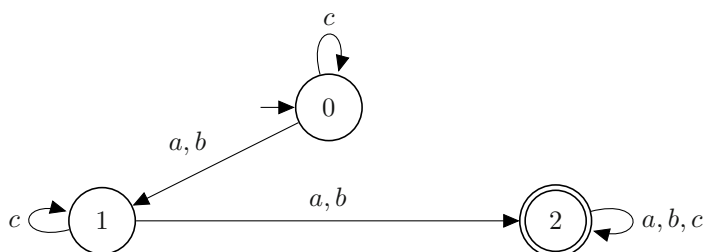
Aufgabe 2.2 Ob die konkreten Wörter akzeptiert werden oder nicht, kannst du sicherlich selbst überprüfen.

Wenn das Programm in Zeile 0 beginnt und das Wort 11 liest, landet es wieder in Zeile 0: Nach dem Lesen der ersten 1 geht es aus Zeile 0 in die Zeile 1 über und beim Lesen der zweiten 1 geht es aus Zeile 1 in die Zeile 0 über. Somit endet das Programm für alle Wörter $11, 1111, 111111, \dots, (11)^n, \dots$ in Zeile 0, wenn das Programm in Zeile 0 gestartet wird. Weil $0 \in F$, werden alle Wörter aus der unendlichen Menge $\{(11)^n \mid n \in \mathbb{N}\}$ von dem Programm akzeptiert.

Die Zeile 1 ist nicht in F . Somit wird das Wort 1 von dem Programm nicht akzeptiert. Alle Wörter 111, 11111, ..., $1(11)^n$, ... werden von dem Programm nicht akzeptiert. Erkläre, warum.

Finde andere unendliche Mengen von Wörtern, die nur aus Symbolen 0 bestehen, die von dem Programm nicht akzeptiert werden.

Aufgabe 2.3



Beim Lesen des Symbols c wird keine Zeile gewechselt. Somit spielen die Buchstaben c keine Rolle für die Akzeptierung. Die einzige akzeptierende Zeile 2 kann das Programm erreichen, wenn es mindestens zwei Symbole unterschiedlich von c (a oder b) gelesen hat. Wenn die Zeile 2 einmal erreicht wurde, wird sie nie wieder verlassen. Somit wird sichergestellt, dass dieser endliche Automat die Sprache $L = \{w \in \{a, b, c\}^* \mid |w|_a + |w|_b \geq 2\}$ akzeptiert.

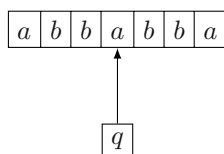
Aufgabe 2.4 Das Programm zu dem endlichen Automaten in Abbildung 2.4 sieht wie folgt aus:

```

0:      if input = 0 then goto 1 else goto 0
1:      if input = 0 then goto 2 else goto 1
2:      if input = 0 then goto 0 else goto 2
  
```

Die Zeile 1 ist die einzige akzeptierende Zeile. Das leere Wort λ wird damit nicht akzeptiert, weil das Programm mindestens ein Symbol braucht, um aus der Zeile 0 die Zeile 1 zu erreichen. Damit ist das kürzeste akzeptierte Wort das Wort 0. Es wird kein anderes Wort der Länge 1 akzeptiert. Von den Wörtern der Länge 2 werden die Wörter 10 und 01 akzeptiert. Von den Wörtern der Länge 3 sind es 011, 101 und 110.

Aufgabe 2.8 Welches Wort gelesen wurde, bis man die Konfiguration $(q, abba)$ erreicht hat, kann man aus der Konfiguration selbst nicht ablesen. Da wir aber wissen, dass der Automat die Konfiguration $(q, abba)$ bei der Berechnung auf dem Eingabewort *abbabba* erreicht hat, können wir die Konfiguration wie folgt darstellen:



Aufgabe 2.9 Die Relation „Schritt“ ist für keinen endlichen Automaten reflexiv oder symmetrisch. In einem Schritt wird immer ein Symbol gelesen, z. B. $(q, ax) \not\vdash_M (p, x)$. Somit ist es nicht möglich, in einem Schritt in der gleichen Konfiguration zu bleiben oder in eine Konfiguration überzugehen, in der der Automat noch mehr Symbole zu lesen hat als in der Ausgangskonfiguration. Somit kann $(p, x) \not\vdash_M (q, ax)$ nicht vorkommen und das Paar $((p, x), (q, ax))$ ist in keiner Relation Schritt vorhanden. Die Relation Schritt kann auch nicht transitiv sein. Wenn wir $(q, abx) \vdash_M (p, bx)$ und $(p, bx) \vdash_M (s, x)$ haben (das heißt, wenn $((q, abx), (p, bx))$ und $((p, bx), (s, x))$ Elemente der Relation Schritt sind), dann müsste auch $((q, abx), (s, x))$ ein Element der Relation Schritt sein. Aber $(q, abx) \not\vdash_M (s, x)$ ist unmöglich, weil der Automat in keinem Schritt zwei Buchstaben auf einmal lesen kann.

Aufgabe 2.12

- (a) Die Konfiguration $C_2 = (q_3, 1101)$ kann aus der Konfiguration $C_1 = (q_0, 110110)$ nicht erreicht werden, weil das Wort 1101 aus C_2 kein Suffix des Wortes 110110 aus C_1 ist.

Formale Sprachen

Endliche Automaten, Grammatiken, lexikalische und
syntaktische Analyse

Böckenhauer, H.-J.; Hromkovič, J.

2013, VIII, 245 S. 84 Abb., Softcover

ISBN: 978-3-658-00724-9