14 ENDLICHE AUTOMATEN

14.1 ERSTES BEISPIEL: EIN GETRÄNKEAUTOMAT

Als erstes Beispiel betrachten wir den folgenden primitiven Getränkeautomaten (siehe Abbildung 14.1). Man kann nur 1-Euro-Stücke einwerfen und vier Tasten drücken: Es gibt zwei Auswahltasten für Mineralwasser rein und Zitronensprudel zitro, eine Abbruch-Taste C und eine OK-Taste.

- Jede Flasche Sprudel kostet 1 Euro.
- Es kann ein Guthaben von 1 Euro gespeichert werden. Wirft man weitere Euro-Stücke ein, werden sie sofort wieder ausgegeben.
- Wenn man mehrfach Auswahltasten drückt, wird der letzte Wunsch gespeichert.
- Bei Drücken der Abbruch-Taste wird alles bereits eingeworfenen Geld wieder zurückgegeben und kein Getränkewunsch mehr gespeichert.
- Drücken der OK-Taste wird ignoriert, solange noch kein Euro eingeworfen wurde oder keine Getränkesorte ausgewählt wurde.

  Andernfalls wird das gewünschte Getränk ausgeworfen.

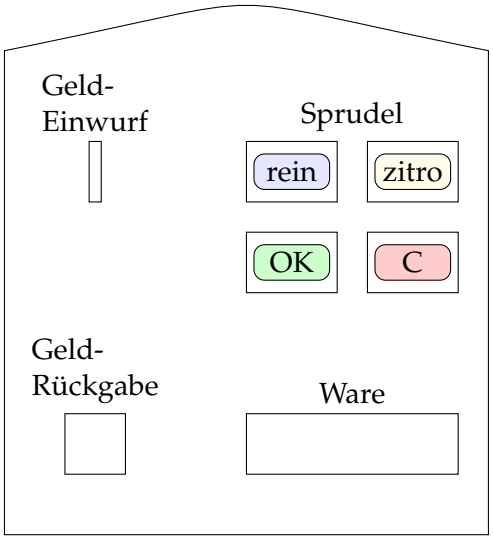

Abbildung 14.1: Ein primitiver Getränkeautomat

Dieser Getränke*automat* im umgangssprachlichen Sinne ist auch ein *endlicher Automat* wie sie in der Informatik an vielen Stellen eine Rolle spielen.

Offensichtlich muss der Automat zwischen den vielen Eingaben, die sein Verhalten beeinflussen können (Geldeinwürfe und Getränkewahl), gewisse Nachrichten (im Sinne von Abschnitt 2.3) speichern. Und zwar

- zum einen, ob schon ein 1-Euro-Stück eingeworfen wurde, und
- zum anderen, ob schon ein Getränk ausgewählt wurde und wenn ja, welches.

Man kann das zum Beispiel modellieren durch Paare (x, y), bei denen die Komponente $x \in \{0, 1\}$ den schon eingeworfenen Geldbetrag angibt und Komponente $y \in \{\texttt{-}, \texttt{R}, \texttt{Z}\}$ die Getränkewahl repräsentiert. Wir wollen $Z = \{0, 1\} \times \{\texttt{-}, \texttt{R}, \texttt{Z}\}$ die Menge der möglichen Zustände des Automaten nennen.

Der erste wesentliche Aspekt jedes Automaten ist, dass Einflüsse von außen, die wir *Eingaben* nennen, zu *Zustandsänderungen* führen. Bei dem Getränkeautomaten sind mögliche Eingaben der Einwurf eines 1-Euro-Stückes und das Drücken einer der Tasten (wir wollen davon absehen, dass jemand vielleicht mehrere Tasten gleichzeitig drückt). Wir modellieren die möglichen Eingaben durch Symbole `1`, `R`, `Z`, `C` und `O`, die zusammen das sogenannte *Eingabealphabet* X bilden. Ein aktueller Zustand $z \in Z$ und ein Eingabesymbol $x \in X$ legen — jedenfalls bei dem Getränkeautomaten — eindeutig den neuen Zustand fest. Dieser Aspekt eines endlichen Automaten kann also durch eine endliche Funktion $f : Z \times X \to Z$ formalisiert werden. In vielen Fällen ist es hilfreich, diese Funktion nicht durch eine Tabelle zu spezifizieren, sondern durch eine Darstellung als Graph wie in Abbildung 14.2.

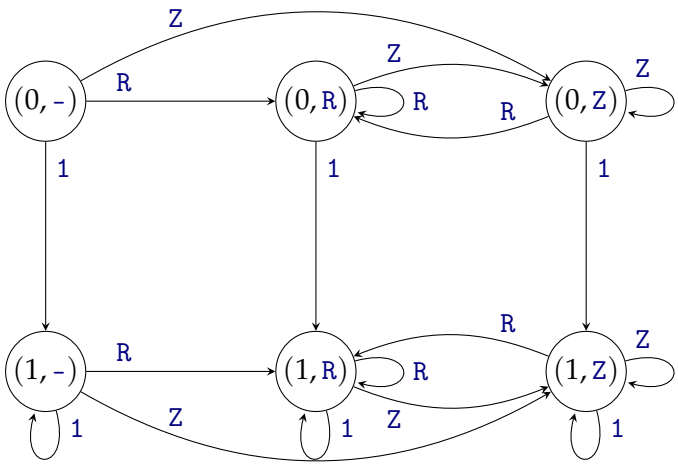

Abbildung 14.2: Graphische Darstellung der Zustandsübergänge des Getränkeautomaten für die drei Eingabesymbole `1`, `R` und `Z`.

Die Zustände sind die Knoten des Graphen, und es gibt gerichtete Kanten, die mit Eingabesymbolen beschriftet sind. Für jedes $z \in Z$ und jedes $x \in X$ führt eine mit

x beschriftete Kante von z nach $f(z, x)$.

Aus Gründen der Übersichtlichkeit sind in Abbildung 14.2 zunächst einmal nur die Zustandsübergänge für die Eingabesymbole 1, R und Z dargestellt. Hinzu kommen noch die aus Abbildung 14.3 für die Eingaben C und O. Wenn bei einem Zustand für mehrere Eingabesymbole der Nachfolgezustand der gleiche ist, dann zeichnet man oft nur einen Pfeil und beschriftet ihn mit allen Eingabesymbolen, durch Kommata getrennt. In Abbildung 14.3 betrifft das den Übergang von Zustand $(1, \mathtt{R})$ nach Zustand $(0, \mathtt{-})$ für die Eingaben O und C (und analog von den Zuständen $(1, \mathtt{Z})$ und $(0, \mathtt{-})$).

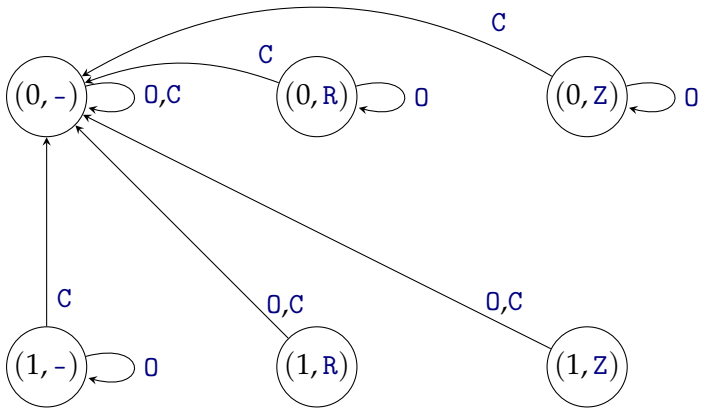

Abbildung 14.3: Graphische Darstellung der Zustandsübergänge des Getränkeautomaten für die Eingabesymbole C und O.

Stellt man alle Übergänge in einem Diagramm dar, ergibt sich Abbildung 14.4.
Der zweite wichtige Aspekt jedes Automaten ist, dass sich seine Arbeit, im vorliegenden Fall also die Zustandsübergänge, zumindest von Zeit zu Zeit in irgendeiner Weise auf seine Umwelt auswirken (warum sollte man ihn sonst arbeiten lassen). Beim Getränkeautomaten zeigt sich das in der Ausgabe von Geldstücken und Getränkeflaschen. Dazu sehen wir eine Menge $Y = \{1, \mathtt{R}, \mathtt{Z}\}$ von Ausgabesymbolen vor, deren Bedeutung klar sein sollte. Beim Getränkeautomaten ist es plausibel zu sagen, dass jedes Paar (z, x) von aktuellem Zustand z und aktueller Eingabe x eindeutig einen neuen Zustand festlegt, es ebenso eindeutig eine Ausgabe festlegt. Wir formalisieren das als eine Funktion $g : Z \times X \to Y^*$. Als Funktionswerte sind also Wörter von Symbolen aus Y erlaubt, einschließlich des leeren Wortes, das man zur Modellierung von „keine Ausgabe“ verwenden kann.

Auch die Funktion g wird üblicherweise in den Zustandsübergangsdiagrammen mit angegeben, und zwar an der jeweiligen Kante neben dem Eingabesym-

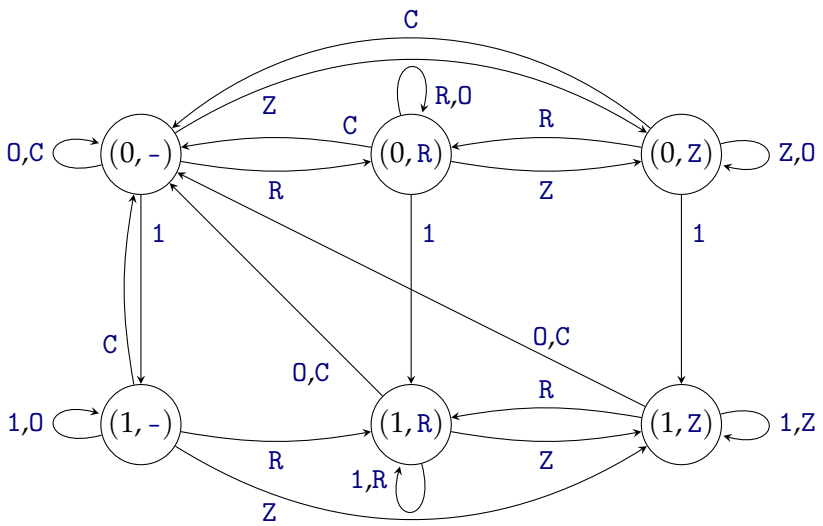

Abbildung 14.4: Graphische Darstellung der Zustandsübergänge des Getränkeautomaten für alle Eingabesymbole.

bol, von diesem durch einen senkrechten Strich getrennt (manche nehmen auch ein Komma). Aus Abbildung 14.4 ergibt sich Abbildung 14.5.

14.2 MEALY-AUTOMATEN

Mealy-Automat

Ein *(endlicher) Mealy-Automat* $A = (Z, z_0, X, f, Y, g)$ ist festgelegt durch

- eine endliche Zustandsmenge Z,
- einen Anfangszustand $z_0 \in Z$,
- ein Eingabealphabet X,
- eine Zustandsüberführungsfunktion $f : Z \times X \to Z$,
- ein Ausgabealphabet Y,
- eine Ausgabefunktion $g : Z \times X \to Y^*$

Für einen Zustand $z \in Z$ und ein Eingabesymbol $x \in X$ ist $f(z, x)$ der Zustand nach Eingabe dieses einzelnen Symbols ausgehend von Zustand z. Gleichzeitig mit jedem Zustandsübergang wird eine Ausgabe produziert. Wir modellieren das als Wort $g(z, x) \in Y^*$. In graphischen Darstellungen von Automaten wird der Anfangszustand üblicherweise dadurch gekennzeichnet, dass man einen kleinen Pfeil auf ihn zeigen lässt, der *nicht* bei einem anderen Zustand anfängt.

Manchmal möchte man auch über den nach Eingabe eines ganzen Wortes $w \in X^*$ erreichten Zustand oder über alle dabei durchlaufenen Zustände (ein-

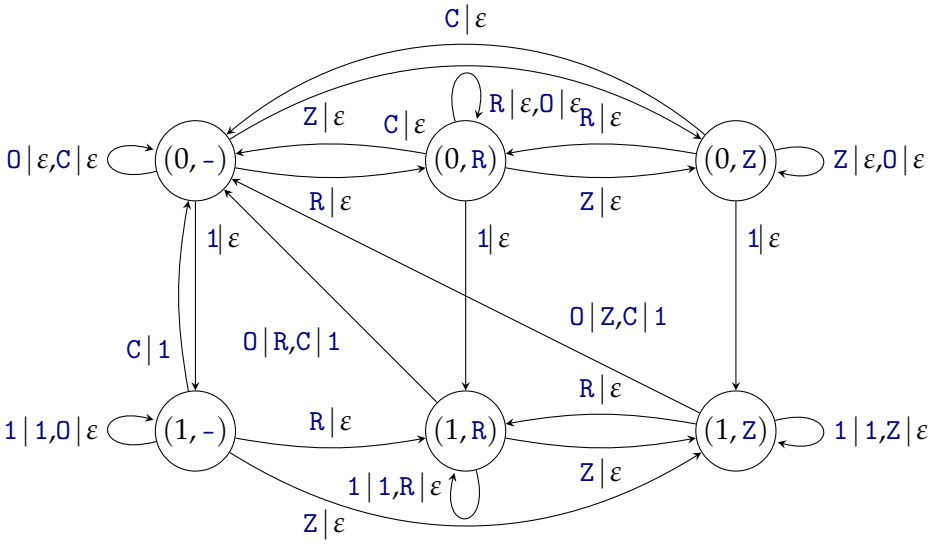

Abbildung 14.5: Graphische Darstellung der Zustandsübergänge und Ausgaben des Getränkeautomaten für alle Eingabesymbole.

schließlich des Anfangszustands) reden. Und manchmal will man auch bei den Ausgaben über allgemeinere Aspekte sprechen.

Um das bequem hinzuschreiben zu können, definieren wir Abbildungen f^* und f^{**} und analog g^* und g^{**}. Dabei soll der erste Stern andeuten, dass zweites Argument nicht ein einzelnes Eingabesymbol sondern ein ganzes Wort von Eingabesymbolen ist; und der zweite Stern soll gegebenenfalls andeuten, dass wir uns nicht für einen einzelnen Funktionswert (von f bzw. g) interessieren, sondern wiederum für ein ganzes Wort von ihnen. Als erstes legen wir $f^* : Z \times X^* \to Z$ fest: f^*

$$\begin{aligned} f^*(z, \varepsilon) &= z \\ \forall w \in X^* : \forall x \in X : \quad f^*(z, wx) &= f(f^*(z, w), x) \end{aligned}$$

Alternativ hätte man auch definieren können:

$$\begin{aligned} \bar{f}^*(z, \varepsilon) &= z \\ \forall w \in X^* : \forall x \in X : \quad \bar{f}^*(z, xw) &= \bar{f}^*(f(z, x), w) \end{aligned}$$

Machen Sie sich bitte klar, dass beide Definitionen die gleiche Funktion liefern (also $f^* = \bar{f}^*$): Für Argumente $z \in Z$ und $w \in X^*$ ist $f^*(z, w)$ der Zustand, in dem der Automat sich am Ende befindet, wenn er in z startet und der Reihe

nach die Eingabesymbole von w eingegeben werden. Je nachdem, was für einen Beweis bequem ist, können Sie die eine oder die andere Definitionsvariante zu Grunde legen. Das gleiche gilt für die folgenden Funktionen. (Sie dürfen sich aber natürlich nicht irgendeine Definition aussuchen, sondern nur eine, die zur explizit angegebenen äquivalent ist.)

Da wir vielleicht auch einmal nicht nur über den am Ende erreichten Zustand, sondern bequem über alle der Reihe nach durchlaufenen (einschließlich des Zustands, in dem man anfängt) reden wollen, legen wir nun $f^{**} : Z \times X^* \to Z^*$ für alle $z \in Z$ wie folgt fest:

f^{**}

$$\begin{aligned} & f^{**}(z, \varepsilon) = z \\ \forall w \in X^* : \forall x \in X : \qquad & f^{**}(z, wx) = f^{**}(z, w) \cdot f(f^*(z, w), x) \end{aligned}$$

Auch hier gibt es wieder eine alternative Definitionsmöglichkeit, indem man nicht das letzte, sondern das erste Symbol des Eingabewortes separat betrachtet.

Nun zu den verallgemeinerten Ausgabefunktionen. Zuerst definieren wir die Funktion $g^* : Z \times X^* \to Y^*$, deren Funktionswert die zum letzten Eingabesymbol gehörende Ausgabe sein soll. Das geht für alle $z \in Z$ so:

g^*

$$\begin{aligned} & g^*(z, \varepsilon) = \varepsilon \\ \forall w \in X^* : \forall x \in X : \qquad & g^*(z, wx) = g(f^*(z, w), x) \end{aligned}$$

Um auch über die Konkatenation der zu allen Eingabesymbolen gehörenden Ausgaben reden zu können, definieren wir die Funktion $g^{**} : Z \times X^* \to Y^*$ für alle $z \in Z$ wie folgt:

g^{**}

$$\begin{aligned} & g^{**}(z, \varepsilon) = \varepsilon \\ \forall w \in X^* : \forall x \in X : \qquad & g^{**}(z, wx) = g^{**}(z, w) \cdot g^*(z, wx) \end{aligned}$$

14.3 MOORE-AUTOMATEN

Manchmal ist es näherliegend, sich vorzustellen, dass ein Automat „in jedem Zustand“ eine Ausgabe produziert, und nicht bei jedem Zustandsübergang. Dementsprechend ist ein *Moore-Automat* $A = (Z, z_0, X, f, Y, h)$ festgelegt durch

Moore-Automat

- eine endliche Zustandsmenge Z,
- einen Anfangszustand $z_0 \in Z$,
- ein Eingabealphabet X,
- eine Zustandsüberführungsfunktion $f : Z \times X \to Z$,
- ein Ausgabealphabet Y,

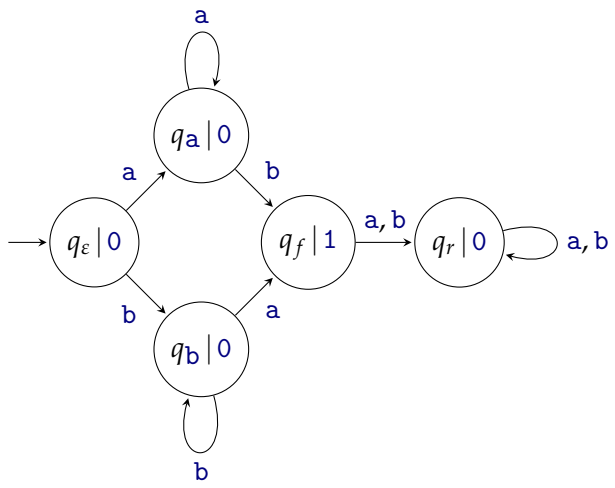

Abbildung 14.6: Ein einfacher Moore-Automat (aus der Dokumentation des LaTeX-Pakets `tikz`; modifiziert)

- eine Ausgabefunktion $h : Z \to Y^*$

Als einfaches Beispiel betrachten wir den Automaten in Abbildung 14.6 mit 5 Zuständen, Eingabealphabet $X = \{\mathtt{a}, \mathtt{b}\}$ und Ausgabealphabet $Y = \{\mathtt{0}, \mathtt{1}\}$.

In jedem Knoten des Graphen sind jeweils ein Zustand z und, wieder durch einen senkrechten Strich getrennt, die zugehörige Ausgabe $h(z)$ notiert.

Die Definitionen für f^* und f^{**} kann man ohne Änderung von Mealy- zu Moore-Automaten übernehmen. Zum Beispiel ist im obigen Beispiel $f^*(q_\varepsilon, \mathtt{aaaba}) = q_r$, denn bei Eingabe $\mathtt{aaaba}$ durchläuft der Automat ausgehend von q_ε nacheinander die Zustände f^* f^{**}

$$q_\varepsilon \xrightarrow{\mathtt{a}} q_{\mathtt{a}} \xrightarrow{\mathtt{a}} q_{\mathtt{a}} \xrightarrow{\mathtt{a}} q_{\mathtt{a}} \xrightarrow{\mathtt{b}} q_f \xrightarrow{\mathtt{a}} q_r$$

Und folglich ist auch $f^{**}(q_\varepsilon, \mathtt{aaaba}) = q_\varepsilon q_{\mathtt{a}} q_{\mathtt{a}} q_{\mathtt{a}} q_f q_r$.

Bei Mealy-Automaten hatten wir zu g die Verallgemeinerungen g^* und g^{**} definiert, die als Argumente einen Startzustand $z \in Z$ und ein Eingabewort $w \in X^*$ erhielten und deren Funktionswerte „die letzte Ausgabe“ bzw. „die Konkatenation aller Ausgaben“ waren.

Entsprechendes kann man natürlich auch bei Moore-Automaten festlegen. Die Definitionen fallen etwas einfacher aus als bei Mealy-Automaten. Zum Beispiel ist $g^* : Z \times X^* \to Y^*$ einfach hinzuschreiben als $g^*(z, w) = h(f^*(z, w))$ (für alle $(z, w) \in Z \times X^*$). Also kurz: $g^* = h \circ f^*$. g^*

Im obigen Beispielautomaten ist etwa

$$g^*(q_\varepsilon, \mathtt{aaaba}) = h(f^*(q_\varepsilon, \mathtt{aaaba})) = h(q_r) = \mathtt{0}$$

das zuletzt ausgegebene Bit, wenn man vom Startzustand ausgehend aaaba eingibt.

g^{**}

Auch $g^{**} : Z \times X^* \to Y^*$ für die Konkatenation aller Ausgaben ist leicht hinzuschreiben, wenn man sich des Begriffes des Homomorphismus erinnert, den wir in Unterabschnitt 10.2.2 kennengelernt haben. Die Ausgabeabbildung $h : Z \to Y^*$ induziert einen Homomorphismus $h^{**} : Z^* \to Y^*$ (indem man einfach h auf jeden Zustand einzeln anwendet). Damit ist für alle $(z, w) \in Z \times X^*$ einfach $g^{**}(z, w) = h^{**}(f^{**}(z, w))$, also $g^{**} = h^{**} \circ f^{**}$.

In unserem Beispiel ist

$$\begin{aligned} g^{**}(q_\varepsilon, \texttt{aaaba}) &= h^{**}(f^{**}(q_\varepsilon, \texttt{aaaba})) \\ &= h^{**}(q_\varepsilon q_{\texttt{a}} q_{\texttt{a}} q_{\texttt{a}} q_f q_r) \\ &= h(q_\varepsilon) h(q_{\texttt{a}}) h(q_{\texttt{a}}) h(q_{\texttt{a}}) h(q_f) h(q_r) \\ &= \texttt{000010} \end{aligned}$$

14.4 ENDLICHE AKZEPTOREN

Ein besonders wichtiger Sonderfall endlicher Moore-Automaten sind sogenannte endliche Akzeptoren. Unser Beispiel im vorangegangenen Abschnitt war bereits einer.

Die Ausgabe ist bei einem Akzeptor immer nur ein Bit, das man interpretiert als die Mitteilung, dass die Eingabe „gut“ oder „schlecht“ war, oder mit anderen Worten „syntaktisch korrekt“ oder „syntaktisch falsch“ (für eine gerade interessierende Syntax). Formal ist bei einem endlichen Akzeptor also $Y = \{0, 1\}$ und $\forall z : h(z) \in Y$. Man macht es sich dann üblicherweise noch etwas einfacher, und schreibt statt der Funktion h einfach die Teilmenge der sogenannten *akzeptierenden Zustände* auf. Damit ist $F = \{z \mid h(z) = 1\} \subseteq Z$ gemeint. Zustände, die nicht akzeptierend sind, heißen auch *ablehnend*.

akzeptierender Zustand
ablehnender Zustand
endlicher Akzeptor

Ein *endlicher Akzeptor* $A = (Z, z_0, X, f, F)$ ist also festgelegt durch

- eine endliche Zustandsmenge Z,
- einen Anfangszustand $z_0 \in Z$,
- ein Eingabealphabet X,
- eine Zustandsüberführungsfunktion $f : Z \times X \to Z$,
- eine Menge $F \subseteq Z$ akzeptierender Zustände

In graphischen Darstellungen werden die akzeptierenden Zustände üblicherweise durch doppelte Kringel statt einfacher gekennzeichnet. Abbildung 14.7 zeigt „den gleichen“ Automaten wie Abbildung 14.6, nur in der eben beschriebenen Form dargestellt. Es ist $F = \{q_f\}$, weil q_f der einzige Zustand mit Ausgabe 1 ist.

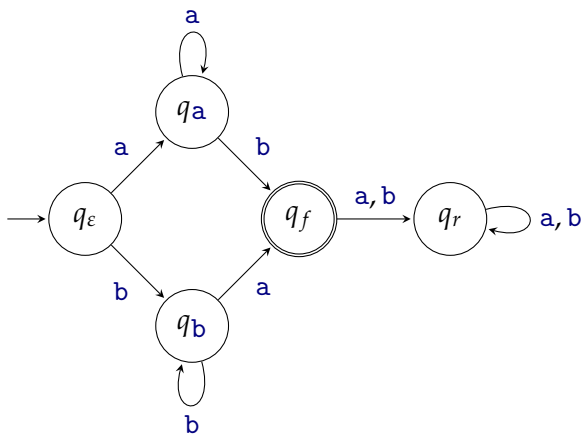

Abbildung 14.7: Ein einfacher Akzeptor (aus der Dokumentation des LaTeX-Pakets `tikz`; modifiziert)

14.4.1 Beispiele formaler Sprachen, die von endlichen Akzeptoren akzeptiert werden können

Man sagt, ein Wort $w \in X^*$ werde *akzeptiert*, falls $f^*(z_0, w) \in F$ ist, d. h. wenn man ausgehend vom Anfangszustand bei Eingabe von w in einem akzeptierenden Zustand endet. Wird ein Wort nicht akzeptiert, dann sagt man, dass es *abgelehnt* wird. Das schon mehrfach betrachtete Wort `aaaba` wird also abgelehnt, weil $f^*(z_0, \mathtt{aaaba}) = q_r \notin F$ ist. Aber z. B. das Wort `aaab` wird akzeptiert. Das gilt auch für alle anderen Wörter, die mit einer Folge von mindestens einem `a` beginnen, auf das genau ein `b` folgt, also alle Wörter der Form $\mathtt{a}^k\mathtt{b}$ für ein $k \in \mathbb{N}_+$. Und es werden auch alle Wörter akzeptiert, die von der Form $\mathtt{b}^k\mathtt{a}$ sind ($k \in \mathbb{N}_+$).

akzeptiertes Wort

abgelehntes Wort

Die von einem Akzeptor A *akzeptierte formale Sprache* $L(A)$ ist die Menge aller von ihm akzeptierten Wörter:

akzeptierte formale Sprache

$$L(A) = \{w \in X^* \mid f^*(z_0, w) \in F\}$$

In unserem Beispiel ist also

$$L(A) = \{\mathtt{a}\}^+\{\mathtt{b}\} \cup \{\mathtt{b}\}^+\{\mathtt{a}\}\ ,$$

denn außer den oben genannten Wörtern werden keine anderen akzeptiert. Das kann man sich klar machen, in dem man überlegt,

- dass Wörter ohne ein `b` oder ohne ein `a` abgelehnt werden
- dass Wörter, die sowohl mindestens zwei `a` als auch mindestens zwei `b` enthalten, abgelehnt werden, und

- dass Wörter abgelehnt werden, die z. B. nur genau ein a enthalten, aber sowohl davor als auch dahinter mindestens ein b, bzw. umgekehrt.

Eine im Alltag öfters vorkommende Aufgabe besteht darin, aus einer Textdatei diejenigen Zeilen zu extrahieren und z. B. auszugeben, in denen ein gewisses Wort vorkommt (und alle anderen Zeilen zu ignorieren). Jede Zeile der Textdatei ist eine Zeichenkette w, die darauf hin untersucht werden muss, ob ein gewisses Textmuster m darin vorkommt. So etwas kann ein endlicher Akzeptor durchführen.

Als Beispiel betrachten wir das Textmuster $m = \texttt{ababb}$ über dem Eingabealphabet $X = \{\texttt{a}, \texttt{b}\}$. Ziel ist es, einen endlichen Akzeptor A zu konstruieren, der genau diejenigen Wörter akzeptiert, in denen irgendwo m als Teilwort vorkommt. Die erkannte Sprache soll also $L(A) = \{w_1\texttt{ababb}w_2 \mid w_1, w_2 \in \{\texttt{a}, \texttt{b}\}^*\}$ sein.

Man kann diese Aufgabe natürlich ganz unterschiedlich angehen. Eine Möglichkeit, besteht darin, erst einmal einen Teil des Akzeptors hinzumalen, der „offensichtlich“ oder jedenfalls (hoffentlich) plausibel ist.

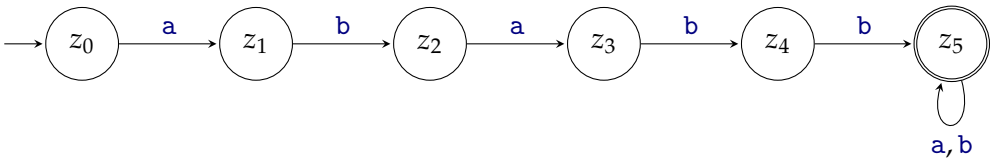

Abbildung 14.8: Teil eines Akzeptors für Wörter der Form $w_1\texttt{ababb}w_2$

Damit sind wir aber noch nicht fertig. Denn erstens werden noch nicht alle gewünschten Wörter akzeptiert (z. B. ababab), und zweitens verlangt unsere Definition endlicher Akzeptoren, dass für *alle* Paare (z, x) der nächste Zustand $f(z, x)$ festgelegt wird.

Zum Beispiel die genauere Betrachtung des Wortes abababb gibt weitere Hinweise. Nach Eingabe von abab ist der Automat in z_4. Wenn nun wieder ein a kommt, dann darf man nicht nach Zustand z_5 gehen, aber man hat zuletzt wieder aba gesehen. Das lässt es sinnvoll erscheinen, A wieder nach z_3 übergehen zu lassen. Durch weitere Überlegungen kann man schließlich zu dem Automaten aus Abbildung 14.9

Wir unterlassen es hier, im Detail zu beweisen, dass der Akzeptor aus Abbildung 14.9 tatsächlich die gewünschte Sprache erkennt. Man mache sich aber klar, dass für $0 \leq i \leq 4$ die folgende Aussage richtig ist:

> A ist genau dann in Zustand z_i, wenn das längste Suffix der bisher gelesenen Eingabe, das Präfix von ababb ist, gerade Länge i hat.

Für z_5 ist die Aussage etwas anders; überlegen Sie sich eine passende Formulierung!

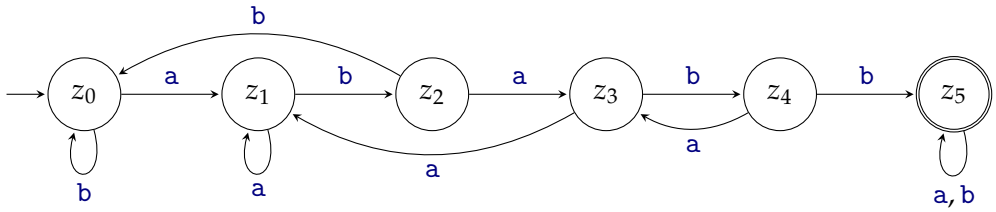

Abbildung 14.9: Der vollständige Akzeptor für alle Wörter der Form $w_1$ababbw_2

14.4.2 Eine formale Sprache, die von keinem endlichen Akzeptoren akzeptiert werden kann

Wir haben gesehen, dass es formale Sprachen gibt, die man mit endlichen Akzeptoren erkennen kann. Es gibt aber auch formale Sprachen, die man mit endlichen Akzeptoren *nicht* erkennen kann. Ein klassisches Beispiel ist die Sprache

$$L = \{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\} \ .$$

Bevor wir diese Behauptung beweisen, sollten Sie sich (falls noch nötig) klar machen, dass man natürlich ein (Java-)Programm schreiben kann, dass von einer beliebigen Zeichenkette überprüft, ob sie die eben angegebene Form hat. Man kann also mit „allgemeinen“ Algorithmen *echt mehr* Probleme lösen als mit endlichen Automaten.

14.1 Lemma. Es gibt keinen endlichen Akzeptor A mit

$$L(A) = \{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\} \ .$$

Können Sie sich klar machen, was diese Sprache „zu schwer“ macht für endliche Akzeptoren? Informell gesprochen „muss“ man zählen und sich „genau“ merken, wieviele a am Anfang eines Wortes vorkommen, damit ihre Zahl mit der der b am Ende „vergleichen“ kann.

14.2 Beweis. Machen wir diese Idee präzise. Wir führen den Beweis indirekt und nehmen an: Es gibt einen endlichen Akzeptor A, der genau $L = \{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\}$ erkennt. Diese Annahme müssen wir zu einem Widerspruch führen.

A hat eine gewisse Anzahl Zustände, sagen wir $|Z| = m$. Betrachten wir ein spezielles Eingabewort, nämlich $w = \mathtt{a}^m\mathtt{b}^m$.

1. Offensichtlich ist $w \in L$. Wenn also $L(A) = L$ ist, dann muss A bei Eingabe von w in einen akzeptierenden Zustand z_f gelangen: $f^*(z_0, w) = z_f \in F$.

2. Betrachten wir die Zustände, die A bei Eingabe der ersten Hälfte des Wortes durchläuft: $f^*(z_0, \varepsilon) = z_0$, $f^*(z_0, \mathtt{a})$, $f^*(z_0, \mathtt{aa})$, ..., $f^*(z_0, \mathtt{a}^m)$. Nennen wir diese Zustände allgemein z_i, d. h. für $0 \leq i \leq m$ ist $z_i = f^*(z_0, \mathtt{a}^i)$. Mit anderen Worten: $f^{**}(z_0, \mathtt{a}^m) = z_0 z_1 \cdots z_m$.

   Offensichtlich gilt dann: $f^*(z_m, \mathtt{b}^m) = z_f$.

   Andererseits besteht die Liste $z_0 z_1 \cdots z_m$ aus $m+1$ Werten. Aber A hat nur m verschiedene Zustände. Also kommt mindestens ein Zustand doppelt vor.

   D. h. der Automat befindet sich in einer Schleife. Sei etwa $z_i = z_j$ für gewisse $i < j$. Genauer sei z_i das erste Auftreten irgendeines mehrfach auftretenden Zustandes und z_j das zweite Auftreten des gleichen Zustandes. Dann gibt es eine „Schleife" der Länge $\ell = j - i > 0$. Und ist der Automat erst einmal in der Schleife, dann bleibt er natürlich darin, solange er weitere $\mathtt{a}$ als Eingabe erhält. Also ist auch $z_{m-\ell} = z_m$.
3. Nun entfernen wir einige der $\mathtt{a}$ in der Eingabe, so dass die Schleife einmal weniger durchlaufen wird, d. h. wir betrachten die Eingabe $w' = \mathtt{a}^{m-\ell}\mathtt{b}^m$. Wie verhält sich der Akzeptor bei dieser Eingabe? Nachdem er das Präfix $\mathtt{a}^{m-\ell}$ gelesen hat, ist er in Zustand $z_{m-\ell}$. Dieser ist aber gleich dem Zustand z_m, d. h. A ist in dem Zustand in dem er auch nach der Eingabe $\mathtt{a}^m$ ist. Und wir wissen: $f^*(z_m, \mathtt{b}^m) = z_f \in F$. Also ist $f^*(z_0, \mathtt{a}^{m-\ell}\mathtt{b}^m) = f^*(f^*(z_0, \mathtt{a}^{m-\ell}), \mathtt{b}^m) = f^*(z_{m-\ell}, \mathtt{b}^m) = f^*(z_m, \mathtt{b}^m) = z_f$, d. h. A akzeptiert die Eingabe $w' = \mathtt{a}^{m-\ell}\mathtt{b}^m$. Aber das Wort w' gehört nicht zu L, da es verschieden viele $\mathtt{a}$ und $\mathtt{b}$ enthält! Also ist $L(A) \neq L$. Widerspruch!

Also war die Annahme falsch und es gibt gar keinen endlichen Akzeptor, der L erkennt. ■

14.5 AUSBLICK

Wir haben nur endliche Automaten betrachtet, bei denen $f : Z \times X \to Z$ eine Funktion, also linkstotal und rechtseindeutig, ist. Auch Verallgemeinerungen, bei denen f eine beliebige Relation sein darf, sind als sogenannte *nichtdeterministische endliche Automaten* ausgiebig untersucht und spielen an vielen Stellen in der Informatik eine Rolle (zum Beispiel bei Compilerbau-Werkzeugen).

Nichtdeterminismus

Deterministischen Automaten, also die, die wir in dieser Einheit betrachtet haben, werden Sie in Vorlesungen über Betriebssysteme und Kommunikation wiederbegegnen, z. B. im Zusammenhang mit der Beschreibung von sogenannten *Protokollen*.