3 Endliche Automaten

In Kapitel 1 wird exemplarisch eine typische Situation in der Datebverarbeitung mit Hilfe von Zustandsgraphen modelliert. Zustandsgraphen sind graphische Darstellungen endlicher Automaten, die in diesem Kapitel formal eingeführt werden. Sie bilden ein in der Informatik häufig und sehr erfolgreich eingesetztes Modellierungswerkzeug, das vergleichsweise einfach ist und sich deshalb als Einstieg in den Bereich der formalen Modellierungsmethoden eignet. Ein endicher Automat spezifiziert in seiner einfachsten Form eine Sprache, d.h. eine Menge von Wörtern. Als ein erstes interessantes Ergebnis stellt sich heraus, dass sich jeder endliche Automat deterministisch machen lässt, ohne seine Sprache zu verändern. Das ist deshalb signifikant, weil ein deterministischer Automat sehr schnell erkennen kann, ob ein Wort zu seiner Sprache gehört oder nicht.

Unter einem endlichen Automaten hat man sich ein taktweise arbeitendes System vorzustellen, das sich zu jedem Zeitpunkt in einem bestimmten Zustand (von endlich vielen verfügbaren Zuständen) befindet und das dann innerhalb eines Taktes einen Buchstaben eines Eingabewortes einliest, daraufhin seinen Zustand ändert und den Lesekopf auf den nächsten Eingabebuchstaben rechts einstellt usw. Am Anfang befindet sich der Automat in einem Anfangszustand und der Lesekopf steht ganz links auf dem ersten Buchstaben des Eingabewortes. Ein Eingabewort gilt dann als erkannt, wenn vom Anfangszustand aus nach Verarbeitung des Wortes ein Endzustand erreicht wird. Die erkannten Wörter bilden die Sprache des Automaten.

3.1 Endlicher Automat, fortgesetzte Zustandsüberführung und erkannte Sprache

Die Veranschaulichung führt zu folgender Definition:

1. Ein *endlicher relationeller erkennender Automat* – kurz *endlicher Automat* – ist ein System $A = (Z, I, d, s_0, F)$, wobei
   - Z eine endliche Menge von *Zuständen* ist,
   - I ein endliches *Eingabealphabet*,
   - $s_0 \in Z$ der *Anfangszustand*,
   - $d \subseteq Z \times I \times Z$ eine Relation ist, geschrieben $d\colon Z \times I \rightsquigarrow Z$, die *Zustandsüberführung* genannt wird und jedem Zustand und jeder Eingabe eine Menge von Folgezuständen zuordnet und
   - $F \subseteq Z$ eine Menge von *Endzuständen* ist.
2. Die *Zustandsüberführung* d lässt sich rekursiv zu einer Relation $d^* \subseteq Z \times I * \times Z$ bzw. $d^*\colon Z \times I^* \rightsquigarrow Z$ fortsetzen, die nicht nur Zeichen, sondern Eingabewörter verarbeitet:
   (i) $d^*(s, \lambda) = \{s\}$ für alle $s \in Z$ und
   (ii) $d^*(s, wx) = \bigcup_{t \in d^*(s,w)} d(t, x)$ für alle $s \in Z$, $x \in I$, $w \in I^*$.
3. Die von einem endlichen Automaten A *erkannte Sprache* $L(A)$ ist dann definiert durch:
$$L(A) = \{w \in I^* \mid d^*(s_0, w) \cap F \neq \emptyset\}.$$
4. Ein Automat $A = (Z, I, d, s_0, F)$ heißt *deterministisch*, wenn d eine Abbildung ist; d.h. für alle $s \in Z$ und $x \in I$ existiert genau ein $s' \in Z$ derart, dass (s, x) und s' bzgl. d in Relation stehen. In diesem Fall wird auch $d\colon Z \times I \to Z$ und $d(s, x) = s'$ geschrieben, wie es für Abbildungen üblich ist.

Bemerkung zur Relationsschreibweise

Ist $R\colon A \rightsquigarrow B$ eine Relation (d.h. $R \subseteq A \times B$), dann wird statt $(a, b) \in R$ oft auch aRb geschrieben oder, wenn R eine Abbildung ist, $R(a) = b$. Ansonsten bezeichnet $R(a)$ die Menge aller Elemente, die bzgl. R zu a in Relation stehen: $R(a) = \{b \in B \mid aRb\}$.

3.2 Fortgesetzte Zustandsüberführung und erkannte Sprache von deterministischen Automaten

Für deterministische Automaten ist mit d auch d^* eine Abbildung, wie man leicht durch Induktion über w zeigt. Deshalb können (i) und (ii) aus Punkt 2 in diesem Fall auch ausgedrückt werden durch:

(i') $d^*(s, \lambda) = s$ und

(ii') $d^*(s, wx) = d(d^*(s, w), x)$.

Ferner ist die von deterministischen Automaten erkannte Sprache bestimmt durch:

$$L(A) = \{w \in I^* \mid d^*(s_0, w) \in F\}.$$

3.3 Zustandsgraph

Jeder endliche Automat A besitzt eine graphische Darstellung in Form eines *Zustandsgraphen*. Dabei werden die Zustände zu Knoten, und von einem Zustand s zu einem Zustand s' wird eine mit $x \in I$ markierte Kante gezogen, falls $s' \in d(s, x)$. Falls es mehrere Zeichen $x_1, \ldots, x_k \in I$ gibt mit $s' \in d(s, x_i)$ für alle $i \in \{1, \ldots, n\}$, so wird der Übersichtlichkeit halber nur eine Kante gezogen, an der dann $x_1, \ldots, x_k$ steht. Der Anfangszustand wird durch einen hineingehenden Pfeil, die Endzustände werden entsprechend durch herausgehende Pfeile gekennzeichnet.

3.4 Beispiel

In Abbildung 1 ist ein kleiner endlicher Automat dargestellt. Er erkennt gerade die Sprache der positiven ganzen Zahlen in Binärdarstellung ohne führende Nullen, d.h. die Menge $\{1w \mid w \in \{0, 1\}^*\}$. Dieser Automat ist nichtdeterministisch; denn für die Zustandsüberführung d gilt: $d(s_0, 1) = \{s_0, s_1\}$ und $d(s_0, 0) = \emptyset$.

3.5 Verarbeitung von Wörtern in iterativer Darstellung

Die fortgesetzte Zustandsüberführung ist für Eingabewörter rekursiv definiert. Es gibt aber auch eine iterative Version, die vielleicht etwas anschaulicher ist, vor allem aber in manchen Situationen besser zu gebrauchen.

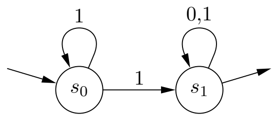

Abbildung 1: Graphische Darstellung eines endlichen Automaten

Sei $A = (Z, I, d, s_0, F)$ ein endlicher Automat und $w = a_1 \cdots a_n \in A^*$ mit $a_i \in A$ für $i = 1, \ldots, n$. Dann ist $s' \in d^*(s, w)$ für $s, s' \in Z$ genau dann, wenn es eine Folge von Zuständen $t_0, \ldots, t_n$ gibt, derart dass $s = t_0$, $s' = t_n$ und $t_i \in d(t_{i-1}, a_i)$ für $i = 1, \ldots, n$.
Dabei ist der Fall $n = 0$ zugelassen und betrifft das leere Wort $w = a_1, \ldots, a_0 = \lambda$ und die Zustandsfolge t_0 mit $s = t_0 = s'$.

Beweisen lässt sich das mit Induktion über die Struktur oder Länge der Eingabewörter, worauf hier allerdings verzichtet wird.
Im Zustandsgraph sieht die Verarbeitung eines Eingabewortes $w = a_1 \cdots a_n$ demnach so aus:

3.6 Schnelle Spracherkennung durch deterministische Automaten

Mit jeder Überführung des Zustandes eines deterministischen Automaten in den (eindeutigen) Nachfolgezustand wird ein Buchstabe des Eingabewortes abgearbeitet. Erst wenn das Eingabewort vollständig durchlaufen ist, bleibt der Automat stehen. Es sind also n Schritte erforderlich, um zu entscheiden, ob ein Eingabewort der Länge n zur erkannten Sprache gehört. Damit ist das sogenannte Wortproblem für jede von einem deterministischen Automaten erkannte Sprache in linearer Zeit lösbar.
Gilt das auch, wenn der erkennende Automat nichtdeterministisch ist?

3.7 Der Potenzautomat

Offensichtlich ist Determinismus eine echte Einschränkung für endliche Automaten. Ist damit aber auch die Klasse der Sprachen, die von deterministischen Automaten erkannt werden, eine echte Teilmenge der von allgemeinen endlichen Automaten erkannten Sprachen? Das folgende Theorem verneint dies.

Theorem 1
Zu jedem endlichen Automaten A lässt sich effektiv ein deterministischer Automat $\mathcal{P}(A)$ konstruieren, der dieselbe Sprache erkennt; d.h.

$$L(A) = L(\mathcal{P}(A)).$$

Beweis.
Sei $A = (Z, I, d, s_0, F)$ ein endlicher Automat.

Daraus lässt sich der *Potenzautomat* konstruieren:

$$\mathcal{P}(A) = (\mathcal{P}(Z), I, D, \{s_0\}, F_{\mathcal{P}} := \{S \subseteq Z \mid S \cap F \neq \emptyset\}),$$

wobei $\mathcal{P}(Z)$ die Potenzmenge von Z ist und die Abbildung $D \colon \mathcal{P}(Z) \times I \to \mathcal{P}(Z)$ definiert ist durch $D(S, x) := \bigcup_{s \in S} d(s, x)$ für alle $S \in \mathcal{P}(Z)$ und $x \in I$.
Dann stehen die fortgesetzten Zustandsüberführungen d^* und D^* in der Beziehung

$$d^*(s, w) = D^*(\{s\}, w).$$

Denn für $w = \lambda$ gilt:

$$d^*(s, \lambda) = \{s\} = D^*(\{s\}, \lambda),$$

und für Wörter wx erhält man, vorausgesetzt, die Behauptung gilt bereits für w:

$$\begin{aligned} d^*(s, wx) &= \bigcup_{t \in d^*(s,w)} d(t, x) \\ &= \bigcup_{t \in D^*(\{s\},w)} d(t, x) \\ &= D(D^*(\{s\}, w), x) = D^*(\{s\}, wx). \end{aligned}$$

Für die Sprachen $L(A)$ und $L(\mathcal{P}(A))$ folgt somit:

$$\begin{aligned} w \in L(A) \quad &\text{gdw.} \quad d^*(s_0, w) \cap F \neq \emptyset \\ &\text{gdw.} \quad D^*(\{s_0\}, w) (= d^*(s_0, w)) \in F_{\mathcal{P}} \\ &\text{gdw.} \quad w \in L(\mathcal{P}(A)); \end{aligned}$$

$L(A)$ und $L(\mathcal{P}(A))$ sind also gleich. □

3.8 Beispiel

Der Potenzautomat zu dem endlichen Automaten aus Abbildung 1 ist in Abbildung 2 dargestellt. Zum Beispiel ist $D(\{s_0, s_1\}, 0) = d(s_0, 0) \cup d(s_1, 0) = \emptyset \cup \{s_1\} = \{s_1\}$ sowie $D(\{s_0, s_1\}, 1) = d(s_0, 1) \cup d(s_1, 1) = \{s_0, s_1\} \cup \{s_1\} = \{s_0, s_1\}$ und immer $D(\emptyset, x) = \emptyset$.

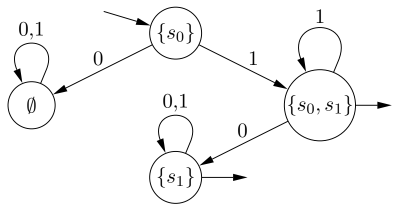

Abbildung 2: Der Potenzautomat zu dem endlichen Automaten aus Abbildung 1

3.9 Schnelle Spracherkennung durch endliche Automaten

Mit dem Theorem 1 folgt sofort, dass das Wortproblem für jede von einem beliebigen endlichen Automaten erkannte Sprache linear lösbar ist. Denn man kann erst den Potenzautomaten konstruieren, was nur konstant viel Zeit in Anspruch nimmt (zumindest bezogen auf die Länge der Eingabewörter). Danach ist das Wortproblem linear lösbar, weil der Potenzautomat deterministisch ist, aber immer noch dieselbe Sprache erkennt.

3.10 Anmerkung zur Literatur

Wer sich über das Skript hinaus über endliche Automaten informieren möchte, denen sei das grundlegende Werk zur theoretischen Informatik empfohlen:

[1] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie. Addison-Wesley, 2002.

Die englische Ausgabe ist 2001 unter dem Titel Introduction to Automata Theory, Languages, and Computation im selben Verlag erschienen:

[2] John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley, 2001.

4 Produktautomat erkennt Durchschnitt

Wie im Kapitel 1 versprochen, kann ein endlicher Automat konstruiert werden, der den Durchschnitt zweier Sprachen erkennt, die selbst von endlichen Automaten erkannt werden. Eine einfache Modifikation der Endzustände befähigt diesen sogenannten Produktautomaten ebenfalls, die Vereinigung der beiden Sprachen zu erkennen.

Die Idee des Produktautomaten ist, zwei endliche Automaten über das kartesische Produkt ihrer Zustandsmengen so zu koppeln, dass sie Eingabewörter parallel abarbeiten.

4.1 Produktautomat

Seien $A_i = (Z_i, I, d_i, s_{0i}, F_i)$ für $i = 1, 2$ zwei deterministische Automaten. Dann ist der *Produktautomat* definiert durch

$$A_1 \times A_2 = (Z_1 \times Z_2, I, d, (s_{01}, s_{02}), F_1 \times F_2)$$

mit $d((s_1, s_2), x) = (d_1(s_1, x), d_2(s_2, x))$ für alle $(s_1, s_2) \in Z_1 \times Z_2$ und $x \in I$.

Der Produktautomat führt also die Zustandsübergänge der beiden Einzelautomaten parallel durch. Wie das folgende Lemma zeigt, gilt das auch für die fortgesetzte Zustandsüberführung. Da der Anfangszustand aus den beiden einzelnen Anfangszuständen und die Endzustände Paare der einzelnen Endzustände sind, ergibt sich aus dem Lemma die gewünschte Durchschnittseigenschaft.

4.2 Beobachtung (Erkennung des Durchschnitts)

$$L(A_1 \times A_2) = L(A_1) \cap L(A_2).$$

Beweis.
$w \in L(A_1 \times A_2)$ gdw. nach folgendem Lemma

$$d^*((s_{01}, s_{02}), w) = (d_1^*(s_{01}, w), d_2^*(s_{02}, w)) \in F_1 \times F_2$$

gdw. $d_i^*(s_{0i}, w) \in F_i$ für $i = 1, 2$ gdw. $w \in L(A_i)$ für $i = 1, 2$. □

4.3 Lemma

$$d^*((s_1, s_2), w) = (d_1^*(s_1, w), d_2^*(s_2, w)) \text{ für alle } (s_1, s_2) \in Z_1 \times Z_2 \text{ und } w \in I^*.$$

Beweis.
Der Beweis wird mit vollständiger Induktion über den Aufbau von w geführt.

IA: $d^*((s_1, s_2), \lambda) = (s_1, s_2) = (d_1^*(s_1, \lambda), d_2^*(s_2, \lambda))$, wobei die Definition der fortgesetzten Zustandsüberführung verwendet wird.

IS: $d^*((s_1, s_2), wx) = d(d^*((s_1, s_2), w), x) =$
$d((d_1^*(s_1, w), d_2^*(s_2, w)), x) =$
$(d_1(d_1^*(s_1, w), x), d_2(d_2^*(s_2, w), x)) =$
$(d_1^*(s_1, wx), d_2^*(s_2, wx))$.

Dabei ergeben sich die Gleichheiten in der gegebenen Reihenfolge aus der Definition der fortgesetzten Zustandsüberführung, der Induktionsvoraussetzung, der Definition der Zustandsüberführung des Produktautomaten und erneut aus der Definition der fortgesetzten Zustandsüberführung. □

4.4 Erkennung der Vereinigung

Wenn beim Abarbeiten eines Eingabewortes einer der beiden Automaten A_1 und A_2 in einen Endzustand kommen, dann gehört das Wort gerade zu der Vereinigung der erkannten Sprachen. Nach Lemma 4.3 erkennt der Produktautomat aber genau diese Vereinigung $L(A_1)) \cup L(A_2)$, wenn man die Menge $(F_1 \times Z_2) \cup (Z_1 \times F_2)$ als Endzustände wählt.
Ginge es nur um die Vereinigung, so ließe sich allerdings ein Vereinigungsautomat bauen, bei dem die Zustandsmengen nicht über das Produkt, sondern die Vereinigung gekoppelt wären, was in den allermeisten Fällen deutlich weniger Zustände ergäbe.