5 Endliche Automaten und reguläre Mengen

Gewöhnlich wird unter einem Automaten oder einer Maschine ein mechanisches oder elektronisches Gerät verstanden, bei dem ausschließlich die Funktion ausschlaggebend für seine Beurteilung ist. Mechanische Geräte wie Hammer, Lupe oder Schreibmaschine werden bei komplexeren Funktionen von schlichten Werkzeugen unterschieden. Ähnliches findet man bei elektronischen Geräten, wie zum Beispiel bei der Unterscheidung zwischen Transistor, Solarzelle oder Computer. Im täglichen Leben finden wir an fast jeder Straßenecke Automaten: Zigarettenautomaten, Fahrkartenautomaten, Musikbox, Glücksspielautomat; aber auch zu Hause: Telefon, CD-Spieler, Fernseher oder Anrufbeantworter. Gemeinsam sind all diesen realen Maschinen folgende Charakteristika: **Eingabeinformationen** (z.B.: Tastendruck, Münzeinwurf, Wählaktion, o.ä.), **interne Verarbeitung** ohne Rückgriff auf externe weitere Informationsquellen (z.B.: Prüfen der eingeworfenen Münzen, Berechnung des Fahrpreises auf Grund der Zielangabe, Summation der eingeworfenen Beträge) und **Ausgabeinformationen** (z.B.: Geldauswurf, Fahrkarte und Wechselgeld, Zigaretten, o.ä.). All diese Funktionen werden in der Regel unzweideutig stets in der gleichen Weise ausgeführt. Wir beginnen mit dem einfachsten Automatenmodell: dem endlichen Automaten.

5.1 Deterministische endliche Automaten

Um nun von den nicht die Funktion betreffenden Aspekten wie Material, Farbe oder Herstellungspreis absehen zu können, wird in der Theoretischen Informatik von realen Bauweisen abstrahiert und nur Wesentliches beschrieben. Wir beginnen mit dem einfachsten Modell, dem **endlichen Automaten**. Dieser modelliert in der Regel eine Maschine, die ihre internen Zustände auf Grund von festgelegten Eingaben ändern kann und *keine Ausgabe* erzeugt. Deterministische endliche Automaten werden aber auch mit Ausgabefunktionen versehen und heißen dann nach ihren Erfindern Moore-Automat oder Mealy-Automat. Ein deterministischer endlicher Automat wird **DFA** abgekürzt (***d**eterministic **f**inite **a**utomaton*).

5.1 Definition (DFA)
*Ein **deterministischer endlicher Automat** (**DFA**) wird durch ein 5-Tupel $A := (Z, \Sigma, \delta, z_0, Z_{end})$ beschrieben:*

- Z *ist eine endliche Menge von* ***Zuständen****.*
- Σ *ist ein endliches Alphabet von* ***Eingabesymbolen****.*
- $\delta : Z \times \Sigma \longrightarrow Z$ *ist die* ***Überführungsfunktion*** *(Diese muss nicht total sein).*
- $z_0 \in Z$ *ist der* ***Startzustand****.*
- $Z_{end} \subseteq Z$ *ist die Menge der* ***Endzustände****.*

Zustände werden meist mit $z_0, z_1, \ldots$ oder mit $p_i, q_j, \ldots$ etc. bezeichnet. Für Eingabesymbole werden üblicherweise Buchstaben vom Anfang des lateinischen Alphabetes verwendet, also $a, b, c, \ldots$ o.ä.

Wir veranschaulichen einen DFA durch seinen **Zustandsgraphen**. Dies ist ein Kanten-bewerteter, gerichteter Graph, dessen Knoten eineindeutig den Zuständen des DFA zugeordnet sind.

Da die Knotenmenge des Zustandsgraphen isomorph zu der Zustandsmenge des Automaten ist, wollen wir hier jeden Knoten als einen Kreis um die Zustandsbezeichnung zeichnen: $\textcircled{z}$. Eine mit dem Zeichen

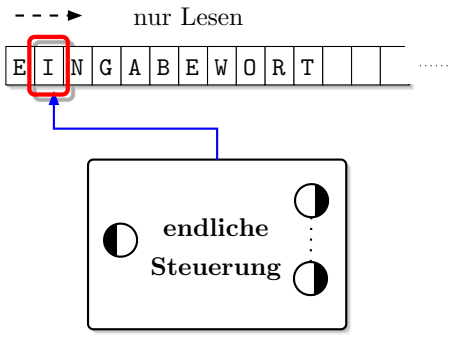


Abbildung 5.1: Grundschema des deterministischen endlichen Automaten

x beschriftete Kante $(z_1) \xrightarrow{x} (z_2)$ vom Knoten z_1 zum Knoten z_2 wird genau dann gezeichnet, wenn $\delta(z_1, x) = z_2$ gilt.

Startzustände werden als (z) und Endzustände als (z) gezeichnet.[1]

Einen DFA kann man sich als Maschine vorstellen, die ein Eingabeband besitzt auf dem in einzelne Felder jeweils Symbole $x_i \in \Sigma$ geschrieben sind, die in ihrer Folge von links nach rechts jenes Wort $w = x_1 x_2 \dots x_n$ bilden, das als Eingabe des DFA dient. Um diese Eingabe verarbeiten zu können, besitzt der DFA einen Lesekopf, der auf dem Eingabeband jeweils ein Feld besuchen kann, dessen Beschriftung liest. Der DFA wechselt in Abhängigkeit vom gelesenen Symbol und dem momentanen Zustand in den durch die Übergangsfunktion δ bestimmten Folgezustand. Der DFA startet in seinem Startzustand z_0, liest das Eingabewort von links nach rechts in der beschriebenen Weise und akzeptiert es, wenn er nach Lesen des letzten Symbols des Eingabewortes in einem Endzustand ist.

Diese informale Beschreibung wird nun mathematisch präziser, also formal, formuliert.

5.2 Definition (akzeptierte Sprache $L(A)$)
Zu der Überführungsfunktion $\delta : Z \times \Sigma \longrightarrow Z$ eines gegebenen DFA $A := (Z, \Sigma, \delta, z_0, Z_{end})$ definieren wir die ***erweiterte Überführungsfunktion*** *$\widehat{\delta} : Z \times \Sigma^* \longrightarrow Z$ rekursiv, indem wir für alle Zustände $z \in Z$, alle Symbole $x \in \Sigma$ und alle Wörter $w \in \Sigma^*$ folgendes definieren:*

$$\begin{aligned} \widehat{\delta}(z, \lambda) &:= z \\ \widehat{\delta}(z, xw) &:= \widehat{\delta}(\delta(z, x), w) \end{aligned}$$

Die von dem DFA A ***akzeptierte Sprache*** *ist die Menge*

$$L(A) := \{w \in \Sigma^* \mid \widehat{\delta}(z_0, w) \in Z_{end}\}.$$

Zwei verschiedene DFA A und B heißen genau dann ***äquivalent****, wenn sie die gleiche Sprache akzeptieren, d.h. $L(A) = L(B)$ ist.*

Offensichtlich kann ein DFA auch die leere Menge $\emptyset$ akzeptieren. Dazu muss nur die Menge Z_{end} der Endzustände als leere Menge definiert werden, was ja nicht ausgeschlossen wurde.

Die Funktion $\widehat{\delta}$ ist tatsächlich eine Erweiterung der Funktion δ, denn für alle Zustände $z \in Z$ und alle Symbole $x \in \Sigma$ gilt ja $\widehat{\delta}(z, x) = \delta(z, x)$. Ferner gilt offensichtlich

$$\widehat{\delta}(z, x_1 x_2 \cdots x_n) = \delta(\cdots \delta(\delta(z, x_1), x_2) \cdots, x_n).$$

Es ist üblich, anstelle der erweiterten Funtion $\widehat{\delta}$ etwas unpräzise nur δ zu notieren, sofern dadurch keine Verwirrung entsteht. Auch wir wollen diese Notation künftig verwenden.

[1] Weiterhin gebräuchlich ist die Darstellung von Endzuständen durch einen Kreis mit doppelter Umrandung. Startzustände werden dann als Kreis dargestellt, auf den ein kleiner Pfeil zeigt.

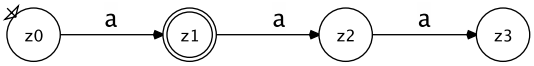
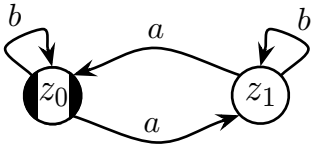


Abbildung 5.2: Ein DFA



	a	b
z_0	z_1	z_0
z_1	z_0	z_1

Abbildung 5.3: Beispiel-DFA

5.3 Beispiel

Betrachten wir den DFA A aus Abbildung 5.2. Akzeptiert dieser das Wort $w = a^3 = aaa$? Die Antwort lautet "Nein", denn obwohl der DFA während der Verarbeitung *durch* einen Endzustand kommt, ist der *erreichte* Zustand $\widehat{\delta}(z_0, aaa) = z_3$ kein Endzustand.

5.4 Beispiel

Betrachten wir den Zustandsgraphen aus Abbildung 5.3. Er stellt einen DFA dar, der die Menge aller Wörter $w \in \{a, b\}^*$ erkennt, die eine gerade Anzahl von Symbolen a enthalten. Auch das leere Wort λ ist ein solches Wort und wird von dem Automaten akzeptiert, ohne den Startzustand zu verlassen. Dies ist möglich, da der Startzustand gleichzeitig ein Endzustand ist.

> Bedenken Sie bei der Betrachtung eines endlichen Automaten immer genau, ob das Akzeptieren des leeren Wortes λ gewünscht ist oder nicht!

5.5 Beispiel

Der DFA B aus Abb. 5.4 besitzt die Sprache $L(B) = L := \{aba\}^*$. Die Mengengleicheit zeigen wir, indem wir zwei Inklusionen zeigen:

1. $L(B) \subseteq L$ (D.h. jedes von B akzeptierte Wort ist in L.)
2. $L \subseteq L(B)$ (D.h. jedes Wort in L wird von B akzeptiert.)

1. $L(B) \subseteq L$: Sei $w \in L(B)$. Wir müssen $w \in L$ zeigen.

   Nach Definition gilt $w \in L(B)$ gdw. $\widehat{\delta}(z_0, w) \in Z_{\text{end}}$, d.h. hier: $\delta^*(z_0, w) = z_0$, da $Z_{\text{end}} = \{z_0\}$.

   Die einzige Möglichkeit, von z_0 nach z_0 zu gelangen, besteht darin, beliebig oft (also n-mal für ein $n \in \mathbb{N}$) den Kreis $z_0 z_1 z_2$ zu durchlaufen.

   Entlang des Kreises $z_0 z_1 z_2$ lesen wir das Wort aba. Durchlaufen wir den Kreis n-mal, dann haben wir also das Wort $w = (aba)^n$ gelesen. Da $(aba)^n \in \{aba\}^* = L$ für jedes n gilt, haben wir $w \in L$ gezeigt.

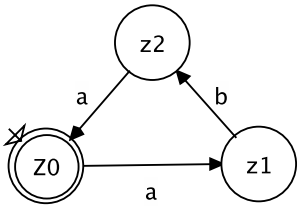


Abbildung 5.4: Ein DFA B mit der Sprache $L(B) = L := \{aba\}^*$.

2. $L \subseteq L(B)$: Sei $w \in L$. Wir müssen $w \in L(B)$ zeigen.

   Wenn $w \in L = \{aba\}^*$ ist, dann muss es ein $n \in \mathbb{N}$ geben, so dass $w = (aba)^n$ ist. Dieses Wort durchläuft folgende Zustände:

$$z_0 \underbrace{\xrightarrow{a} z_1 \xrightarrow{b} z_2 \xrightarrow{a} z_0}_{} \xrightarrow{a} \cdots z_0 \underbrace{\xrightarrow{a} z_1 \xrightarrow{b} z_2 \xrightarrow{a} z_0}_{} \in Z_{\mathsf{end}}$$

   Also gilt für jedes n auch $w = (aba)^n \in L(B)$. Also: $L \subseteq L(B)$.

5.1.1 Vollständige und initial zusammenhängende DFA

Neben der allgemeinsten Form des deterministischen endlichen Automaten (DFA) nach Def. 5.1 gibt es spezielle Modelle, die besonderen Einschränkungen unterworfen sind. Zum Beispiel war in der Definition des DFA nicht gefordert worden, dass es mindestens einen Endzustand geben muss, oder zu jedem Eingabesymbol in jedem Zustand auch ein definierter Übergang existiert: Weder die Überführungsfunktion δ, noch folglich deren Erweiterung $\widehat{\delta}$, musste eine totale Funktion sein!

5.6 Definition
Ein DFA $A := (Z, \Sigma, \delta, z_0, Z_{end})$ heißt:

1. ***vollständig** (Abk.: vDFA) genau dann, wenn für jedes $(p, x) \in Z \times \Sigma$ ein Zustand $q \in Z$ existiert, so dass $q = \delta(p, x)$ ist.*

   Mit anderen Worten: Bei einem vollständigen DFA ist die Überführungsfunktion total.

2. ***initial zusammenhängend** (Abk.: izDFA) genau dann, wenn gilt: Zu jedem Zustand $p \in Z$ existiert ein Wort $w \in \Sigma^*$, so dass $p = \widehat{\delta}(z_0, w)$ gilt.*

Zu einem gegebenen vDFA A schreiben wir $(z)^w$ für den Zustand $\widehat{\delta}(z, w)$, der auf eindeutige bestimmte Weise von z aus bei Eingabe von w erreicht wird.

Die Frage, ob es zu jedem DFA einen äquivalenten vollständigen DFA oder einen äquivalenten initial zusammenhängenden DFA gibt, werden wir in Kürze beantworten – und zwar positiv.

Wr haben in Definition 4.8 Klassen von formalen Sprachen zu sogenannte Sprachfamilien zusammengefasst. Wir definieren nun die durch DFA beschriebene Sprachfamilie der regulären Sprachen.

5.7 Definition ($\mathcal{R}eg$, $\mathcal{A}kz(\Sigma)$)
*Die von einem DFA akzeptierte Menge von Wörtern, nennt man **reguläre Menge**. Die Familie aller reguläre Mengen bezeichnet man mit $\mathcal{R}eg$.*

Mit $\mathcal{A}kz(\Sigma)$ wird die Familie aller jener Sprachen $L \subseteq \Sigma^*$ bezeichnet, die von deterministischen endlichen Automaten mit dem Eingabe-Alphabet Σ akzeptiert werden können. Mithin gilt:

$$\mathcal{A}kz(\Sigma) = \{L \subseteq \Sigma^* \mid L = L(A) \text{ für einen DFA } A\}$$

und

$$\mathcal{R}eg = \bigcup_{\substack{\Sigma \text{ ist endl.} \\ \text{Alphabet}}} \mathcal{A}kz(\Sigma).$$

Reguläre Mengen sind solche Sprachen, die von beliebigen DFA's erkannt werden. Bislang wissen wir noch nicht, ob die Beschränkung auf vDFA oder izDFA (Definition 5.6) vielleicht das Akzeptieren einiger regulärer Mengen verhindert. Dass dem zum Glück nicht so ist, werden wir im Verlauf der nächsten Untersuchungen noch feststellen (vgl. Theorem 5.18).

5.1.2 Konstruktion der initialen Zusammenhangskomponente eines vDFA

Natürlich kann man getrost alle diejenigen Komponenten eines vDFA weglassen, die nicht vom Startzustand aus erreicht werden können. Entsprechend muss dann die Übergangsfunktion auf die, eventuell kleinere, neue Zustandsmenge eingeschränkt werden. An der endlichen Darstellung des Zustandsgraphen kann man dies sofort sehen, falls der vDFA eine überblickbare Größe besitzt. Wir wollen hier ein einfaches, wenn auch nicht das effizienteste, Verfahren vorstellen, wie man in dem Zustandsgraphen eines, nicht notwendig vollständigen, DFA die initial zusammenhängende Komponente bestimmen kann.

5.8 Algorithmus (initiale Zusammenhangskomponente eines vDFA)
Sei $A := (Z, \Sigma, \delta, z_0, Z_{\mathsf{end}})$ ein beliebiger DFA. Wir berechnen schrittweise die induktiv definierten Mengen $M_i \subseteq Z$, beginnend bei M_0, gemäß:

$$M_i := \begin{cases} M_{i-1} \cup \bigcup\limits_{\substack{z \in M_{i-1} \\ x \in \Sigma}} \delta(z,x) & \text{für } i > 0 \\ \{z_0\} & \text{für } i = 0 \end{cases}$$

Offensichtlich ist jeder Zustand $z \in M_i$, $i \in \mathbb{N}$, mit einer passenden Eingabe $w \in \Sigma^*$, mit $|w| \leq i$, vom Startzustand aus erreichbar. Wegen $M_{i-1} \subseteq M_i \subseteq Z$ und der Endlichkeit der Zustandsmenge Z muss es einen Index $k \leq |Z|$ geben, für den $M_{k+1} = M_k$ ist (mit jeder Erhöhung des Indexes i um 1 kann möglicherweise nur ein weiterer Zustand in die Menge M_i aufgenommen werden).

Das Konstruktionsverfahren endet, wenn das erste mal $M_k = M_{k+1}$ wird, was spätestens bei $k = |Z| - 1$ der Fall sein wird. Die Menge M_k enthält bei Abschluss des Verfahrens also genau die von z_0 aus erreichbaren Zustände im DFA A.

5.2 Nichtdeterministische endliche Automaten

Syntaxdiagramme zu Programmiersprachen wie zum Beispiel PASCAL oder MODULA geben an, auf welche Weise ein korrekter Ausdruck entsprechend der Definition geschrieben werden kann. Dies Vorgehen ist wahlfrei oder nichtdeterministisch in dem Sinne, dass alles erlaubt ist, was nicht verboten wurde.

5.9 Beispiel
In Abbildung 5.5 wird die Syntax für Dezimalzahlen durch ein rekursionsfreies Syntaxdiagramm angegeben. Die Interpretation dieses Diagramms geschieht auf folgende Weise: Links beginnend dürfen die Kanten, den Pfeilen folgend, gegangen werden, wobei die in den Kreisen (oder Ovalen) stehenden Symbole aufgeschrieben werden sollen, und am Ende das Diagramm rechts verlassen werden muss. An den Abzweigungen kann jeder beliebige Weg gewählt werden, und jede so notierbare Zeichenkette ist eine in der betreffenden Programmiersprache erlaubte Dezimalzahl, wobei lediglich eine eventuelle Obergrenze in der Zahl der Ziffern mit dem Syntaxdiagramm nicht formuliert wurde.

Aufmerksame Betrachter(innen) finden zum Beispiel für die Ziffernfolge 2000 mehr als nur einen Weg durch obiges Diagramm: Die kleine Rückwärtsschleife mit der Ziffer „0“ kann nach Besuchen des Knotens ② dreimal oder nur zweimal durchlaufen werden. In letzterem Fall muss die dritte Null der Ziffernfolge 2000 – nach Benutzen des oberen Rückwegs nach ganz links – dann über den unteren Weg gewählt werden.

Ganz diesem Beispiel entsprechend, kann man sich einen endlichen nichtdeterministischen Automaten als einen bewerteten, gerichteten Graphen vorstellen. Dieser besitzt extra ausgezeichnete Start- und Zielknoten, und beschreibt dann die Menge aller jener Zeichenketten, die durch Aneinanderketten aller Kantenbewertungen entstehen können, die entlang gerichteter Kanten auf einem Weg von einem beliebigen Startknoten zu einem ebenso beliebigen Zielknoten zu finden sind. So wie ein Taxifahrer in einer Großstadt im allgemeinen mehrere Möglichkeiten hat, seinen Kunden zum Ziel zu fahren, gibt es in solch einem gerichteten Graphen ebenfalls mehrere, oft beliebig viele, Möglichkeiten von einem Start- zu einem Zielknoten zu gelangen. In vielen Fällen vereinfachen sich die Definitionen von formalen Sprachen auf diese Weise erheblich, auch wenn damit die anschaulichere Vorstellung vom endlichen Automaten als einer real existierenden Maschine, aufgegeben werden muss.

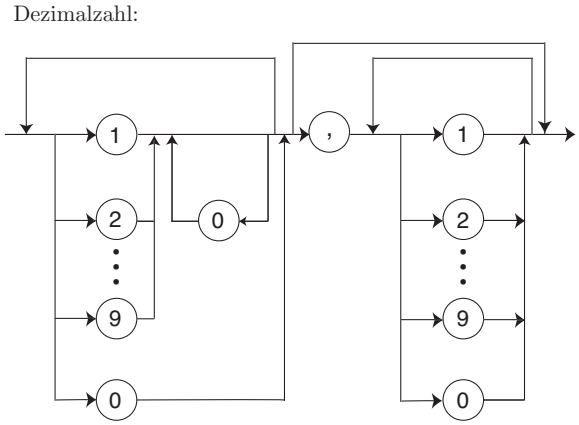


Abbildung 5.5: Syntaxdiagramme

5.2.1 Nichtdeterministischer, endlicher Automat

Eine andere Vorstellung sieht einen nichtdeterministischen endlichen Automaten lediglich als eine Erweiterung des bekannten DFA. In dieser Erweiterung gibt die Überführungsfunktion bei Eingabe eines Symbols in einem Zustand möglicherweise mehrere Folgezustände an. Beide Vorstellungen haben ihre Vorteile und wir geben beide Definitionsvarianten zum Vergleich an, werden danach aber die „graphenorientierte“ Definition 5.12 bevorzugen.

Ein nichtdeterministischer, endlicher Automat wird **NFA** abgekürzt (engl. ***n**ondeterministic **f**inite **a**utomaton*).

5.10 Definition

*Ein **nichtdeterministischer, endlicher Automat** (**NFA**) wird durch ein Tupel $A := (Z, \Sigma, \Delta, Z_{start}, Z_{end})$ spezifiziert:*

- *Z ist eine endliche Menge von **Zuständen**.*
- *Σ ist ein endliches Alphabet von **Eingabesymbolen**.*
- *$\Delta : Z \times \Sigma \longrightarrow 2^Z$ ist eine Abbildung, die **Überführungsfunktion** genannt wird.*
- *$Z_{start} \subseteq Z$ ist die **Menge der Startzustände**.*
- *$Z_{end} \subseteq Z$ is die **Menge der Endzustände**.*

Die Überführungsfunktion Δ ist beim NFA stets eine totale Funktion, weil dort, wo die Überführungsfunktion eines DFA undefiniert wäre, hier als Bild die leere Mange $\emptyset$ gewählt werden kann.

Wie beim DFA auch, wird die Abbildung Δ auf Wörter erweitert. Dies geschieht hier aber etwas anders, weil wir es hier mit Zustandsmengen zu tun haben.

Die erweiterte Überführungsfunktion $\widehat{\Delta} : 2^Z \times \Sigma^* \longrightarrow 2^Z$ wird für alle Wörter $w \in \Sigma^*$, alle Symbole $x \in \Sigma$ und jede Teilmenge $Z' \subseteq Z$ definiert durch:

$$\begin{aligned} \widehat{\Delta}(Z', \lambda) &:= Z' \\ \widehat{\Delta}(Z', xw) &:= \widehat{\Delta}\left(\bigcup_{z \in Z'} \Delta(z, x), w\right) \end{aligned}$$

$L(A) := \{w \in \Sigma^* \mid \widehat{\Delta}(Z_{start}, w) \cap Z_{end} \neq \emptyset\}$ ist die von A akzeptierte Sprache.

*Zwei verschiedene NFA A und B heißen genau dann **äquivalent**, wenn sie die gleiche Sprache akzeptieren, d.h. $L(A) = L(B)$ ist.*

Anmerkung: In der Literatur wird die Überführungsfunktion $\Delta : Z \times \Sigma \longrightarrow 2^Z$ meist auch nur mit dem Kleinbuchstaben δ bezeichnet. Wir wollen dies auch so machen, es sei denn wir wollen die Notationen des NFA von denen des DFA abgrenzen.

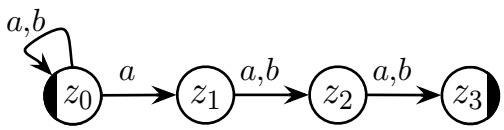


Abbildung 5.6: NFA, der alle Wörter akzeptiert, die an drittletzter Position ein a haben

5.11 Beispiel

In Abbildung 5.6 ist der Zustandsgraph eines NFA dargestellt, der genau diejenigen Wörter $w \in \{a,b\}^*$ akzeptiert, die an drittletzter Position das Symbol a enthalten. Dieser endliche Automat akzeptiert also z.B. das Wort $abaaba$, jedoch *nicht* das Wort $abaabaa$.

Beispielhaft berechnen wir $\hat{\Delta}(\{z_0, z_1\}, ab)$:

$$\begin{aligned}
\hat{\Delta}(\{z_0,z_1\},ab) &= \widehat{\Delta}\left(\bigcup_{z\in\{z_0,z_1\}} \Delta(z,a), b\right)\\
&= \widehat{\Delta}\left(\Delta(z_0,a)\cup\Delta(z_1,a), b\right)\\
&= \widehat{\Delta}\left(\{z_0,z_1\}\cup\{z_2\}, b\right)\\
&= \widehat{\Delta}\left(\{z_0,z_1,z_2\}, b\right)\\
&= \widehat{\Delta}\left(\bigcup_{z\in\{z_0,z_1,z_2\}} \Delta(z,b), \lambda\right)\\
&= \widehat{\Delta}\left(\Delta(z_0,b)\cup\Delta(z_1,b)\cup\Delta(z_2,b), \lambda\right)\\
&= \widehat{\Delta}\left(\{z_0\}\cup\{z_2\}\cup\{z_3\}, \lambda\right)\\
&= \widehat{\Delta}\left(\{z_0,z_2,z_3\}, \lambda\right)\\
&= \{z_0,z_2,z_3\}
\end{aligned}$$

Jeder DFA $A := (Z, \Sigma, \delta, z_0, Z_{\text{end}})$ kann auch als ein (spezieller) NFA $B := (Z, \Sigma, \Delta, Z_{\text{start}}, Z_{\text{end}})$ angesehen werden. Wir definieren dazu:

$$\Delta(z,x) = \{\delta(z,x)\} \quad \text{und} \quad Z_{\text{start}} = \{z_0\}$$

Wir werden daher im folgenden jedem NFA $A := (Z, \Sigma, \Delta, Z_{\text{start}}, Z_{\text{end}})$, bei dem für alle $z \in Z$ und $x \in \Sigma$ auch $|\Delta(z,x)| \leq 1$ sowie $|Z_{\text{start}}| = 1$ gilt, auch als deterministischen Automaten bezeichnen.

5.2.2 Der Zustandsgraph eines Endlichen Automaten

Manchmal ist es praktischer den NFA nicht durch die Überführungsfunktion δ, sondern durch die Menge der Kanten im Zustandsgraphen zu definieren. Die folgende Definitionsvariante orientiert sich mehr an dem Zustandsgraphen als an der funktionalen Sichtweise mit einer Überführungsfunktion. Die Kanten werden nun als Relation definiert (vergl. Def. 3.13)

5.12 Definition

Ein ***nichtdeterministischer, endlicher Automat*** *(in der am Zustandsgraphen orientierten Form) ist ein Tupel $A := (Z, \Sigma, K, Z_{start}, Z_{end})$, wobei die einzelnen Größen wie folgt erklärt sind:*

Z ist eine endliche Menge von Zuständen.

$Z_{start} \subseteq Z$ ist die Menge der Startzustände.

$Z_{end} \subseteq Z$ ist die Menge der Endzustände.

Σ ist ein endliches Alphabet von Eingabesymbolen.

$K \subseteq Z \times \Sigma \times Z$ ist die Relation der Zustandsübergänge.

Die Darstellung können leicht ineinander überführt werden. Zu einen gilt: Wenn der NFA als Tupel $A := (Z, \Sigma, \delta, Z_{\text{start}}, Z_{\text{end}})$ gegeben ist, dann erhalten wir die Kantenrelation $K_\delta \subseteq Z \times \Sigma \times Z$ wie folgt:

$$K_\delta := \{(p,x,q) \mid p \in Z, x \in \Sigma, q \in \delta(p,x)\}$$

Umgekehrt gilt: Wenn der NFA als Tupel $A := (Z, \Sigma, K, Z_{\text{start}}, Z_{\text{end}})$ gegeben ist, dann erhalten wir die Überführungsfunktion $\Delta_K : Z \times \Sigma \longrightarrow 2^Z$ wie folgt:

$$\Delta_K(p, x) := \{q \in Z \mid (p, x, q) \in K\}$$

Aus diesem Grunde betrachten wir beide Darstellungen im folgenden als austauschbar.

Ebenso kann jeder DFA $A := (Z, \Sigma, \delta, z_0, Z_{\text{end}})$ auch als ein (spezieller) NFA $A' := (Z, \Sigma, K, Z_{\text{start}}, Z_{\text{end}})$ angesehen werden, indem wir die Kantenmenge wie folgt definieren:

$$K := \{(p, x, \delta(p, x)) \mid p \in Z, x \in \Sigma, \delta(p, x) \text{ ist def. }\}$$

Einen speziellen nichtdeterministischen endlichen Automaten A kann man natürlich immer durch explizite Angabe der fünf Mengen $Z, \Sigma, K, Z_{\text{start}}$ und Z_{end} notieren, besser ist aber meist auch hier die Angabe des Zustandsgraphen, der ähnlich erklärt wird wie beim DFA. Bei kleinen Automaten ist dies in der Regel auch übersichtlicher.

5.2.3 Darstellung von Graphen (Exkurs)

Da wir die Zustandsgraphen von endlichen Automaten nicht nur als einfache oder bequeme Darstellungen ansehen wollen, sondern bei deren Benutzung auch Analysen vornehmen wollen, werden wir die hier benötigten mathematischen Begriffe einführen und erläutern. Wir schließen dabei an die mathematischen Grundbegriffe aus Kapitel 3 an.

Relationen bilden ein wichtiges und anschauliches Hilfsmittel, nicht nur bei der Darstellung von endlichen Automaten, sondern auch bei der Beschreibung und Formulierung von anderen Modellen oder Algorithmen. Jede binäre Relation lässt sich als gerichteter Graph auffassen. Algorithmen für Graphen können umgekehrt auch oft bei Fragen zu Relationen sinnvoll genutzt werden.

5.13 Definition

Sei R eine binäre Relation auf $A \times B$. Der ***Graph*** *$G(R)$ der Relation ist der* ***gerichtete Graph*** *(directed graph) $G(R) := (V, E)$ mit der Menge der* ***Knoten*** *(Ecken, vertices) $V := A \cup B$ und der Menge $E := R \subseteq V \times V$ der* ***gerichteten Kanten*** *(edges). Die Kante $(a, b) \in E$, notiert als $a \longrightarrow b$, wird also genau dann gezeichnet, wenn $(a, b) \in R$ gilt.*

Notiert wird ein gerichteter Graph $G = (V, E)$ oft durch seine ***Adjazenzmatrix*** *$C_G \in \{0, 1\}^{A \times B}$. Dies ist eine Matrix, deren Zeilen, mit den Elementen des Vorbereichs von R, also A, und deren Spalten, mit den Elementen des Nachbereichs von R, also B, bezeichnet werden. Die einzelnen Matrizenelemente $C_G(x, y)$ sind definiert durch:*

$$\forall x \in A : \forall y \in B : C_G(x, y) := [(x, y) \in E].$$

Mit dieser Notation kann die Komposition von Relationen (vergl. Def. 3.13) durch ein modifiziertes Produkt $A \otimes B$ der Adjazenzmatrizen gebildet werden. Wir benutzen das gewöhnliche Matrizenprodukt, nur interpretieren wir die verwendeten Operationen Summe und Multiplikation anders!

5.14 Definition

Seien $A \in \{0, 1\}^{m \times n}$ und $M_2 \in \{0, 1\}^{n \times r}$ spezielle Matrizen über $\mathbb{Z}$, die wir in diesem Zusammenhang als ***boolesche Matrizen*** *bezeichnen wollen, da die Werte 0 und 1 in ihren Komponenten wie boolesche Wahrheitswerte benutzt werden sollen.*

Das ***boolesche Matrizenprodukt*** *von A und B ist für jedes Element $C(i, j)$, $1 \leq i \leq m$, $1 \leq j \leq r$ für $C := A \otimes B$ definiert durch:*

$$C(i, j) := \max\left\{ \min\left\{ A(i, k), B(k, j) \right\} \;\middle|\; 1 \leq k \leq n \right\}$$

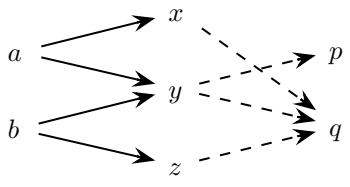


Abbildung 5.7: Die Graphen von R_1 (durchgehend) und R_2 (gestrichelt) aus Beispiel 5.15

Statt $x+y$ in $\mathbb{R}$ bilden wir also $\max\{x,y\}$ und anstelle von $x \cdot y$ in $\mathbb{R}$ bilden wir $min\{x,y\}$. Interpretiert man die Zahlen 0 und 1 als Wahrheitswerte (1 für *wahr* und 0 für *falsch*), so entspricht die modifizierte Summe (max) über den Werten 0 und 1 nun gerade dem logischen Junktor $\vee$ (*„oder“*) und das Produkt (min) dem logischen Junktor $\wedge$ (*„und“*). Daher heißt diese Multiplikation eben auch boolesches Matrizenprodukt. Es passt zur Definition der Adjazenzmatrix $C_{G(R)}$ zum Graph der Relation R, mit

$$C_{G(R)}(x,y) = [(x,y) \in R] = \begin{cases} 1, & \text{falls } (x,y) \in R \\ 0, & \text{falls } (x,y) \notin R. \end{cases}$$

Mit dieser Darstellung erhalten wir für die Komposition $R_1 \odot R_2$ der Relationen $R_1 \subseteq A \times B$ und $R_2 \subseteq B \times C$ dann $C_{G(R_1)} \otimes C_{G(R_2)} = C_{G(R_1 \odot R_2)}$.

Ein kleines Beispiel soll dies veranschaulichen:

5.15 Beispiel

Seien $A := \{a,b\}$, $B := \{x,y,z\}$, $C := \{p,q\}$ und $R_1 \subseteq A \times B$, $R_2 \subseteq B \times C$ gegeben durch $R_1 := \{(a,x),(a,y),(b,y),(b,z)\}$ und $R_2 := \{(x,q),(y,p),(y,q),(z,q)\}$. Die Graphen der Relationen sind in Abbildung 5.7 dargestellt.

Mit den Adjazenzmatrizen $C_{G(R_1)} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$ und $C_{G(R_2)} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \\ 0 & 1 \end{pmatrix}$ errechnet man leicht, dass deren boolesches Produkt gerade die Adjazenzmatrix $C_{G(R_1 \odot R_2)} = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$ zum Graphen der Relationskomposition $R_1 \odot R_2$ ergibt.

Wichtig bei dieser Auffassung ist, dass der Graph einer Relation keine doppelten Kanten zwischen den gleichen Punkten besitzt, denn eine Relation ist eine Menge von Paaren, und kann diese nicht mehrfach enthalten. Das wird erst möglich, wenn wir die einzelnen Kanten von einander unterscheiden können, was durch deren **Beschriftung** oder **Bewertung** geschieht. In nachfolgender Definition 5.16 wird ausdrücklich eine beliebige Menge X von möglichen Bewertungen oder Kantenbeschriftungen gestattet, damit wir diese Kanten bewerteten Graphen möglichst flexibel handhaben können.

5.16 Definition

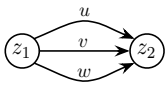
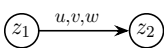
*Sei $K \subseteq A \times X \times B$ eine Relation „von A nach B“ mit Kantenbewertungen aus der Menge X. Der **kantenbewertete gerichtete Graph** zu K ist spezifiziert durch das Tripel $G_K := (V, X, K)$ mit Knotenmenge $V := A \cup B$. Eine bewertete Kante $(a,x,b) \in K$ von G_K wird als $a \xrightarrow{x} b$ gezeichnet.*

Wie beim DFA ist auch die Knotenmenge des Zustandsgraphen eines NFA isomorph zu seiner Zustandsmenge und jeder Knoten wird wieder als Kreis um die Zustandsbezeichnung gezeichnet. Start- und Endzustände werden in der gleichen Weise dargestellt wie beim Zustandsgraphen eines DFA's. Eine formale Definition können wir nun mit Hilfe des Graphen zu der Relation K eines NFA geben:

5.17 Definition

Sei $A := (Z, \Sigma, K, Z_{start}, Z_{end})$ ein NFA, dann ist der Zustandsgraph zu A der Graph $G_K := (Z, \Sigma, K)$.

Da von einem Zustand p in einem nichtdeterministischen endlichen Automaten nun mehrere verschiedene Kanten mit unterschiedlichen Symbolen als Beschriftung zum selben Zustand q führen können, wird eine vereinfachte Darstellung bevorzugt:

Sind zum Beispiel die Kanten $(z_1, u, z_2), (z_1, v, z_2)$ und (z_1, w, z_2) in K enthalten, so notieren wir statt  abkürzend  .

5.3 Äquivalenz von DFA und NFA

Da jeder DFA ein spezieller NFA ist, kann natürlich jede reguläre Menge von einem NFA akzeptiert werden. Zu fragen ist jedoch, ob es Sprachen gibt, die ein NFA akzeptieren kann, die nicht regulär sind und wie überhaupt gezeigt werden kann, ob es überhaupt eine formale Sprache gibt, die nicht regulär ist und wie man dieses eventuell beweisen könnte. Wir werden dies Antwort hier nicht schuldig bleiben, und sogar mehrere, jedoch sicher nicht alle, Möglichkeiten zu deren Beantwortung darstellen.

Das folgende Ergebnis über akzeptierende endliche Automaten ist sehr wichtig, da es deutlich macht, dass mit der Definition eines *nichtdeterministischen* gegenüber der des *deterministischen* endlichen Automaten, keine größere Klasse von definierbaren Sprachen entsteht.

5.18 Theorem

Jede von einem NFA akzeptierte Menge kann auch von einem initial zusammenhängenden, vollständigen DFA akzeptiert werden und ist daher regulär.

Beweis: Sei $A := (Z, \Sigma, K, Z_{\text{start}}, Z_{\text{end}})$ ein NFA. Wir konstruieren den sog. **Potenzautomaten** zu A – einen zu A äquivalenten, vollständigen DFA B.

Der Potenzautomat „merkt“ sich im Zustandsnamen, in welchen Zuständen der nichtdeterministische endliche Automat sich nach derselben Sequenz von gelesenen Symbolen befinden könnte. Er akzeptiert ein Wort genau dann, wenn der ursprüngliche NFA mit diesem Wort in einnen Endzustand hätte gelangen können, d.h. genau dann, wenn es im ursprünglichen Automaten einen Erfolgspfad für dieses Wort gibt.

Die Zustände des Potenzautomaten B erhalten als Namen gerade die Teilmengen von Z. Die Zustandmenge von B ist also gerade die Potenzmenge 2^Z – daher der Name.[2] Dieser vDFA B werde wie folgt erklärt: $B := (2^Z, \Sigma, \delta, z_0, F)$ mit $F := \{M \in 2^Z \mid M \cap Z_{\text{end}} \neq \emptyset\}$ und $z_0 := Z_{\text{start}}$, d.h. die Menge der Startzustände von A bildet nun den einzigen Startzustand z_0 von B. Die Überführungsabbildung δ ist für alle $M \in 2^Z$ und jedes $x \in \Sigma$ definiert durch:

$$\delta(M, x) := \bigcup_{z \in M} \{z' \in Z \mid (z, x, z') \in K\}.$$

Um die Korrektheit der Konstruktion zu zeigen, müssen wir beweisen, dass der Potenzautomat B äquivalent zu A ist. Herzstück der Argumentation ist die folgende Eigenschaft (Der Leser möge diese durch Induktion über w beweisen.)

$$\text{Für jedes Wort } w \in \Sigma^* \text{ gilt } \widehat{\Delta}(Z_{\text{start}}, w) = \widehat{\delta}(z_0, w).$$

Wir skizzieren nun einen Beweis der Korrektheit dieser Konstruktion:

$L(B) \subseteq L(A)$: Zu jeder akzeptierenden Rechnung des vDFA B auf einem Wort w gibt es wegen der Definition von δ ebenfalls einen Erfolgspfad von A auf w.

[2] Wer damit besser umgehen kann, mag sich die Zustandsmenge so vorstellen, als stünde die Teilmenge $Q \subseteq Z$ für den Zustand z_Q einer Menge $\{z_P \mid P \subseteq Z\}$.

$L(A) \subseteq L(B)$: Andererseits findet sich auch für jeden nichtdeterministisch ausgewählten Erfolgspfad von A eine akzeptierenden Rechnung in B. Die in A besuchten Zustände kommen dabei in den einzelnen Zuständen des Pfades von B, die ja selbst Teilmengen von Z sind, in der gleichen Reihenfolge vor.

Also sind A und B äquivalent. □

5.19 Beispiel

Wir konstruieren mit der Potenzautomatenkonstruktion zu dem in Abbildung 5.8 gezeichneten NFA $A := (Z, \Sigma, K, Z_{\mathsf{start}}, Z_{\mathsf{end}})$ einen äquivalenten vDFA B. Für den gegebenen NFA A seien die Bestandteile erklärt durch:

$$\begin{aligned} Z &:= \{z_0, z_1, z_2\} \\ \Sigma &:= \{a, b\} \\ K &:= \{(z_0, a, z_0), (z_0, b, z_0), (z_0, a, z_1), (z_1, a, z_2), (z_1, b, z_2)\} \\ Z_{\mathsf{start}} &:= \{z_0\} \\ Z_{\mathsf{end}} &:= \{z_2\} \end{aligned}$$

Der Zustandsgraph von A ist in Abbildung 5.8 dargestellt.

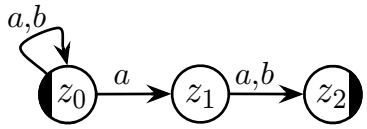


Abbildung 5.8: Der NFA akzeptiert alle Wörter, die an vorletzter Stelle ein a haben

Nach Konstruktion sind die Bestandteile von $B := (Z', \Sigma, \delta, \{z_0\}, F)$ gegeben durch:

$$\begin{aligned} Z' &:= 2^Z = \Big\{\emptyset, \{z_0\}, \{z_1\}, \{z_2\}, \{z_0, z_1\}, \{z_0, z_2\}, \{z_1, z_2\}, \{z_0, z_1, z_2\}\Big\}, \\ \Sigma &:= \{a, b\}, \\ F &:= \Big\{\{z_2\}, \{z_0, z_2\}, \{z_1, z_2\}, \{z_0, z_1, z_2\}\Big\} \end{aligned}$$

Die graphische Darstellung des Potenzautomaten B und δ sind dem (nicht zusammenhängenden) Zustandsgraphen aus Abbildung 5.9 zu entnehmen.

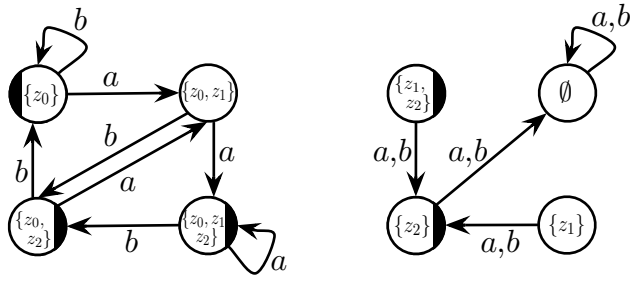


Abbildung 5.9: Potenzautomat zu dem NFA aus Abbildung 5.8

Wie man in Abbildung 5.9 sehen kann, muss der Potenzautomat nicht initial zusammenhängend sein. Der rechte Teil der Abbildung zeigt einen vom Startzustand aus nicht erreichbaren Teilgraphen, der folglich für die Akzeptierung von Wörtern nicht relevant ist.

Wenn wir in Zukunft vom Potenzautomaten sprechen, meinen wir üblicherweise nur den initial zusammenhängenden Teil des Potenzautomaten nach dem Beweis von Theorem 5.18.

5.3.1 Ist die Potenzautomatenkonstruktion optimal?

Nach der gegebenen Konstruktion des Potenzautomaten, hat der äquivalente DFA exponentiell mehr Zustände als der zu Beginn bekannte NFA. Dass dies keine Schwäche dieser Konstruktion ist, sondern in einigen Fällen tatsächlich soviele Zustände für den DFA nötig sind, zeigt das folgende Resultat.

Wir beweisen zunächst das folgende Theorem. Es gibt eine Sprache, die ein NFA mit $n+1$ Zuständen akzeptieren kann, aber der kleinste äquivalente DFA hat 2^n Zustände – also fast die 2^{n+1} Zustände, die der Potenzautomat konstruieren würde.

5.20 Theorem

(i) Es gibt einen NFA mit $n+1$ Zuständen, der die Sprache $L_n := \{w \in \{a,b\}^* \mid$ das n-letzte Symbol in w ist ein $a\}$ akzeptiert.

(ii) Jeder DFA, der die Menge L_n akzeptiert, hat 2^n Zustände.

Beweis: Zu (i): Der NFA aus Abbildung 5.10 akzeptiert die Sprache L_n.

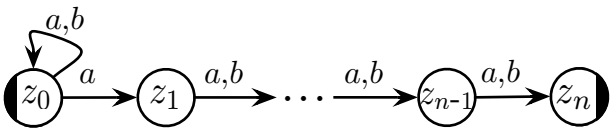


Abbildung 5.10: Ein NFA für die Sprache $L_n := \{w \in \{a,b\}^* \mid$ das n-letzte Symbol in w ist $a\}$

Zu (ii): Sei $B = (Z, \Sigma, \delta, z_0, Z_{\text{end}})$ ein DFA mit $L(B) = L_n$ und seien $w_1, w_2 \in \{a,b\}^n$ verschiedene Wörter. Wir zeigen $(z_0)^{w_1} \neq (z_0)^{w_2}$.

Da w_1 und w_2 verschieden sind, gibt es mindestens einen Buchstaben, an dem sich die beiden Wörter unterscheiden, d.h. es gibt Wörter $u_1, u_2, v \in \{a,b\}^*$, so dass o.B.d.A. $w_1 = u_1 a v$ und $w_2 = u_2 b v$ gilt.

Für $v' \in \{a,b\}^*$ mit $|v'| := n - |v| - 1$ gilt nun 1.): $w_1 v' \in L_n$ und 2.): $w_2 v' \notin L_n$, denn der n-letzte Buchstabe von $w_1 v' = u_1 a v v'$ ist ein a aber der von $u_2 b v v'$ nicht.

Wäre nun aber $(z_0)^{w_1} = (z_0)^{w_2} = z'$, so hieße dies doch $(z')^{v'} \in Z_{\text{end}}$ wegen 1.) und gleichzeitig auch $(z')^{v'} \notin Z_{\text{end}}$ wegen 2.). Dies ist aber ein Widerspruch, folglich kann die Annahme $(z_0)^{w_1} = (z_0)^{w_2}$ nicht richtig gewesen sein.

Da es 2^n paarweise verschiedene Wörter der Länge n in $\{a,b\}^*$ gibt, folgt die Existenz von ebenso vielen verschiedenen Zuständen im Automaten B. □

Man kann die Aussage sogar noch weiter verschärfen.

5.21 Theorem

Zu jeder natürlichen Zahl $n \geq 2$ gibt es einen NFA A_n mit genau n Zuständen, so dass jeder *äquivalente vollständige DFA mindestens $2^n - 1$ Zustände besitzt.*

Beweis: Die Automaten A_n haben die in Abb. 5.11 dargestellte Form.

Den Beweis, dass *jeder* äquivalente vollständige DFA mindestens $2^n - 1$ Zustände besitzen muss, wollen wir an dieser Stelle nicht führen.

□

5.4 Generalisierte endliche Automaten

Wenn wir NFAs zur Beschreibung regulärer Mengen benutzen, so müssen wir einen NFA, der genau das Wort $w \in \Sigma^*$ akzeptieren soll, mit $|w| + 1$ Zuständen modellieren, auch wenn diesem Wort in dem

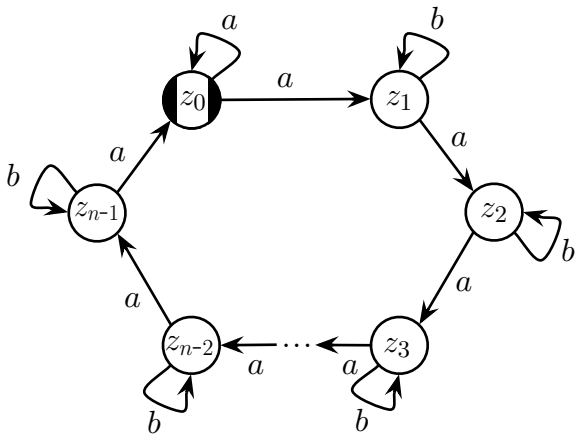


Abbildung 5.11: Der Automat A_n hat genau n Zustände.

Zustandsgraphen genau ein Pfad zugeordnet ist. Auch darf das leere Wort λ nicht als Kanteninschrift verwendet werden.

Wir verallgemeinern daher NFAs weiter. Die Kanten werden nun als Relation definiert (vergl. Def. 3.13), wobei an Kanten jetzt nicht nur Symbole, sondern sogar Zeichenketten stehen dürfen. Insbesondere kann dieser Automatentyp seinen Zustand wechseln, ohne dass er dazu etwas von der Eingabe lesen muss. Diese allgemeine Form des nichtdeterministischen endlichen Automaten wird als **GFA** abgekürzt (engl. ***g**eneralised **f**inite **a**utomaton*).

5.22 Definition (GFA)
*Ein **generalisierter endlicher Automat** ist ein Tupel $A := (Z, \Sigma, K, Z_{start}, Z_{end})$, wobei die einzelnen Größen wie folgt erklärt sind:*

Z ist eine endliche Menge von Zuständen.

$Z_{start} \subseteq Z$ ist die Menge der Startzustände.

$Z_{end} \subseteq Z$ ist die Menge der Endzustände.

Σ ist ein endliches Alphabet von Eingabesymbolen.

$K \subseteq Z \times \Sigma^* \times Z$ ist die endliche(!) Relation der Zustandsübergänge.

*Ein Element $(p, \lambda, q) \in K$ wird als **λ-Kante** bezeichnet.*

Die Einschränkung, dass $K \subseteq Z \times \Sigma^* \times Z$ endlich sein muuss, ist wichtig, damit wir weiterhin eine endliche Darstellung besitzen. Dies muss für K explizit gefordert werden, denn $Z \times \Sigma^* \times Z$ ist eine unendliche Menge, da Σ^* dies ist.

5.23 Beispiel
Die λ-Übergänge modellieren „interne“ Entscheidungen des Automaten. Abbildung 5.12 zeigt rechts den GFA C, der sich aus den beiden Automaten A und B zusammensetzt und von dem neuen Startzustand q_0 mit einer λ-Kante zu den vormaligen Startzuständen der Automaten A und B wechseln kann. Da der Automat mit einem λ-Übergang die Eingabe nicht verändert, liest er also entweder in A ein Wort oder in B, so dass C insgesamt die Vereinigung $L(A) \cup L(B)$ akzeptiert.

Um mit diesen Automaten formal arbeiten zu können, benötigen wir noch die Begriffe der *Rechnung* und der *Erfolgsrechnung* sowie den daraus resultierenden Begriff der *akzeptierten Sprache* eines GFA.

5.24 Definition
Sei $A := (Z, \Sigma, K, Z_{start}, Z_{end})$ ein GFA.

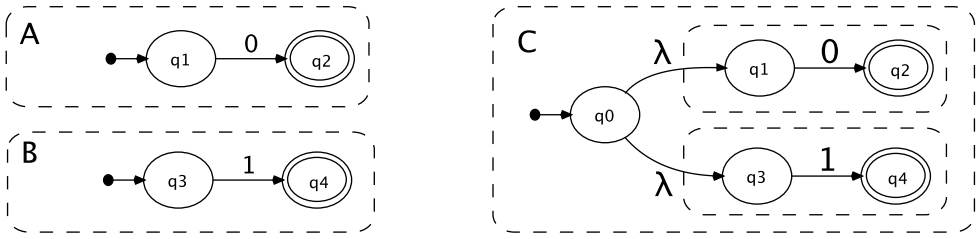


Abbildung 5.12: Ein GFA

Eine Kantenfolge $p := (z_0, x_1, z_1), (z_1, x_2, z_2), \ldots, (z_{n-1}, x_n, z_n)$ *mit* $(z_{i-1}, x_i, z_i) \in K$ *für alle* $0 < i \leq n$ *heißt* ***Rechnung*** *von* A *für das Wort* $|p| := w := x_1 x_2 \ldots x_n \in \Sigma^*$. *Dies wird kurz durch* $z_0 \xrightarrow[w]{*} z_n$ *notiert.*

Weiterhin gilt auch noch $z_0 \xrightarrow[w]{*} z_n$ *für* $w = \lambda$ *genau dann, wenn* $z_0 = z_n$ *gilt, d.h. mit dem leeren Wort erreicht man den Zustand selbst.*

Für jedes $w \in \Sigma^*$ *kann* $\xrightarrow[w]{*}$ *als eine Relation auf* Z *angesehen werden und stellt somit eine Teilmenge von* $Z \times Z$ *dar.*

Eine ***Erfolgsrechnung*** $z_0 \xrightarrow[w]{*} z_n$ *für das Wort* $w \in \Sigma$ *ist eine Rechnung, bei der* $z_0 \in Z_{start}$ *und* $z_n \in Z_{end}$ *ist.*

Hierbei ist auch $w = \lambda$ *möglich, nämlich, wenn* $z_0 = z_n$ *und* $z_0 \in Z_{start} \cap Z_{end}$.

$L(A) := \{w \in \Sigma^* \mid \exists z_0 \in Z_{start} : \exists z_n \in Z_{end} : z_0 \xrightarrow[w]{*} z_n\}$ *bezeichnet die von* A ***akzeptierte Sprache***.

Zwei GFA A *und* B *heißen genau dann* ***äquivalent****, wenn sie die gleiche Sprache akzeptieren, d.h.* $L(A) = L(B)$ *ist.*

Es werden also genau solche Wörter $w \in \Sigma^*$ akzeptiert, für die in dem Automaten A eine Erfolgsrechnung existiert. Das leere Wort wird – wie beim DFA – akzeptiert, wenn der Anfangs- auch ein Endzustand ist (Es ist also möglich, dass auch ein λ-freier NFA das leere Wort akzeptiert.) In einem GFA kann aber das leere Wort auch akzeptiert werden, wenn im Zustandsgraphen des GFA ein Pfad von einem Startzustand zu einem Endzustand existiert, der sich ausschließlich aus λ-Kanten zusammensetzt.

5.4.1 Teilklassen generalisierter Automaten

Automaten, die einer bestimmten Einschränkung genügen, sind in manchen Fällen einfacher zu benutzen, besonders bei Beweisen oder beim Programmieren von Algorithmen. Daher geben wir hier drei besondere Klassen an.

5.25 Definition
Ein GFA $A := (Z, \Sigma, K, Z_{start}, Z_{end})$ *heißt ...*

1. λ***-frei*** *genau dann, wenn* $K \subseteq Z \times \Sigma^+ \times Z$ *ist.*
2. ***fast-buchstabierend*** *genau dann, wenn* $K \subseteq Z \times (\{\lambda\} \cup \Sigma) \times Z$ *ist.*
3. ***buchstabierend*** *genau dann, wenn* $K \subseteq Z \times \Sigma \times Z$ *ist.*

Aus der Definition folgt, dass jeder buchstabierende Automat stets auch λ-frei ist.

Beachte: Ein buchstabierender GFA entspricht genau einem NFA nach Definition 5.10, weil die Überführungsfunktion $\delta : Z \times \Sigma \longrightarrow 2^Z$ genau das gleiche wie die Relation $K \subseteq Z \times \Sigma \times Z$ beschreibt.

In der Literatur wird ein fast-buchstabierender GFA auch als λ-FA bezeichnet.

5.4.2 Äquivalenz von generalisierten und nichtdeterministischen Automaten

Wir werden nun zeigen, dass wir jeden GFA stets in einen äquivalenten NFA umformen können. Um dieses Ergebnis auf einfache Weise beweisen zu können ist es hilfreich, vorher spezielle Umformungen an einem vorgegebenen beliebigen, endlichen Automaten vorzunehmen. So kann ein GFA eine oder mehrere λ-Kanten besitzen, die in deterministischen endlichen Automaten ja gar nicht erlaubt waren. Man kann diese nicht einfach entfernen, ohne die akzeptierte Sprache zu verändern, aber man kann einen beliebigen GFA zunächst derart umformen, dass er danach immer noch die gleiche Sprache akzeptiert jedoch keine λ-Kanten mehr besitzt, also λ-frei ist.

5.26 Theorem

Zu jedem GFA gibt es einen äquivalenten λ-freien GFA.

Beweis: Sei $A := (Z, \Sigma, K, Z_{\text{start}}, Z_{\text{end}})$ ein gegebener GFA, der möglicherweise λ-Kanten besitzt. Ein neuer GFA $B := (Z, \Sigma, K', Z_{\text{start}}, Z'_{\text{end}})$ ohne λ-Kanten wird wie folgt erklärt:

$$\begin{aligned} K' \quad := \quad & \{(z, w, z''') \in Z \times \Sigma^* \times Z \mid \exists (z', w, z'') \in K : \\ & w \neq \lambda \wedge z \xrightarrow[\lambda]{*} z' \wedge z'' \xrightarrow[\lambda]{*} z''' \text{ (in A)}\} \end{aligned}$$

und $Z'_{\text{end}} := Z_{\text{end}} \cup \{z \in Z_{\text{start}} \mid \exists z' \in Z_{\text{end}} : z \xrightarrow[\lambda]{*} z' \text{ (in A)}\}$.

Wir zeigen die Gleichheit von $L(A)$ und $L(B)$ durch den Beweis der gegenseitigen Inklusion:

$L(B) \subseteq L(A)$ gilt, weil (z, w, z') nur dann in K' ist, wenn im Zustandsgraphen von A ein Pfad von z nach z' existiert, auf dem das Wort w gelesen wird. Die Definition von Z'_{end} sichert, dass jeder Erfolgspfad in B auch einer in A ist.

$L(A) \subseteq L(B)$ gilt aus aus folgender Überlegung: Zunächst folgt aus $\lambda \in L(A)$ auch $\lambda \in L(B)$, denn dann ist einer der Startzustände von B auf Grund der Definition von Z'_{end} auch zugleich Endzustand.

Weiter ist klar, dass jede Kante $(z', w, z'') \in K$ mit $w \neq \lambda$ auch eine Kante von K' ist!

Nun lässt sich jeder Erfolgspfad p in A schreiben als:

$$z_1 \xrightarrow[\lambda]{*} z'_1 \xrightarrow[w_1]{} z_2 \xrightarrow[\lambda]{*} z'_2 \xrightarrow[w_2]{} z_3 \xrightarrow[\lambda]{*} \cdots \xrightarrow[w_{n-1}]{} z_n \xrightarrow[\lambda]{*} z'_n \xrightarrow[w_n]{} z_{n+1} \xrightarrow[\lambda]{*} z'_{n+1}$$

Hierbei ist $z'_i \xrightarrow[w_i]{} z_{i+1}$ gerade der Zustandsübergang mit einer Kante (z'_i, w_i, z_{i+1}) von A, mit $w_i \neq \lambda$.

Wenigstens eine solche Kante muss in einem Pfad p mit $|p| \neq \lambda$ ja vorkommen. Nach Definition von K' ist aber jetzt auch $(z_i, w_i, z_{i+1}) \in K'$ für jedes $1 \leq i < n$ sowie $(z_n, w_n, z'_{n+1}) \in K'$. Folglich gilt in dem Automaten B auch $z_1 \xrightarrow[|p|]{*} z'_{n+1}$, was wegen $z_1 \in Z_{\text{start}}$ und $z'_{n+1} \in Z'_{\text{end}}$ offensichtlich ein Erfolgspfad in B ist.

□

Obwohl der Beweis oben gegeben ist, können wir damit noch nicht ganz zufrieden sein, denn eigentlich wissen wir doch gar nicht, auf welche Weise wir den neuen, λ-freien Automaten nun im speziellen Fall *effektiv* konstruieren können. Effektiv ist die Konstruktion erst, wenn wir für je zwei beliebige Zustände z und z' feststellen können, ob die Beziehung $z \xrightarrow[\lambda]{*} z'$ gilt. Nun ist die Relation $\xrightarrow[\lambda]{*} \subseteq Z \times Z$ gerade die reflexive, transitive Hülle der Relation $\xrightarrow[\lambda]{} := \{(p, q) \mid (p, \lambda, q) \in K\}$. Die Relation $\xrightarrow[\lambda]{}$ kann sofort aus dem Zustandsgraphen von A abgelesen werden: Es sind genau die λ-Kanten dieses Automaten. Die reflexive, transitive Hülle $\xrightarrow[\lambda]{*}$ als Relation über Z kann bestimmt werden, da die Zustandsmenge

Z endlich ist. (Ein allgemeines Verfahren zur Bestimmung des reflexiven, transitiven Abschlusses gibt Abschnitt 5.4.3 an.)

Bis jetzt können wir also zu jedem GFA einen äquivalenten λ-freien GFA konstruieren, aber dieser ist i.a. noch nicht buchstabierend. Wir konstruieren jetzt diesen buchstabierenden GFA.

5.27 Theorem
Zu jedem λ-freien GFA gibt es einen äquivalenten buchstabierenden GFA.

Beweis: Zu jeder Kante $k := (z, w, z') \in K$ des GFA $A := (Z, \Sigma, K, Z_{\text{start}}, Z_{\text{end}})$ mit $|w| \geq 2$ werden $|w| - 1$ neue (Zwischen-)Zustände z_{k_i} definiert. Die Kante $k = (z, w, z')$ mit $w = x_1 x_2 \cdots x_n$ wird dann ersetzt durch die Kanten der Menge

$$\{(z, x_1, z_{k_1}), (z_{k_1}, x_2, z_{k_2}), \ldots, (z_{k_{n-2}}, x_{n-1}, z_{k_{n-1}}), (z_{k_{n-1}}, x_n, z')\}.$$

Die Äquivalenz ist offensichtlich. □

Nach diesen Umformungen erhält man also mit Sicherheit einen äquivalenten λ-freien, buchstabierenden GFA, also einen NFA.

5.28 Theorem
Jede von einem GFA akzeptierte Menge kann auch von einem NFA akzeptiert werden und ist daher regulär.

Beweis: Direkt aus Theorem 5.26 und 5.27. □

Der generierte NFA braucht aber nicht vollständig zu sein, da in manchen Zuständen nicht für jedes Eingabesymbol $x \in X$ eine herausführende Kante existieren muss. Will man auch dieses gewährleisten, so fügt man einen weiteren Zustand z_s als *Senke* hinzu, und ergänzt den Automaten durch vorher fehlende Kanten, die in diese Senke führen. Im Zustand z_s gibt es für jedes Symbol $x \in X$ eine Kante (z_s, x, z_s).

Bei der Umformung des NFA in einen deterministischen Automaten mit der Konstruktion des Potenzautomaten wird aber ohnehin ein vollständiger äquivalenter DFA entstehen, so dass dieses Verfahren nicht detallierter dargestellt werden braucht.

5.4.3 Effektive Bestimmung des des reflexiven, transitiven Abschlusses (Exkurs)

Zur effektiven Bestimmung des reflexiven, transitiven Abschlusses R^* einer endlichen zweistelligen Relation R (vergl. Def.3.19), also der Menge aller Wege zwischen den Knoten des gerichteter Graphen $G(R)$, taugt die Darstellung nach Theorem 3.28 natürlich nichts, denn es wäre ja ein unendlicher Durchschnitt zu bilden. Nach Theorem 3.29 wissen wir, dass es ausreicht von der Relation $\xrightarrow[\lambda]{}$ nur endlich viele Kompositionen $\xrightarrow[\lambda]{k}$ zu bilden um deren transitive Hülle $\xrightarrow[\lambda]{+}$ zu erhalten. Die dort benutzte Oberschranke ist jedoch etwas groß, und es stellt sich die Frage, ob es nicht bessere Verfahren gibt.

In dem gerichteten Graphen $G(R)$ beschreibt die reflexive transitive Hülle R^* von R alle gerichteten Pfade in diesem Graphen. Die Lösung dieser Frage ist auch an anderen Stellen von großem Interesse. Das Verfahren von Warshall ist ein Algorithmus zur Bestimmung des reflexiven, transitiven Abschlusses $\xrightarrow{*}$ einer endlichen zweistelligen Relation $\longrightarrow \subseteq A \times A$.

Wir benutzen an dieser Stelle eine konstruktive Variante von Theorem 3.29, die eine recht gute Komplexität besitzt.

Sei $R \subseteq A \times A$ eine endliche Relation auf der endlichen Menge $A, n := |A|$. (In der hier gesuchten Anwendung wird $R := \xrightarrow[\lambda]{}$ mit $A := Z$ sein). Mit den Definitionen von $R^{i+1} := R^i \cdot R$ und $R^0 := Id_A$

gilt nun

$$\begin{aligned} R^* &= \bigcup_{i\geq 0} R^i \\ &= \bigcup_{i=0}^{n-1} R^i = Id_A \cup R \cup R^2 \cup R^3 \cup \cdots \cup R^{n-1} \\ &= (Id_A \cup R)^{n-1}, \end{aligned}$$

wobei letztere Gleichung leicht mit Induktion zu verifizieren ist. Man beachte hierbei, dass mit R^i hier **nicht** das i-fache kartesische Produkt gemeint wurde sondern die i-fache Komposition! Warum der Exponent nicht größer als $n-1$ gewählt werden muss sieht man so ein:

Es gilt doch $(x, y) \in R^i$ genau dann, wenn es in dem Graphen $G(R)$ einen Pfad vom Knoten x zum Knoten y gibt. Wenn es aber solch eine Pfad gibt, so existiert immer auch einer mit höchsten $n = |A|$ verschiedenen Knoten, also mit höchstens $n-1$ Kanten. (Alle Schleifen können weggelassen werden, wodurch sich jeder Pfad sukzessive (rekursiv) verkürzen lässt.)

Also muss zur Bestimmung der Relation $\xrightarrow[\lambda]{*}$ nur $(Id_Z \cup \xrightarrow[\lambda]{})^{|Z|-1}$ gebildet werden. Für diese Aufgabe gibt es eine sehr effiziente Methode, die linear in der Größe $|A|$ arbeitet.