Endliche Automaten

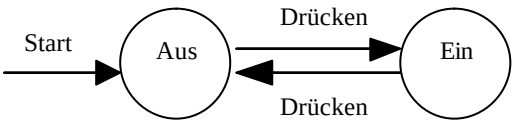
Stoyan Mutafchiev

Programming Systems Lab, Universität des Saarlandes, Saarbrücken

Abstract Gegenstand dieser Arbeit ist der endliche Automat, sowie die Abschlusseigenschaften der Sprachen, die von endlichen Automaten beschrieben werden können. Der endliche Automat wird einführend vorgestellt. Als nächstes werden die Fragen, die in Bezug auf endlichen Automaten auftreten können, ausgeschildert. Anschließend wird der Tupel-Automat, Projektion und Zylindrifikation der Sprache vorgestellt.

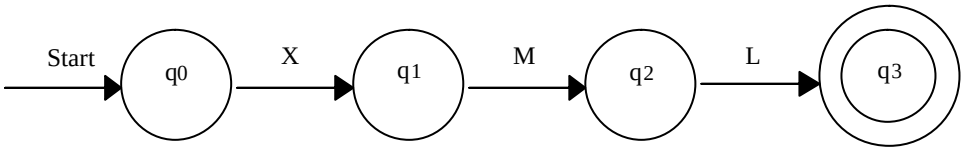
1. Einführung in den endlichen Automaten

Ein Kippschalter ist vielleicht der einfachste nicht triviale endliche Automat.



Dieses Gerät weiß, wann es sich im Zustand *Ein* oder *Aus* befindet und ermöglicht dem Benutzer, einen Schalter zu drücken, der, abhängig vom Zustand des Kippschalters, eine unterschiedliche Wirkung hat. Wenn sich der Kippschalter im Zustand *Aus* befindet, dann wird er durch das Drücken des Schalters in den Zustand *Ein* versetzt und umgekehrt. Die Zustände werden durch die Kreise repräsentiert (hier Ein und Aus genannt).Die Pfeile zwischen den Zuständen sind die "Eingaben"(im Beispiel Drücken). Einer der Zuständen wird als "Anfangzustand" ausgewählt d.h. der Zustand in dem sich das System ursprunglich befindet (in dem Beispiel ist *Aus* der Startzustand). Es ist oft notwendig, ein oder mehrere Zustände als "end" oder "akzeptierende" Zustände zu kennzeichnen. Die Endzustände werden durch die Doppelkreise repräsentiert. Wird nach eine Reihe von Eingaben in einem Endzustand gewechselt, dann heißt es, dass die Eingabesequenz in irgendeine Weise gültig ist, oder anders ausgedrückt, sie wird akzeptiert.

Das nächste Beispiel zeigt einen anderen endlichen Automaten, der das Schlüsselwort *XML* zu erkennen hat.



Als Eingaben werden Buchstaben empfangen. Der Endzustand q_3 wird erreicht, wenn die Eingabe dem Word *xml* genau entspricht. Wie bereits erwähnt, besitzt ein endlicher Automat eine Menge von Zuständen und seine „Steuerung“ geht in Reaktion auf externe „Eingaben“ von einem Zustand in einen anderen über. Der Unterschied zwischen endlichen Automaten besteht darin, ob diese Steuerung „deterministisch“, was bedeutet, dass der Automat zu einem bestimmten Zeitpunkt nur einen Zustand haben kann, oder ob sie „nichtdeterministisch“ ist, das heißt, dass

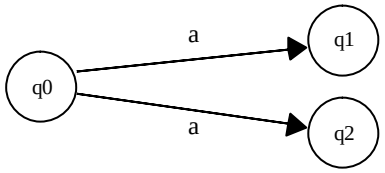
der Automat gleichzeitig mehrere Zustände besitzen kann. Die deterministischen endlichen Automaten werden wir als eine Teilmenge von nichtdeterministischen endlichen Automaten betrachten.

2. Nichtdeterministische endliche Automaten(NEA)

Ein nichtdeterministischer endlicher Automat ist ein 5-Tupel *(Q, S,* δ *, S, F)*, mit:

- *Q* ist eine endliche Menge von Zuständen.
- *S* ist eine endliche Menge von *Eingabesymbolen* (Eingabealphabet).
- $\delta : Q \times \Sigma \cup \{ \varepsilon \} \rightarrow 2^Q$ ist Übergangsfunktion , der ein Zustand aus *Q* und ein Eingabesymbol aus *S* oder leeres Wort ε als Argumente bekommt, aber eine Menge von null, einem oder mehrere Zustände zurückgibt, die Teilmenge von *Q* ist.
- *S* ist eine Menge von *Startzuständen*, die Teilmenge von *Q* ist.
- *F* ist eine Menge von *Endzuständen*, die Teilmenge von *Q* ist.

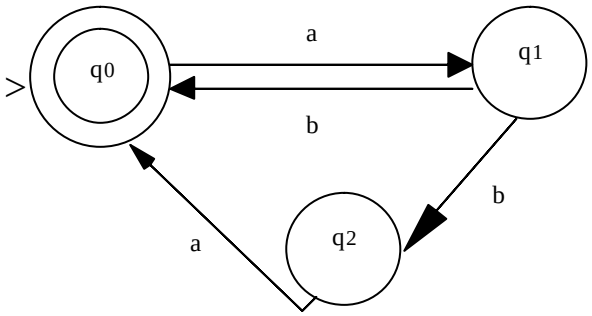
Bei nichtdeterministischen endlichen Automaten ist es zugelassen, dass vom selben Zustand $q \in Q$ aus mehrere (oder auch gar keine) Pfeile ausgehen, die mit $a \in \Sigma$ beschriftet sind.



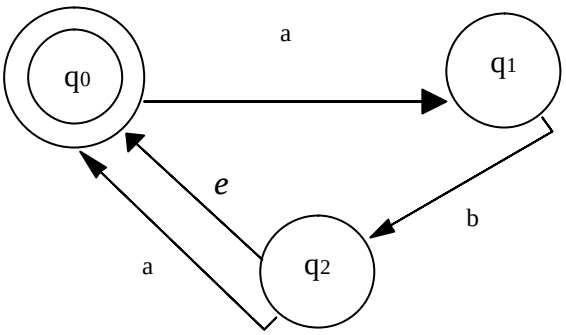
Wie „funktioniert " das überhaupt? Wenn im Zustand q_0 das Zeichen *a* gelesen wird, so *kann* der Automat sowohl in q_1 als auch in q_2 übergehen. Beide Möglichkeiten werden gleichwertig behandelt. Eine Zeichenreihe $w \in \Sigma^*$ wird vom Automaten akzeptiert, wenn er beim lesen von w eine Zustandsfolge durchlaufen *kann*, die von Startzustand auf einen Endzustand führt. Andere mögliche Zustandfolgen können durchaus auf Nicht-Endzustände führen. Wenn es aber *zumindest* eine akzeptierende Zustandsfolge gibt, dann wird das Wort *akzeptiert*.

Die Sprache eines nichtdeterministischen endlichen Automaten ist die Menge aller Zeichenreihen $w \in \Sigma^*$, die von dem Automaten akzeptiert werden. Die Sprachen, die mit einem endlichen Automaten beschrieben werden können, nennt man regulär.

Der nächste Beispiel zeigt einen nichtdeterministischen Automaten, der die Sprache L(A) = (ab È aba)* beschreibt.



Bei nichtdeterministischen endlichen Automaten sind die Übergänge für die leere Zeichenreihe e zugelassen. Dies bedeutet im Endeffekt, dass ein NEA spontan in einen anderen Zustand übergehen kann, ohne ein Eingabesymbol empfangen zu haben. Die Sprache von dem vorherigen Beispiel wird von dem folgenden nichtdeterministischen endlichen Automat mit e Übergang beschrieben:



3. Deterministische endliche Automaten (DEA)

Die deterministische endliche Automaten, oder DEA, betrachten wir als eine Teilmenge von nichtdeterministischen endlichen Automaten. Ein deterministischer endlicher Automat besteht aus:

- einer endlichen Menge von *Zuständen* Q

- einer endlichen Menge von *Eingabesymbolen* Σ

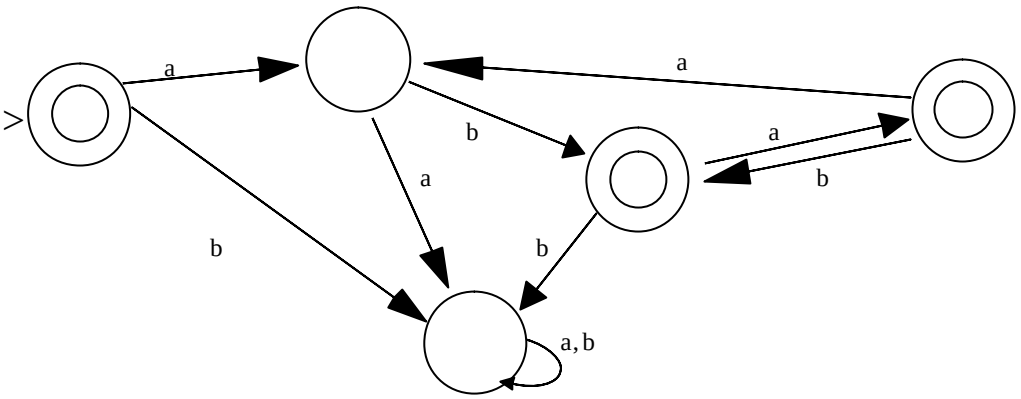
- einer *Übergangsfunktion* d :Q ´ Σ → Q, der ein Zustand und ein Eingabesymbol als Argumente übergeben werden und genau einen Zustand zurückgibt

- nur einem *Startzustand* meistens bezeichnet mit q_0 (nicht wie bei den nichtdeterministischen endlichen Automaten eine Menge von Startzustände)

- einer Menge *Endzustände* F

Bei deterministischen endlichen Automaten sind die e Übergänge nicht zugelassen. Die Sprache des DEA besteht aus der Menge aller Zeichenreihen, die der DEA akzeptiert. Wie berechnet ein DEA seine Eingabe? Nehmen wir an, $a_1a_2...a_n$ ist eine Folge von Eingabesymbolen und der Automat befindet sich in seinem Startzustand. Wir ermitteln mit Hilfe der Übergangsfunktion d den Zustand $d(q_0, a_1) = q_1$, in der der DEA nach der Verarbeitung des Eingabesymbols a_1 wechselt. Wir verarbeiten das nächste Eingabesymbol a_2 indem wir $d(q_1, a_2)$ berechnen, und nehmen wir an, dass diese Auswertung den Zustand q_2 ergibt. Wir fahren auf diese Weise fort und ermitteln die Zustände $q_3, q_4...q_n$, wobei für jedes i gilt $d(q_{i-1}, a_i) = q_i$. Wenn q_n ein Element von F ist, dann wird die Eingabe $a_1a_2...a_n$ akzeptiert, andernfalls wird sie „zurückgewiesen“. Die Sprache eines DEAs ist die Menge aller Zeichenreihen, die vom Startzustand in einen Endzustand führen.

Beispiel für DEA, der die schon von vorher bekannte Sprache L(A) = (ab È aba)* beschreibt



4. Äquivalenz deterministischen und nichtdeterministischen endlichen Automaten

Zwei Automaten bezeichnen wir als äquivalent, wenn sie die gleiche Sprache akzeptieren.

Satz. (Rabin, Scott)

Jede von einem NEA akzeptierbare Sprache ist auch durch einen DEA akzeptierbar.

Dies bedeutet, dass zu jedem NEA kann man äquivalenter DEA konstruieren. Man kann so genante "Teilmengekonstruktion" verwenden. Sie beginnt mit einem NEA $N = (Q_N, \Sigma, d_N, q_0, F_N)$. Das Ziel hierbei ist einen DEA ***D*** zu beschreiben sodass L(D) = L(N). Die Idee funktioniert auf den folgenden Art und Weise: Angenommen, N befindet sich in seinem Startzustand q_0 und die Eingabewort ist $w = a_1 ... a_m$. Wir verarbeiten das erste Eingabesymbol a_1 indem wir $d_N(q_0, a_1)$ berechnen. Weil der Automat nichtdeterministisch ist, transformiert a_1 den Zustand q_0 in der Menge Z_1 von Zuständen aus Q_N, oder auch in gar keinen Zustand. Nach der Berechnung von a_1 kann sich der Automat in jeden Zustand aus Z_1 befinden. Angenommen, besteht Z_1 aus *k* Zuständen: $s_1, ..., s_k$. Danach liest *N* das nächste Eingabesymbol a_2. Angenommen, ergibt die Berechnung von $d_N(s_1, a_2)$ die Menge von Zuständen S_1, die Berechnung von $d_N(s_2, a_2)$ die Menge von Zuständen S_2 und so weiter. Also nach dem Lesen von dem zweiten Eingabesymbol a_2 von der Zeichenreihe *w* kann sich N in jedem Zustand von $Q_2 = S_1 \cup ... \cup S_k$ befinden. Fahren wir so mit der Berechnungen fort, nach dem lesen von dem letzten Eingabesymbol a_m werden wir *m* Zustandsmengen $Q_1 ... Q_m$ haben, die man als Zustände von D bezeichnen kann.

Also nach dem wir das Gefühl bekommen haben, wie der gewünschte DEA eine Zeichenreihe berechnen wird, können wir den äquivalenten DEA wie folgt definieren:

$D = (Q_D, \Sigma, d_D, \{q_0\}, F_D)$ mit

- Q_D ist die Zustandmenge von D und ist die Potenzmenge von Q_N . Wenn Q_N n Zustände enthält, dann umfasst Q_D 2^n Zustände (also die Anzahl der Zustände nimmt exponentiell zu). Häufig sind nicht alle Zustände vom Startzustand erreichbar. Man kann nicht erreichbare Zustände eliminieren und daher kann die tatsächliche Anzahl von Zuständen von D viel kleiner sein als 2^n.

- Σ ist endliche Alphabet. Die beiden Automaten verfügen über dieselben Eingabealphabete.

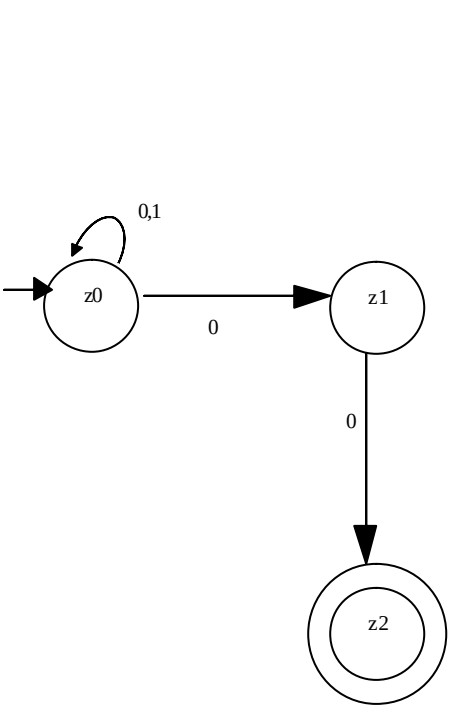
- {q0}- Startzustand . Bei dem Startzustand von D handelt sich um die Menge die den Startzustand von N enthält.

- F_D umfasst alle Teilmengen von Q_N , die mindestens einen akzeptierenden Zustand von N enthalten.

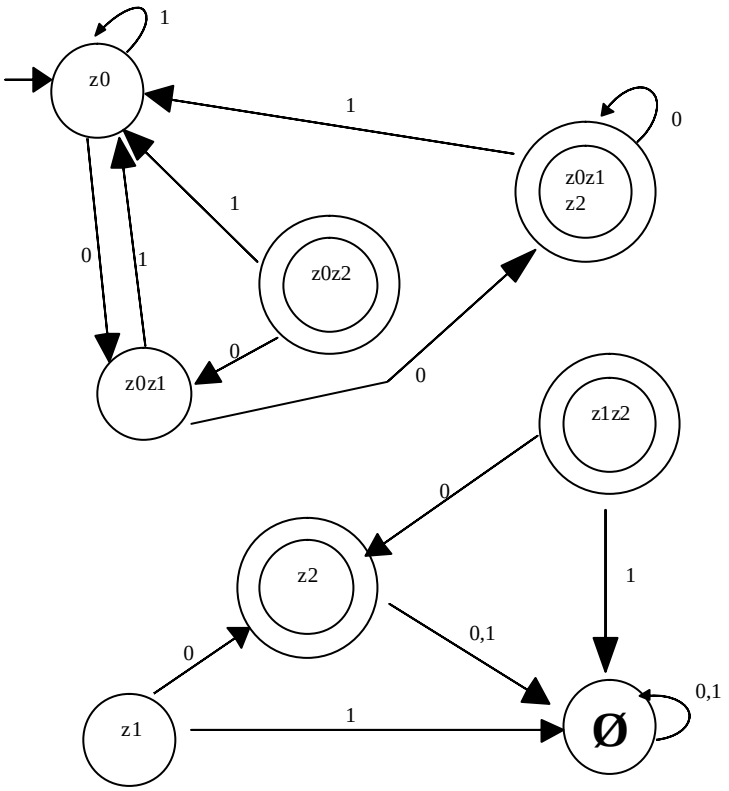
- d_D ist die Übergangsfunktion, die eine Teilmenge Q´ von Q_N und ein Eingabesymbol *a* aus Σ als Argumente bekommt :

$$d_D(Q', a) = \bigcup_{q \in Q'} d_N(q, a)$$

Also zur Berechnung von $d_D(Q', a)$ betrachten wir alle in Q´ enthaltenen Zustände q, untersuchen zu welchen Zuständen N auf die Eingabe a hin von p aus übergeht und nehmen die Vereinigung aller dieser Zustände. Bei NEA mit e Übergängen muss man die Übergangsfunktion mehrmals anwenden und die Zeichen konkatenieren.
<u>*Beispiel:*</u> Der folgende NEA akzeptiert genau die Wörter *w* über {0, 1}, die mit 00 enden.



NEA mit 3 Zustände



äquivalenter DEA mit 8 Zustände

Für diese NEA ergibt sich aus der Teilmengekonstruktion einen DEA mit 8 Zuständen:

$\emptyset, \{z_0\}, \{z_1\}, \{z_2\}, \{z_0, z_1\}, \{z_0, z_2\}, \{z_1, z_2\}, \{z_0, z_1, z_2\}$.

Die Endzustände des neuen DEA sind alle Zustände, die mindestens einen ursprünglichen Endzustand enthalten.

5. Abschlusseigenschaften regulärer Sprachen

Die Sprachen, die mit den endlichen Automaten beschrieben werden können, sind abgeschlossen unter Vereinigung, Komplement und Schnitt.

1. Seien L und M Sprachen über dem Alphabet Σ; dann ist $L \cup M$ die Sprache, die alle Zeichenreihen enthält, die in L oder in M oder in beiden Sprachen enthalten sind. Der Komplexität im Speicher der Konstruktion eines endlichen Automaten der die Sprache $L \cup M$ beschreibt ist.
2. Sei L eine Sprache über dem Alphabet Σ; dann ist $\overline{L}$, das Komplement von L, die Menge der in Σ^* enthaltenen Zeichenreihen, die nicht in L enthalten sind. Der Komplexität im Speicher der Konstruktion eines endlichen Automaten der die Sprache $\overline{L}$ beschreibt ist exponentiell.
3. Seien L und M Sprachen über dem Alphabet Σ; dann ist $L \cap M$ die Sprache, die alle Zeichenreihen enthält, die sowohl in L als auch in M enthalten sind. Der Komplexität im Speicher der Konstruktion eines endlichen Automaten der die Sprache $L \cap M$ beschreibt ist quadratisch.

5.1 Vereinigung

Um zu zeigen, das wenn L_1 und L_2 reguläre Sprachen sind dann ist auch $L_1 \cup L_2$ eine reguläre Sprache, brauchen wir einen EA zu konstruieren, der die Sprache $L_1 \cup L_2$ akzeptiert.

Sei $A_1 = (Q_1, \Sigma, d_1, S_1, F_1)$ akzeptiert die Sprache L_1 und $A_2 = (Q_2, \Sigma, d_2, S_2, F_2)$ akzeptiert die Sprache L_2. Dann bauen wir einen endlichen Automaten A, der in der Lage sein muss, die beide Automaten A_1 und A_2 zu simulieren, d. h. beim Eingabe *w*, A simuliert A_1. Wenn A_1 akzeptiert, dann akzeptiert auch A. Wenn A1 nicht akzeptiert, dann beginnt A A_2 zu simulieren und wieder A akzeptiert nur wenn A_2 akzeptiert, wenn nicht, A akzeptiert die Eingabe auch nicht. Diese Idee funktioniert wegen Nichtdeterminismus, weil die nichtdeterministische endliche Automaten mehrere Berechnungen auf eine Eingabe machen können. Der endlichen Automat A sieht so aus (wir annehmen, dass S1nS2 = 0):

$$A = (\{q_0\} \cup Q_1 \cup Q_2, \Sigma, d, q_0, F_1 \cup F_2) \text{ mit}$$

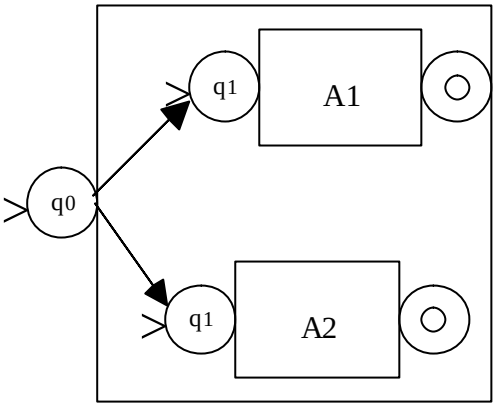
- die Zustandsmenge von A ist die Vereinigung von Zustandsmengen von A_1, A_2 und neuen Startzustand q_0

- q_0 ist der Startzustand von A, der mit e Übergänge mit der Startzustände von A_1 und A_2 verbunden ist

- $d = d_1 \cup d_2 \cup \{(q_0, e, q_1), (q_0, e, q_2)\}$ ist die Vereinigung von der Übergangsfunktionen von A_1, A_2 und die e Übergänge von dem Startzustand von A zu den Startzuständen von A_1 und A_2.

- die Menge von Endzustände von A ist einfach die Vereinigung von Endzustände von A_1 und A_2 .

Graphisch kann der endlichen Automat, der die Sprache $L_1 \bigcup L_2$ beschreibt, so dargestellt werden:



5.2 Komplement

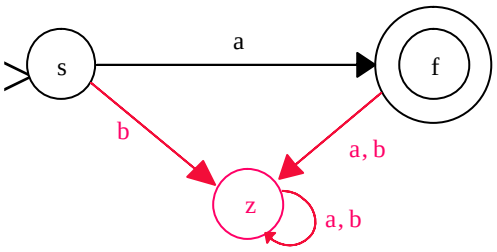
Wenn L eine reguläre Sprache über dem Alphabet Σ ist, dann ist $\overline{L} = \Sigma^* - L$ auch eine reguläre Sprache. Also sei $L = L(A)$ für einen endlichen Automaten $A = (Q, \Sigma, d, S, F)$. Falls A ein deterministischer endlicher Automat ist dann gilt $\overline{L} = L(B)$ wobei B für einen DEA $(Q, \Sigma, d, S, Q - F)$ steht. Dabei muss B vollständig sein. Vollständig bedeutet, dass bei B keine Übergänge fehlen .B unterscheidet sich dann von A nur dadurch, dass die akzeptierende Zuständen von A den nicht akzeptierenden Zuständen von B entsprechen und umgekehrt. Diese Konstruktion funktioniert nur im Fall das A ein DEA ist. Was ist aber wenn A ein nichtdeterministischer endlicher Automat wäre? Dann wird es bisschen komplizierter. Wie wir schon wissen bei NEA ist es zulässig, dass von manchen Zuständen keine Pfeile ausgehen. Es ist aber wichtig dass in A keine Übergänge fehlen. Wäre dies der Fall, dann könnten bestimmten Zeichenreihen weder zu einem akzeptierenden noch zu einem nicht akzeptierenden Zustand von A führen, und diese Zeichenreihen würden sowohl in L(A) als auch in L(B) fehlen. Damit alles korrekt funktioniert muss man:

- A vervollständigen, falls A nicht vollständig ist

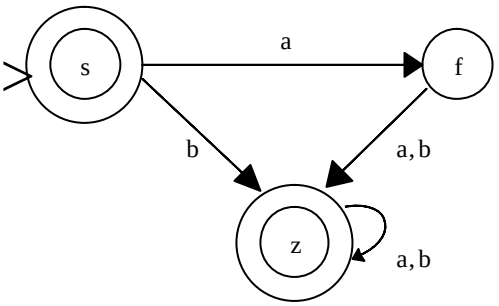
- A determinisieren, falls A ein NEA ist

- End- und Nichtendzustände jeweils vertauschen

In dem folgenden Beispiel wandeln wir einen DEA A, der nicht vollständig ist, in $\overline{A}$ um.



EA A vervollständigen



End- und Nichtendzustände jeweils vertauschen

2.3 Schnitt

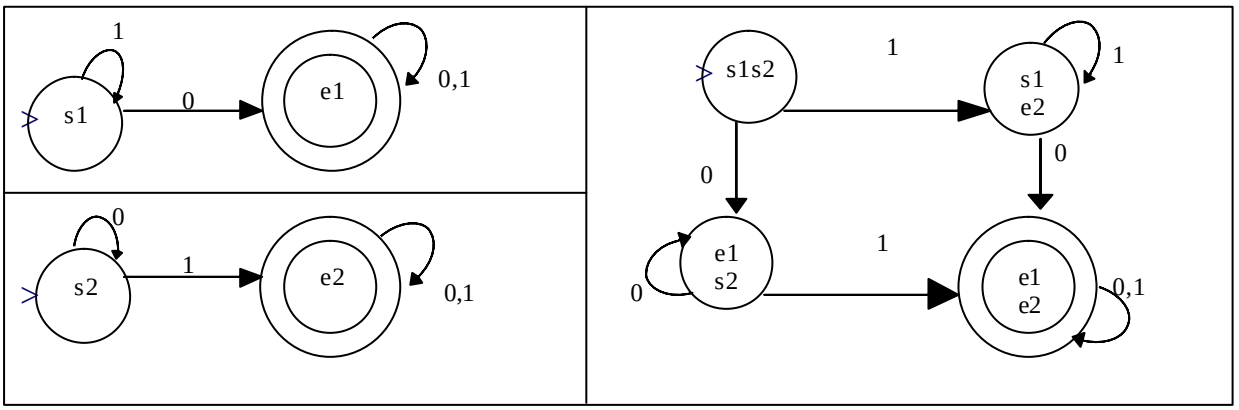
Wenn L und M reguläre Sprachen sind, dann ist auch $L \bigcap M$ eine reguläre Sprache. Das kann man zeigen, indem man einen EA baut, der die Sprache $L \bigcap M$ beschreibt.

Seien L und M die Sprachen der Automaten $A_L = (Q_L, \Sigma, d_L, q_L, F_L)$ und $A_M = (Q_M, \Sigma, d_M, q_M, F_M)$. Angenommen, haben die beiden Automaten identische Alphabete, d.h. Σ ist die Vereinigung der Alphabete von L und M, falls diese Alphabete verschieden sind. Für $L \bigcap M$ konstruieren wir einen EA A (bekannt auch als Produktautomat), der sowohl A_L als auch A_M simuliert. Bei der Zustände von A handelt es sich um Paare von Zuständen wobei die erste von A_L und die zweite A_M stammt. Angenommen, befindet sich im Zustand *(p, q),* wobei *p* der Zustand von A_L und *q* der Zustand von A_M ist. Sei *a* das Eingabesymbol. Dann beobachten wir wie A_L auf diese Eingabe reagiert. Nehmen wir an, dass A_L geht in Zustand *s* über. Wir können auch die Reaktion von A_M auf die Eingabe *a* ablesen. Nehmen wir an, A_M wechselt in den Zustand *t*. Der nächste Zustand von A ist also *(s, t)*. Auf diese Weise simuliert A die Arbeitsweise von A_L und A_M. Der Startzustand von entspricht natürlich dem Paar (q_L, q_M) der Startzustände von A_L und A_M. Da der Automat genau dann akzeptieren soll, wenn beide Automaten akzeptieren, wählen wir akzeptierende Zustände von A als Paare *(p, q)* so, dass *p* ein akzeptierender Zustand von A_L ist und *q* ein akzeptierender Zustand von A_M ist. Formal können wir den Produktautomat so definieren:

$$A = (Q_L \times Q_M, \Sigma, d, (q_L, q_M), F_L \times F_M)$$

wobei $d((p,q),a) = (d_L(p,a), d_M(q,a))$.

Das folgende Beispiel zeigt zwei EAs. Der erste akzeptiert alle Zeichenreihen, die eine Null enthalten. Der zweite akzeptiert alle Zeichenreihen, die eine Eins enthalten. Das Produkt dieser beiden Automaten akzeptiert den Durchschnitt von beiden: jene Zeichenreihen, die sowohl eine Null als auch eine Eins enthalten.



3 Fragen zu den endlichen Automaten

Einige grundlegende Fragen zu den Sprachen sind:

1. Ist die von gegebenen EA beschriebene Sprache leer?

2. Ist die von gegebene EA beschriebene Sprache Σ^*?

Das erste Problem liegt in P, d.h. eine Türing Maschine, die polynomiell viel Zeit benötigt.

Das zweite liegt in PSPACE, d.h. eine Türing Maschine, die polynomiell viel Speicher braucht.

3.1 Leerheit

Wenn man eine Sprache durch endliche Automaten repräsentiert, kann man die Frage, ob die Sprache leer ist, wie folgt umformen:

Gibt es irgendeinen Pfad vom Start- zum Endzustand?

Man kann die Frage beantworten, indem man mit dem Algorithmus aus der Graphentheorie die Menge der erreichbaren Zustände rekursiv berechnet:

Induktionsbeginn: Startzustand ist vom Startzustand erreichbar

Annahme: p von Startzustand aus erreichbar

Induktionsschritt: gibt es einen Übergang von p nach q mit Eingabesymbol oder e (wenn es sich um einen NEA handelt), dann ist q erreichbar.

Auf diese Weise kann man die Menge der erreichbaren Zustände berechnen .Wenn darunter ein akzeptierender Zustand ist, bedeutet dies, das die Sprache des Automaten nicht leer ist, andernfalls beantworten wir die Frage mit „Ja“. Die Berechnung der Erreichbarkeit benötigt einen Zeitaufwand von $O(n^2)$, wenn der Automat n Zustände hat.

3.2 Universalität

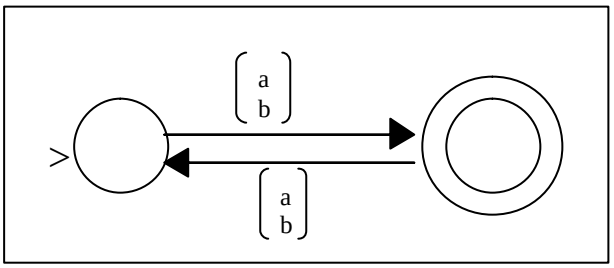
Frage: Ist die von gegebenen endlichen Automaten A beschriebene Sprache $L = \Sigma^*$?

Man kann die Frage beantworten, indem man nach einem Wort $w \in \overline{L(A)}$ sucht. Wenn man eine solche Zeichenreihe findet, bedeutet das, dass die von gegebenen endlichen Automat beschriebene Sprache nicht Σ^* ist. Das ist die Hauptidee des Algorithmus, den wir benutzen werden. Durch partielles Determinisieren kann man einen endlichen Automaten als einen Baum repräsentieren. Die Wurzel des Baumes ist der Startzustand des Automaten. Die Knoten sind seine Zustände und die Kanten, die mit Eingabesymbolen beschriften sind, sind die Übergänge.

Was uns bleibt ist zu suchen, ob ein Pfad existiert, der zu einem Knoten führt, der mit einem Nichtendzustand gezeichnet ist. Wenn wir einen solchen Pfad finden, bedeutet das, dass die Sprache des Automaten nicht Σ* ist, andernfalls geben wir dieser Frage einen positiven Antwort. Aber bis zu welcher Höhe muss man den Baum auffalten? Mann macht das bis zum bestimmte Höhe = $2^{|Q|}$, wobei Q die Anzahl der Zustände des Automaten ist. Das kommt von der Pumping Lemma, die besagt, dass wenn so ein Wort gibt, das von dem Automaten akzeptiert wird, dann wird auch ein Wort mit der Länge kleiner als die Anzahl der Zustände des Automaten akzeptiert. Also wenn ein Wort *w* von dem Automaten $\overline{A}$ akzeptiert wird, bedeutet das automatisch, dass dieses Wort von dem Automaten A nicht akzeptiert wird. Daraus folgt das die von A beschrieben Sprache nicht Σ* sein kann.

4 Tupel Automat

Die endlichen Automaten, die wir bis jetzt gesehen haben, bekamen nur eine Zeichenreihe als Eingabe. Man kann sich fragen, ob es möglich ist, das die Automaten zwei oder mehrere Wörter gleichzeitig als Eingabe bekommen und entscheiden, ob die Eingabe akzeptiert oder nicht akzeptiert wird. Hier kommen die Tupel Automaten im Einsatz. Ein Tupel Automat ist ein ganz normalen endlichen Automat, der aber gleichzeitig über Tupel von Wörtern operieren kann. Das folgende Beispiel zeigt einen 2-Tupel Automat über {a, b}:

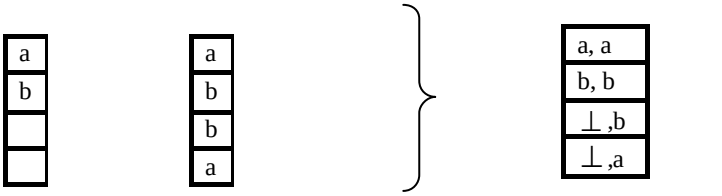


Damit ein Tupel Automat korrekt funktioniert, ist es notwendig, das er Wörter mit gleiche Länge als Eingabe bekommt. Das ist aber nicht immer der Fall. Wenn die Wörter unterschiedlichen Shape haben, muss man die kürzeren mit ⊥ liften (wir legen fest, das $\perp \notin \Sigma$). Auf diese Weise bekommt man Wörter mit der gleichen Länge.

Formal können wir ein k-Tupel Automat so definieren:

Ein k-Tupel Automat über S ist ein endlichen Automat über der Alphabet $(\Sigma \cup \{\perp\})^k$.

Ein Beispiel: Sei A_1 ein EA über Σ, der das Wort *w = ab* akzeptiert und A_2 ein EA über Σ, der das Wort *u= abba* akzeptiert. Dann die Eingabe des 2-Tupel Automaten A_T über $\Sigma_\perp$, der die beide Wörter akzeptieren würde (also (u, w)) sieht folgendermaßen aus:



Eingabe für A_1 | Eingabe für A_2 | geliftete Eingabe für A_T

5 Projektion und Zylindrifikation der Sprachen

Sei L die Sprache, die von einem *k*-Tupel Automaten beschrieben wird, also alle $w \in$ L sind *k*-stellige Tupel von Wörtern $(u_1,...,u_k)$ $für\ u_i \in \Sigma^*\ und\ 1 \le i \le k$.

Projektion und Zylindrifikation sind zwei Operationen über Sprachen, dessen Alphabet Σ ist von der Form: $\Sigma_1 \times ... \times \Sigma_k = \{(a_1,...,a_k) \mid a_1 \in \Sigma_1,...,a_k \in \Sigma_k\}$ wobei $\Sigma_1,...,\Sigma_k$ auch Alphabete sind. Jedes Wort aus Σ ist von der Form:

$$(s_1^1,...,s_1^k)...(s_n^1,...,s_n^k)\,.$$

Wir können aber auch die Zeichenreihen aus Σ als k-Tupel $(u_1,...,u_k)$ von Wörtern darstellen:

$$u_1 = s_1^1...s_n^1 \in \Sigma_1^*, \ldots,\ u_k = s_1^k...s_n^k \in \Sigma_k^*\,.$$

I-te Projektion der Sprache L über Σ definiert man als:

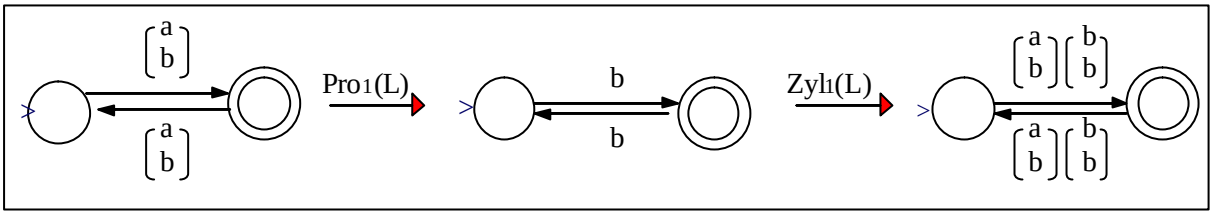
$$\mathrm{Pro}_i(L) = \{ \begin{pmatrix} u_1 \\ . \\ . \\ u_{k-1} \end{pmatrix} \in \Sigma_\perp^{k-1} \mid i \to \begin{pmatrix} u_1 \\ . \\ v \\ . \\ u_{k-1} \end{pmatrix} \in L \ \ für\ ein\ v \in \Sigma_i^* \}$$

d.h. *i*-te Projektion der Sprache L ist die Sprache, bei der das Wort $v \in \Sigma_i^*$, das auf der *i-ten* Stelle in dem Tupelwort steht, fehlt.

Bei der Zylindrifikation ist es umgekehrt. Statt zu entfernen, fügt man ein Wort aus Σ_i^* dem Tupelwort zu.
I-te Zylindrifikation der Sprache L über Σ kann man so definieren:

$$Zyl_i(L) = \{ i \to \begin{pmatrix} u_1 \\ . \\ v \\ . \\ u_{k-1} \end{pmatrix} \in L \ \ für\ ein\ v \in \Sigma_i^* \mid \begin{pmatrix} u_1 \\ . \\ . \\ u_{k-1} \end{pmatrix} \in \Sigma_\perp^{k-1} \}$$

Ein Beispiel für Projektion und Zylindrifikation über $\Sigma = \{a, b\}$:



Referenzen:

-Nerode A., Koussainov B.: *Automata Theory and its Applications.* Birkhäuser 2001

-John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman: *Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie.* Addison Wesley 1990

-Uwe Schöning: *Theoretische Informatik – kurzgefasst.* Spektrum Akademischer Verlag 1995

-Michael Sipser: *Introduction to the Theory of Computation.* Addison Wesley 1997