Automaten und formale Sprachen: Vorlesungsskript
G. Brewka, A. Nittka

Literatur:

- John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, Einführung in die Automatentheorie, Formale Sprachen und Komplexitätstheorie, 2. Auflage, Addison-Wesley, Pearson Studium, 2002, EUR 39,95
- Uwe Schöning, Theoretische Informatik - kurzgefasst, 4. Auflage, Spektrum Akademischer Verlag, 2001
- Alexander Asteroth, Christel Baier, Theoretische Informatik, Pearson Studium, 2002

1. Einführung

1.1 Was sind Automaten?

Beispiele: Geldautomat, Getränkeautomat, Fahrkartenautomat, ...
hier geht es nicht um konkrete Automaten und wie man sie baut, sondern um abstrakte Modelle von (Klassen von) Automaten und ihre prinzipiellen Eigenschaften

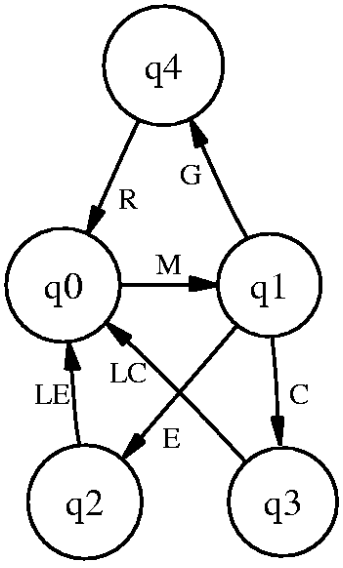
Was haben alle Automaten gemeinsam?

Möglichkeit der Eingabe
Zustände und Übergänge zwischen Zuständen, hervorgerufen durch Eingabe

Bsp.: durch Einwurf einer Münze in Kaffeeautomaten wird Zustand "bezahlt" erreicht. Drücken der Taste "Cappuccino" in diesem Zustand führt zu Nachfolgezustand, in dem Cappuccino ausgegeben wird, Drücken der Taste "Geld zurück" in einen Nachfolgezustand, in dem die Münze zurückgegeben wird.

Etwas genauer: Automat verfügt über Münzschlitz sowie Tasten "Espresso", "Cappuccino", "Geldrückgabe", die vom Kunden bedient werden können. Außerdem "weiß" der Automat, ob er gerade Geld zurückgegeben oder Cappuccino bzw. Espresso geliefert hat ("Eingaben", die durch eigene Aktion hervorgerufen werden). Insgesamt sind die möglichen Eingaben:

M: Münze eingeworfen
E: Espresso-Taste gedrückt
C: Cappuccino-Taste gedrückt
G: Geldrückgabetaste gedrückt
R: Geldrückgabe erfolgt
LC: Lieferung Cappuccino abgeschlossen
LE: Lieferung Espresso abgeschlossen



Der Automat hat folgende Zustände:

q_0: Anfangszustand
q_1: Münze eingeworfen
q_2: Münze eingeworfen und Espresso gedrückt
q_3: Münze eingeworfen und Cappucino gedrückt
q_4: Münze eingeworfen und Geldrückgabe gedrückt

Die durch bestimmte Eingaben hervorgerufenen Übergänge sind im Diagramm veranschaulicht.

Die Beschreibung ist so noch nicht vollständig, denn z.B. kann der Kunde ja auch zuerst E oder C eingeben. Um das Bild nicht komplizierter zu machen, gehen wir davon aus, dass für alle nicht explizit genannten Eingaben jeweils der ursprüngliche Zustand erhalten bleiben soll. Ein realistischer Automat sollte natürlich noch verfeinert werden (z.B. merken, wenn zu viele Münzen eingeworfen wurden, oder in einen Fehlerzustand gehen, wenn ohne Anforderung Kaffee geliefert wird). Es geht hier nur um die Verdeutlichung des Prinzips.

Anmerkung: q_0 ist insofern ein ausgezeichneter Zustand, als alle Eingabefolgen, die wieder in q_0 landen, vollständige Transaktionen darstellen. Zustände, die durch "gewünschte" Eingabefolgen erreicht werden, heißen finale oder akzeptierende Zustände.

Für unsere Zwecke wird es genügen, uns Eingaben als Symbole und Folgen von Eingaben als Zeichenketten aus entsprechenden Symbolen vorzustellen. -> Zusammenhang zu formalen Sprachen

Warum sollte man sich als Informatiker mit Automatentheorie auseinander setzen?

- liefert Grundkonzepte für Design wichtiger Typen von Software (Schaltkreisentwurf, lexikalische Analyse im Compilerbau, Suche von Ausdrücken in Texten, Verifikation von Kommunikationsprotokollen, ...)
- wesentlich für Analyse der (Grenzen der) Berechenbarkeit
- wesentlich für Analyse der Komplexität von Algorithmen

1.2 Formale Sprachen

heißen formal, weil es sich um präzise definierte Kunstsprachen handelt (im Gegensatz zu natürlichen Sprachen)

Def.: Ein Alphabet Σ ist eine endliche, nichtleere Menge von Symbolen.

Bsp.: {0,1}, {a,b,c}, {♦,♥,♠,●},...

Def.: Eine Zeichenreihe (auch: Zeichenkette, Wort) über einem Alphabet Σ ist eine endliche Folge von Elementen von Σ.

In obiger Definition ist die "leere" Zeichenkette mit eingeschlossen. Diese wird durch das Symbol ε dargestellt.
Die Länge der leeren Zeichenkette ist 0. Sei w eine Zeichenkette der Länge n, $a \in \Sigma$, so ist die Länge der Zeichenkette aw (Notation |aw|) n+1.

Σ^k bezeichnet die Menge der Zeichenketten über Σ der Länge k.
Σ^* ist die Menge aller Zeichenketten über Σ, d,h, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ ...$
Σ^+ ist die Menge aller nichtleeren Zeichenketten über Σ, d,h, $\Sigma^+ = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \ ...$

Def.: Seien x und y Zeichenketten mit $|x| = n_x$ und $|y| = n_y$. Die Konkatenation von x und y ist die Zeichenkette xy mit $|xy| = n_x + n_y$, die durch Aneinanderreihung von x und y entsteht.

Def.: Sei Σ ein Alphabet. S heißt Sprache über Σ gdw. $S \subseteq \Sigma^*$.

Anmerkung: in der theoretischen Informatik werden Probleme oft gleichgesetzt mit der Frage, ob ein Wort zu entsprechender Sprache gehört.
Bsp.: das Problem: "ist n Primzahl?" ist äquivalent zu:
gehört Binärdarstellung von n zur Sprache {w | w binäre Zahl, die Primzahl ist}.

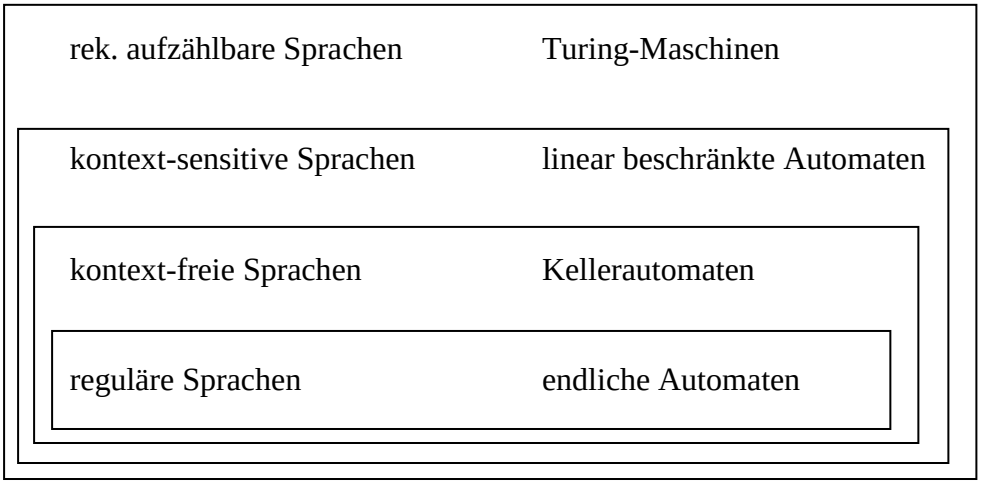
Wie werden Sprachen definiert?

a) Aufzählung der Elemente (klappt nur im endlichen Fall)
b) Mengenvorschrift: {w | w binäre Zahl, die Primzahl ist}.
c) Grammatiken: Erzeugung von Zeichenketten durch Menge von Ersetzungsregeln

Was haben formale Sprachen und Automaten miteinander zu tun?

Automaten können Sprachen erkennen: die von einem Automaten erkannte Sprache besteht aus den Zeichenketten, deren Eingabe vom Anfangs- in einen akzeptierenden Zustand führt. Verschiedene Typen von Automaten entsprechen exakt verschiedenen Typen von Sprachen.

Chomsky Hierarchie (Noam Chomsky, Linguist)



weitere Möglichkeit, Sprache zu definieren: Angabe eines Automaten, der sie akzeptiert.

Geplante Gliederung der Vorlesung:

1. Einführung
2. Endliche Automaten
3. Reguläre Ausdrücke und Sprachen
4. Eigenschaften regulärer Sprachen
5. Kontextfreie Grammatiken und Sprachen
6. Kellerautomaten
7. Eigenschaften kontextfreier Sprachen
8. Turing-Maschinen
9. Unentscheidbarkeit

2. Endliche Automaten

2.1 Deterministische endliche Automaten (DEAs)

Def.: Ein deterministischer endlicher Automat $A = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einer endlichen Menge von Eingabesymbolen Σ,
- einer Übergangsfunktion $\delta: Q \times \Sigma \rightarrow Q$, wobei $\delta(q,a)$ der Zustand ist, der vom Zustand q bei Eingabe a erreicht wird,
- einem Startzustand $q_0 \in Q$, sowie
- einer Menge $F \subseteq Q$ von finalen oder akzeptierenden Zuständen.

Die Übergangsfunktion δ lässt sich auf folgende Weise zu einer Funktion $\delta*: Q \times \Sigma* \rightarrow Q$ erweitern:

für alle Zustände q, Zeichenketten w und Eingabesymbole a gilt:

$\delta*(q, \varepsilon)$	$= q$	
$\delta*(q, wa)$	$= \delta(\delta*(q, w), a)$	Formulierung im Buch, oder
$\delta*(q, aw)$	$= \delta*(\delta(q, a), w)$	zu Formulierung im Buch äquivalent

Beispiel: $Q = \{q_0, q_1\}$, $\Sigma = \{0,1\}$, $F = \{q_0\}$

$\delta(q_0, 0) = q_0$
$\delta(q_1, 0) = q_1$
$\delta(q_0, 1) = q_1$
$\delta(q_1, 1) = q_0$

es gilt z.B.:

$$
\begin{aligned}
\delta*(q_0, 0110) &= \delta*(\delta(q_0, 0), 110) \\
&= \delta*(q_0, 110) \\
&= \delta*(\delta(q_0,1), 10) \\
&= \delta*(q_1, 10) \\
&= \delta*(\delta(q_1,1), 0) \\
&= \delta*(q_0, 0) \\
&= \delta*(\delta(q_0,0), \varepsilon) \\
&= \delta*(q_0, \varepsilon) \\
&= q_0
\end{aligned}
$$

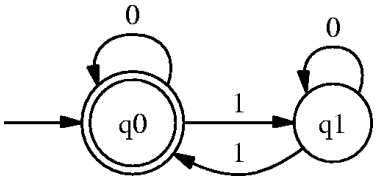
Man sagt, die Eingabe 0110 wird von dem Automaten akzeptiert.

Def.: Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Die von A akzeptierte Sprache (oder kürzer: die Sprache von A) L(A) wird definiert durch:

$$L(A) = \{w \mid \delta*(q_0, w) \in F\}$$

Welche Sprache wird durch den obigen Beispielautomaten akzeptiert? Menge aller Wörter mit gerader Anzahl von 1.

Übergangsdiagramm: (Startzustand q0 gekennzeichnet durch eingehenden Pfeil ohne Vorgängerzustände, finale durch Doppelkreis)

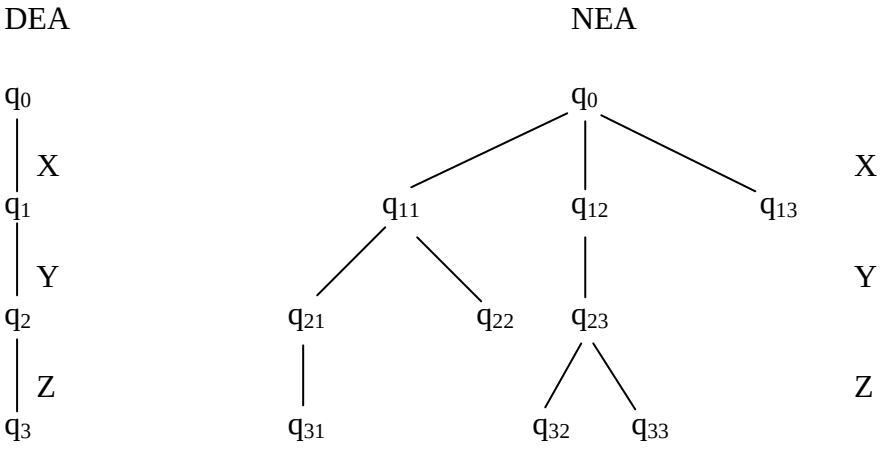


Bemerkung: δ wird oft in Tabellenform dargestellt. Für obigen Automaten (* bezeichnet akzeptierende Zustände, der Pfeil zeigt auf den Startzustand :

δ	0	1
→ * q_0	q_0	q_1
q_1	q_1	q_0

2.2 Nichtdeterministische endliche Automaten (NEAs)

DEA geht bei Eingabe eines Symbols in genau einen definierten Nachfolgezustand über. Ein NEA dagegen kann bei derselben Eingabe mehrere Nachfolgezustände (oder auch keinen) haben. Die verschiedenen möglichen Nachfolgezustände entsprechen Hypothesen über die bisher abgearbeitete Eingabe:



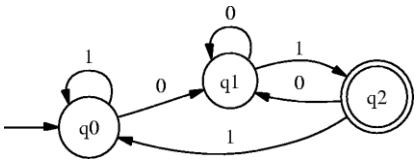
Menge der von q_0 bei Eingabe von XYZ erreichbaren Zustände: q_{31}, q_{32}, q_{33}
Wort akzeptiert, wenn mindestens ein akzeptierender Zustand erreichbar.

Motivation für NEA:

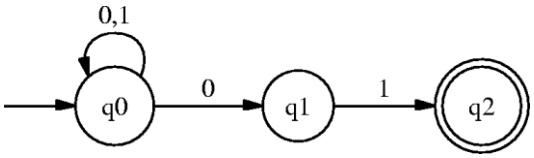
NEAs sind für eine Reihe von praktischen Problemen übersichtlicher und leichter zu entwerfen (und manchmal exponentiell kleiner: siehe weiter unten).

Beispiel: L_{01}={w01} über dem Alphabet {0,1}, d.h. alle, die mit 01 aufhören:

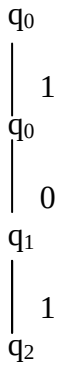
DEA

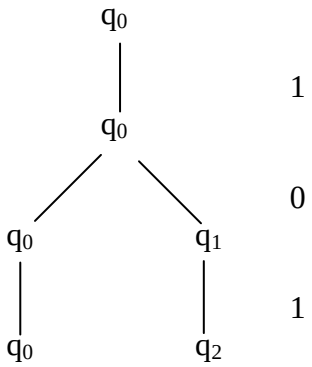


NEA



Eingabe von 101:





Beide Automaten akzeptieren 101. Zu beachten: bei NEA genügt es, dass akzeptierender Zustand q_2 unter den erreichbaren ist. Es dürfen auch nicht-akzeptierende erreichbar sein.

Def.: Ein nichtdeterministischer endlicher Automat $A = (Q, \Sigma, \delta, q_0, F)$ besteht aus

- einer endlichen Menge von Zuständen Q,
- einer endlichen Menge von Eingabesymbolen Σ,
- einer Übergangsfunktion $\delta: Q \times \Sigma \rightarrow 2^Q$, wobei δ(q,a) die (möglicherweise leere) Menge der Zustände ist, die vom Zustand q bei Eingabe a erreicht wird,
- einem Startzustand $q_0 \in Q$, sowie[1]
- einer Menge $F \subseteq Q$ von finalen oder akzeptierenden Zuständen.

2^Q bezeichnet hier die Potenzmenge von Q, also $2^Q = \{R \mid R \subseteq Q\}$.

Die Übergangsfunktion δ lässt sich wieder zu einer Funktion $\delta*: Q \times \Sigma* \rightarrow 2^Q$ erweitern: für alle Zustände q, Zeichenketten w und Eingabesymbole a gilt:

$$\delta*(q, \varepsilon) = \{q\}$$

$$\delta*(q, wa) = \bigcup_{p \in \delta*(q,w)} \delta(p, a)$$ oder äquivalent:

$$\delta*(q, aw) = \bigcup_{p \in \delta(q,a)} \delta*(p, w)$$

[1] Einige Autoren lassen bei NEAs auch mehrere Anfangszustände zu. Wir werden bald endliche Automaten mit ε-Übergängen einführen. Mit diesen lassen sich mehrere Anfangszustände einfach simulieren. Wir halten uns hier an die Darstellung, die auch von Schöning und Hopcroft/Motwani/Ullman verwendet wird.

Eingabe 0110 in unserem Beispielautomaten (NEA für L_{01}):

$\delta^*(q_0, 0110) = \delta^*(q_0, 110) \cup \delta^*(q_1, 110)$
$= \delta^*(q_0, 10) \cup \delta^*(q_2, 10)$
$= \delta^*(q_0, 0)$
$= \delta^*(q_0, \varepsilon) \cup \delta^*(q_1, \varepsilon)$
$= \{q_0\} \cup \{q_1\}$
$= \{q_0, q_1\}$

δ: $\delta(q_0,0) = \{q_0, q_1\}$
$\delta(q_0,1) = \{q_0\}$
$\delta(q_1,0) = \{\}$
$\delta(q_1,1) = \{q_2\}$
$\delta(q_2,0) = \{\}$
$\delta(q_2,1) = \{\}$

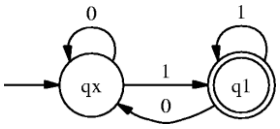
Def.: Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein NEA. Die von A akzeptierte Sprache L(A) wird definiert durch:

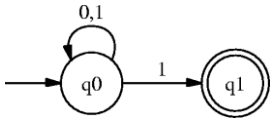
$$L(A) = \{w \mid \delta^*(q_0, w) \cap F \neq \varnothing\}.$$

Bsp. (das zeigt, dass DEA exponentiell größer werden kann als NEA):
Folge von Sprachen $L_i = \{w \in \{0,1\}^* \mid$ i-tes Symbol von hinten in w ist 1$\}$

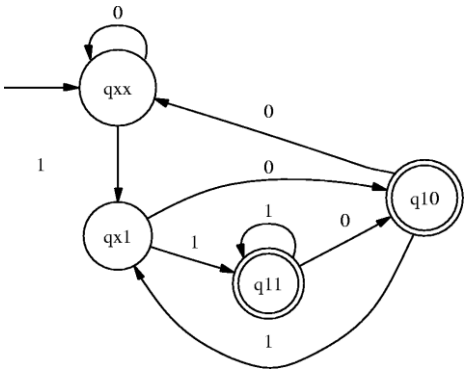
DEA NEA

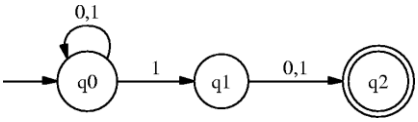
i=1



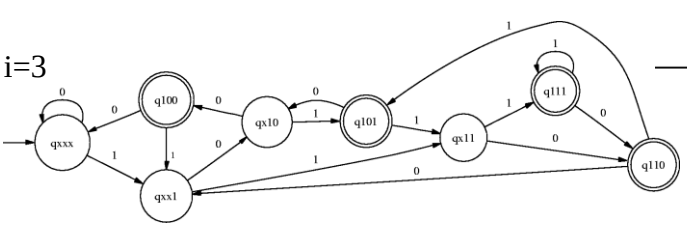


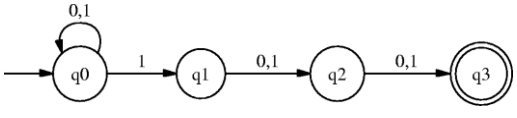
i=2





i=3





2^i Zustände i+1 Zustände

Hinweis zu i=2:
qxx: zuletzt 00 gelesen, oder noch gar nichts, oder nur genau eine 0
qx1: zuletzt 01 gelesen, oder bisher nur genau eine 1

q11: zuletzt 11 gelesen
q10: zuletzt 10 gelesen

Jeder DEA kann als NEA aufgefasst werden, bei dem δ immer eine ein-elementige Menge bestehend aus dem Nachfolgezustand des DEA liefert. Alle Sprachen, die von einem DEA akzeptiert werden, werden deshalb auch von einem NEA akzeptiert.

Aber gibt Sprachen, die von einem NEA, aber nicht von einem DEA akzeptiert werden?
Nein: zu jedem NEA gibt es einen äquivalenten DEA, den sogenannten Potenzautomaten.

Def.: Die Automaten A_1 und A_2 heißen äquivalent, falls $L(A_1) = L(A_2)$.

Konstruktion des Potenzautomaten (Teilmengenkonstruktion):
Sei $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ ein nichtdeterministischer endlicher Automat. Wir definieren einen deterministischen endlichen Automaten wie folgt: $D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ wobei:

$$Q_D = 2^{Q_N}$$

$$\delta_D(S,a) = \bigcup_{p \in S} \delta_N(p, a) \qquad \text{für alle } S \in Q_D,\ a \in \Sigma$$

$$q_{0D} = \{q_0\}$$

$$F_D = \{S \in Q_D \mid S \cap F_N \neq \varnothing\}$$

Bemerkung: wenn n die Anzahl der Zustände von N ist, so besitzt D 2^n Zustände. In vielen Fällen sind aber nicht alle Zustände erreichbar und der DEA lässt sich verkleinern.

Anmerkung: die Menge der erreichbaren Zustände R(A) eines Automaten A ist die kleinste Menge mit:

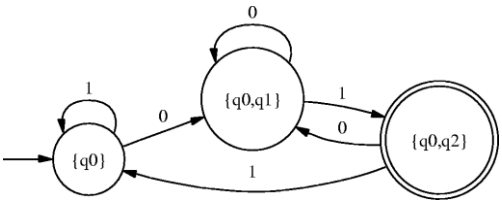
1. $q_0 \in R(A)$,
2. $q \in R(A)$ und $\delta(q,a) = p$ für eine Eingabe a impliziert $p \in R(A)$.

Unser Beispielautomat:

		0	1
	$\varnothing$	$\varnothing$	$\varnothing$
$\rightarrow$	$\{q_0\}$	$\{q_0, q_1\}$	$\{q_0\}$
	$\{q_1\}$	$\varnothing$	$\{q_2\}$
*	$\{q_2\}$	$\varnothing$	$\varnothing$
	$\{q_0,q_1\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$
*	$\{q_0,q_2\}$	$\{q_0,q_1\}$	$\{q_0\}$
*	$\{q_1,q_2\}$	$\varnothing$	$\{q_2\}$
*	$\{q_0,q_1,q_2\}$	$\{q_0,q_1\}$	$\{q_0,q_2\}$

von $\{q_0\}$ aus erreichbare Zustände: $\{q_0\}$, $\{q_0, q_1\}$, $\{q_0,q_2\}$

Graph des Automaten (ohne unerreichbare Zustände):



bis auf die Benennung der Zustände identisch mit unserem ursprünglichen DEA (siehe S. 6).
Satz: Sei $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ ein nichtdeterministischer endlicher Automat, $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ der Potenzautomat von N. D und N sind äquivalent.

Beweis: wir zeigen durch Induktion über die Länge der Eingabe w, dass für alle w

$$\delta_D^*(\{q_0\}, w) = \delta_N^*(q_0, w)$$

Mit diesem Resultat folgt der Satz unmittelbar aus der Definition von L(N) und L(D).

Induktionsanfang: für $w = \varepsilon$ gilt $\delta_D^*(\{q_0\}, \varepsilon) = \delta_N^*(q_0, \varepsilon) = \{q_0\}$

Induktionsschritt: sei $|w| = n+1$ und der Satz gelte für Wörter der Länge n.
$w = xa$ mit $|x| = n$. Damit gilt $\delta_D^*(\{q_0\}, x) = \delta_N^*(q_0, x)$. Nach Definition ist

$$\delta_N^*(q_0, xa) = \bigcup_{p \in \delta_N^*(q_0,x)} \delta_N(p, a) = \bigcup_{p \in \delta_D^*(\{q_0\},x)} \delta_N(p, a) = \bigcup_{p \in \delta_D^*(\{q_0\},x)} \delta_D(\{p\}, a) = \delta_D^*(\{q_0\}, xa)$$

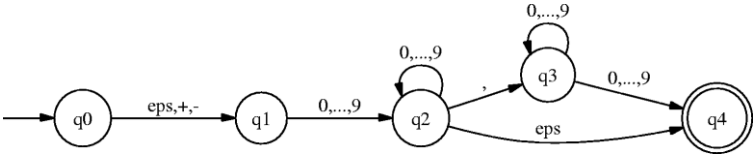
Def. δ_N^* — Ind.-vor. — Def. δ_D — Def. δ_D^*

qed.

2.3 Endliche Automaten mit ε-Übergängen

Es erweist sich als praktisch (z.B. zur Behandlung von optionalen Sprachkonstrukten), Automaten zuzulassen, bei denen Zustandsübergänge auch ganz ohne Eingabe erfolgen können (ε-Übergänge).

Beispiel: Ein Automat, der ganze Zahlen sowie Dezimalzahlen mit optionalem Vorzeichen erkennt (falls Komma vorkommt, muss mindestens eine Ziffer folgen).



Formale Definition: Ein ε-NEA $A = (Q, \Sigma, \delta, q_0, F)$ ist definiert wie ein NEA, bis auf den Definitionsbereich von δ. ε wird als mögliche Eingabe zugelassen:

$$\delta: Q \times \Sigma \cup \{\varepsilon\} \rightarrow 2^Q$$

Die ε-Hülle eines Zustands q, ε-Hülle(q), ist die kleinste Menge von Zuständen für die gilt:

1. q ist in ε-Hülle(q),
2. wenn p in ε-Hülle(q) und r von p aus bei ε-Eingabe erreichbar, so ist r in ε-Hülle(q).

Beispiel:
ε-Hülle(q_0) = {q_0, q_1}
ε-Hülle(q_2) = {q_2, q_4}

Wir erweitern die Definition der ε-Hülle auf Mengen von Zuständen Q:

$$\varepsilon\text{-Hülle}(Q) = \bigcup_{q \in Q} \varepsilon\text{-Hülle}(q)$$

Wieder erweitern wir δ zu δ*. Für alle Zustände q, Zeichenketten w und Eingabesymbole a gilt:

$\delta^*(q, \varepsilon) = \varepsilon\text{-Hülle}(q)$

$\delta^*(q, wa) = \bigcup_{p \in \delta^*(q,w)} \varepsilon\text{-Hülle}(\delta(p, a))$ oder äquivalent:

$\delta^*(q, aw) = \bigcup_{\substack{p \in (\delta(r,a)),\\ r \in \varepsilon\text{-Hülle}(q)}} \delta^*(p, w)$

Akzeptierte Sprache definiert wie bei NEA.
Wiederum lassen sich ε-NEAs in äquivalente DEAs überführen:

Sei $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ ein ε-NEA. Wir definieren einen deterministischen endlichen Automaten wie folgt: $D = (Q_D, \Sigma, \delta_D, q_{0D}, F_D)$ wobei:

Q_D = Menge aller Teilmengen S von Q_E, für die S = ε-Hülle(S)

$\delta_D(S,a) = \bigcup_{p \in S} \varepsilon\text{-Hülle}(\delta_E(p, a))$ für alle $S \in Q_D$, $a \in \Sigma$

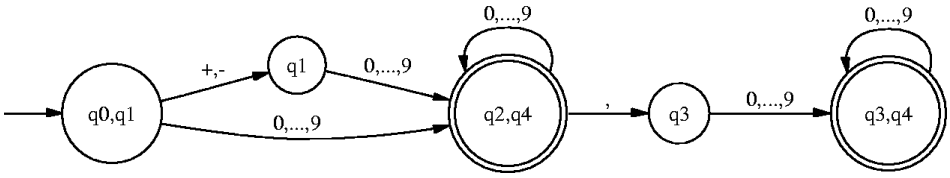
$q_{0D} = \varepsilon\text{-Hülle}(q_0)$

$F_D = \{S \in Q_D \mid S \cap F_E \neq \varnothing\}$

Unser Beispiel ergibt (nur erreichbare Zustände werden dargestellt):

	0,...,9	+,-	Komma
{q_0,q_1}	{q_2,q_4}	{q_1}	∅
{q_2,q_4}	{q_2,q_4}	∅	{q_3}
{q_1}	{q_2,q_4}	∅	∅
∅	∅	∅	∅
{q_3}	{q_3,q_4}	∅	∅
{q_3,q_4}	{q_3,q_4}	∅	∅

Graphische Darstellung des obigen Graphen:



Hinweis: der Zustand ∅ ist ein „Fehlerzustand“, aus dem man nicht mehr rauskommt. ∅ sowie die zugehörigen Kanten werden häufig bei der Darstellung weggelassen. Konvention: wenn bei DEA Eingaben fehlen, so führen diese zu Fehlerknoten.

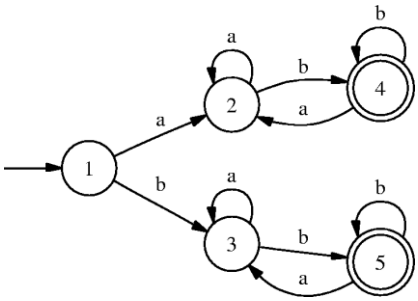
Satz: Sei E ein ε-NEA, D der aus ihm nach obigem Verfahren konstruierte DEA. E und D sind äquivalent.

Hinweis: Wir haben hier einen ε-NEA direkt in einen DEA überführt. Die Überführung in einen NEA ist einfacher. Grundidee: bei der Berechnung der Nachfolgerknoten bei Eingabe a berechnet man zunächst die mit ε erreichbaren Zustände und dann deren mögliche Nachfolger bei Eingabe a. Man muss dann nur noch berücksichtigen, dass akzeptierende Zustände nach Lesen der kompletten Eingabe mit ε erreichbar sein können:

Ein ε-NEA $E=(Q_E, \Sigma_E, \delta_E, q_{0E}, F_E)$ ist äquivalent zum NEA $N= (Q_E, \Sigma_E, \delta_N, q_{0E}, F_N)$ wobei $\delta_N(q,a) = \bigcup_{p \in \varepsilon-Hülle(q)} \delta_E(p,a)$ und $F_N = \{q \in Q_E \mid \varepsilon\text{-Hülle}(q) \cap F_E \neq \emptyset\}$.

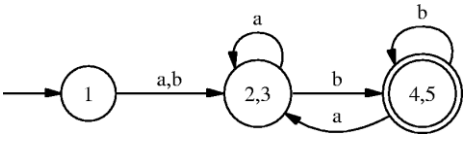
2.4 Äquivalenz und Minimierung von Automaten

Betrachten Sie folgenden Automaten:



Erkennt die Sprache aller Wörter über {a,b}* mit |w| > 1, die mit b enden.

Gibt es einen einfacheren Automaten für diese Sprache? Klar:



Wie findet man die einfachsten Automaten für eine Sprache? Suche äquivalente Zustände und ersetze diese durch einen einzigen Zustand. Äquivalent heißt hier selbes Akzeptanzverhalten:

Def. Sei D = (Q, Σ, δ, q_0, F) ein DEA. Zwei Zustände p, q aus Q heißen äquivalent, falls für alle w aus Σ* gilt: δ*(p,w) ∈ F genau dann wenn δ*(q,w) ∈ F.

Wie findet man äquivalente Zustände in einem Automaten?

Die Menge $DIFF_A$ aller (ungeordneten) Paare nicht-äquivalenter Zustände in einem Automaten A ist die kleinste Menge, für die gilt:

1. falls p akzeptierender, q nicht-akzeptierender Zustand, so ist {p,q} in $DIFF_A$,
2. falls, für eine Eingabe a, {δ(r,a), δ(s,a)} in $DIFF_A$, so ist {r,s} in $DIFF_A$.

Entsprechende Algorithmen sind leicht zu realisieren. Entspricht dem Ausfüllen einer Tabelle M. Wir nehmen an, die Zustände seien von 1 bis n durchnummeriert:

for i < j ≤ n **do if** i ∈ F und j ∉ F oder j ∈ F und i ∉ F **then** markiere M(i,j) mit X_0;
if Feld wurde markiert **then** change:= true **else** change := false;
index:= 0;
while change **do**
    change := false;
    index := index + 1;
    for i < j ≤ n, a ∈ Σ **do**
        if nicht markiert M(i,j) und markiert M(δ(i,a),δ(j,a)) mit $X_{index-1}$ **then**
            markiere M(i,j) mit X_{index};
            change := true;

Anmerkung: die Indizes der Markierung sind nicht unbedingt nötig, aber hilfreich: man kann zeigen: Algorithmus markiert M(i,j) mit X_k genau dann wenn Zustände i, j nicht äquivalent sind und kürzestes Gegenbeispiel hat Länge k.

Unser Beispiel: In der Initialisierungsphase werden alle Paare aus akzeptierenden und nichtakzeptierenden Zuständen markiert:

4				
3	X_0	X_0		
2	X_0	X_0		
1	X_0	X_0		
	5	4	3	2

Jetzt Zustandspaare p,q zu finden, die für bestimmte Eingabe zu bereits markierten Zustandspaaren führen:

δ(1,b) = 3
δ(2,b) = 4 M(3,4) bereits markiert, also markiere M(1,2)

δ(1,b) = 3
δ(3,b) = 5 M(3,5) bereits markiert, also markiere M(1,3)

Resultat:

4				
3	X_0	X_0		
2	X_0	X_0		
1	X_0	X_0	X_1	X_1
	5	4	3	2

Keine weiteren Markierungen sind möglich:

$\delta(4,a) = 2$ $\delta(4,b) = 4$
$\delta(5,a) = 3$ $\delta(5,b) = 5$ weder M(2,3) noch M(4,5) markiert

$\delta(2,a) = 2$ $\delta(2,b) = 4$
$\delta(3,a) = 3$ $\delta(3,b) = 5$ weder M(2,3) noch M(4,5) markiert

Damit sind 4,5 und 2,3 äquivalent.

Sei $A = (Q_A, \Sigma, \delta_A, q_{0A}, F_A)$ ein DEA ohne unerreichbare Zustände (gegebenenfalls vorher eliminieren). Erzeugung des minimalen DEA $M = (Q_M, \Sigma, \delta_M, q_{0M}, F_M)$ aus A:

1. Bestimme mit obiger Methode äquivalente Zustände in A.
2. Zustände von M sind die Klassen äquivalenter Zustände von A.
3. $\delta_M(S, x) = Q$ gdw $\delta_A(s, x) = q$ für ein $s \in S$ und ein $q \in Q$.
4. $S = q_{0M}$ gdw. $q_{0A} \in S$.
5. $S \in F_M$ gdw. $s \in F_A$ für ein $s \in S$.

Bemerkung:
Zu 3: wenn ein $s \in S$ in A in Zustand der Äquivalenzklasse Q landet, so gilt das für alle $s \in S$.
Zu 5: wenn ein $s \in S$ in A akzeptierender Zustand ist, so gilt das für alle $s \in S$.

In unserem Beispiel:
Äquivalenzklassen {1}, {2,3}, {4,5}
Startzustand {1}, F = {{4,5}}
$\delta(\{1\},a) = \delta(\{1\},b) = \{2,3\}$
$\delta(\{2,3\},a) = \{2,3\}$, $\delta(\{2,3\},b) = \{4,5\}$
$\delta(\{4,5\},a) = \{2,3\}$, $\delta(\{4,5\},b) = \{4,5\}$

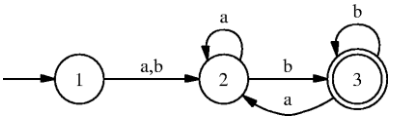
Satz: Sei A ein DEA, M der nach dem oben beschriebenen Verfahren aus A erzeugte DEA. Es gibt keinen zu A äquivalenten DEA mit weniger Zuständen als M.

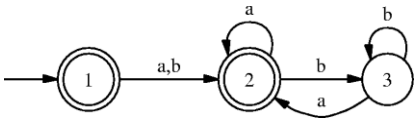
2.5 Kombination endlicher Automaten

In diesem Abschnitt wollen wir untersuchen, wie man endliche Automaten für Sprachen L_1 bzw. L_2 so kombinieren kann, dass Automaten für Komplement, Vereinigung, Schnitt, Konkatenation und Kleene-Abschluss (*-Operator) entstehen.

Seien $M_1 = (Q_1, \Sigma, \delta_1, q_{0,1}, F_1)$ und $M_2 = (Q_2, \Sigma, \delta_2, q_{0,2}, F_2)$ beliebige endliche Automaten, die Sprachen L_1 bzw. L_2 akzeptieren. Annahme: Q_1 und Q_2 disjunkt (sonst Umbenennen von Zuständen).

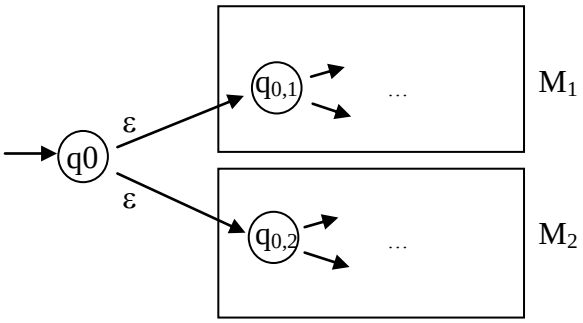
Komplement von L_1:
Voraussetzung: M_1 ist DEA (gegebenenfalls ist Potenzautomat zu konstruieren).
Der endliche Automat für $\Sigma^* \backslash L_1$ entsteht aus dem Automaten M_1 durch Vertauschen von akzeptierenden und nicht-akzeptierenden Zuständen.

Beispiel: aus 

wird 

Konstruktion klappt nicht für NEAs. Warum?

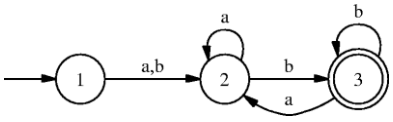
Vereinigung von L_1 und L_2:
Einfügen eines neuen Startzustandes q_0 mit ε-Übergängen zu den beiden ursprünglichen Startzuständen von M_1 und M_2.

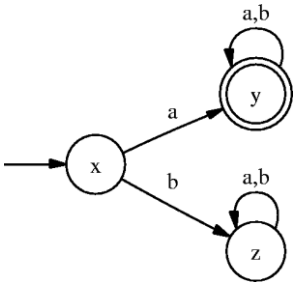


Schnitt von L_1 und L_2:
Konstruktion des Produktautomaten $M_1 \times M_2 = (Q_1 \times Q_2, \Sigma, \delta, (q_{0,1}, q_{0,2}), F_1 \times F_2)$ mit

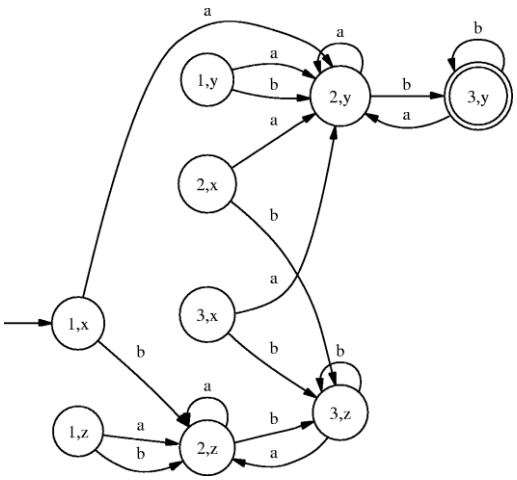
$$\delta((q_1, q_2), a) = \{ (p_1, p_2) \mid p_1 \in \delta_1(q_1, a) \text{ und } p_2 \in \delta_2(q_2, a)\}.$$

Idee: Zustände des neuen Automaten sind Paare von Zuständen der ursprünglichen Automaten. Neuer Automat nach Eingabe w in Zustand (r, s) gdw. (nach Eingabe w) M_1 in Zustand r und M_2 in Zustand s.

Beispiel: aus 

und 

wird:



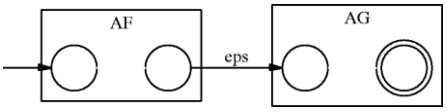
	a	b
→ (1, x)	(2, y)	(2, z)
(1, y)	(2, y)	(2, y)
(1, z)	(2, z)	(2, z)
(2, x)	(2, y)	(3, z)
(2, y)	(2, y)	(3, y)
(2, z)	(2, z)	(3, z)
(3, x)	(2, y)	(3, z)
* (3, y)	(2, y)	(3, y)
(3, z)	(2, z)	(3, z)

erreichbar sind nur: (1,x), (2,y), (2, z), (3,y), (3, z)

Konkatenation von L_1 und L_2 ($L_1L_2 = \{w_1w_2 \mid w_1 \in L_1 \text{ und } w_2 \in L_2\}$:
$M_1 \circ M_2 = (Q_1 \cup Q_2, \Sigma, \delta, q_{0,1}, F_2)$ mit

$$\delta(q, a) = \begin{cases} \delta_1(q, a) & \text{falls } q \in Q_1 \text{ und } (q \notin F_1 \text{ oder } a \neq \varepsilon), \\ \delta_2(q, a) & \text{falls } q \in Q_2, \\ \{q_{0,2}\} \cup \delta_1(q, \varepsilon) & \text{falls } q \in F_1 \text{ und } a = \varepsilon. \end{cases}$$

Illustration: Automat entsteht durch „Hintereinanderschalten“: Von akzeptierenden Zuständen in M_1 ε-Übergang zu Startzustand von M_2, akzeptierende Zustände in M_1 und Startzustand von M_2 werden normale Zustände, akzeptierender Zustand von M_2 ist Endzustand des Gesamtautomaten



Kleene-Abschluss L^* ($L^0=\{\varepsilon\}$, $L^{n+1}=LL^n$, $L^* = \bigcup_{i \geq 0} L^i$):

Sei $M = (Q, \Sigma, \delta, q_0, F)$ Automat für L. Wir definieren $M^* = (Q \cup \{q_s, q_f\}, \Sigma, \delta', q_s, \{q_f\})$, wobei q_s, q_f neue Zustände sind und d’ wie folgt definiert wird:

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{falls } a \in \Sigma \text{ und } q \in Q, \\ \{q_0, q_f\} \cup \delta(q, \varepsilon) & \text{falls } a = \varepsilon \text{ und } q \in F, \\ \delta(q, \varepsilon) & \text{falls } a = \varepsilon \text{ und } q \in Q \backslash F, \\ \{q_0, q_f\} & \text{falls } a = \varepsilon \text{ und } q = q_s, \\ \varnothing & \text{sonst.} \end{cases}$$

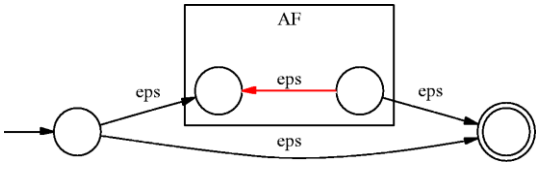


Illustration: Automat für L* durch „Rückkopplung“ im ursprünglichen Automaten: ε-Übergang von alten Endzuständen zu altem Startzustand, neuer Startzustand mit ε-Übergang zu altem Startzustand und neuem Endzustand, neuer Endzustand auch von alten Endzuständen per ε-Übergang erreichbar.
(Direkter ε-Übergang von q_0 zu q_f klappt nicht: es würde sonst *jedes* Wort, das zu q_0 führt, akzeptiert werden; ebenso klappt nicht direkter ε-Übergang von q_s zu alten Finalzuständen: Wörter, die von Zuständen in F zu Zuständen in F führen, würden erkannt).