Theoretische Informatik – Schreibomat

Florian

October 9, 2012

Tools

- Finite State Machine Designer: `http://madebyevan.com/fsm/`

1 Was ist Informatik?

- Konkrete[1] Konzepte: Algorithmus, Berechnung, Komplexität
- Was will die theoretische Informatik? Formale Bestimmung der Begriffe, unabhängig von Hard- und Software
- Ziel: Grundlegende Eigenschaften von Algorithmen, Berechnungen erkennen. Methoden entwickeln zum Entwurf beweisbar korrekter Hard- und Software (nicht Schwerpunkt dieser Vorlesung)
- Grenzen der automatischen Berechnung aufzeigen
- Typische Fragestellungen: Ist es möglich, ein Programm zu schreiben, das ein anderes Programm als Eingabe erhält und feststellen kann, ob dieses in eine Endlosschleife gerät oder nicht? $\Rightarrow$ Halteproblem. Nein, es ist natürlich nicht möglich. Eine andere typische Fragestellung ist die Folgende: Wir haben einen Rucksack, und der hat 30 Liter Fassungsvermögen. Und wir haben noch 50 Gegenstände, und jetzt wollen wir wissen: wie lange dauert es, herauszufinden, wieviele Gegenstände in den Rucksack passen? Rucksackproblem, nicht in polynomialer Zeit lösbar $\Rightarrow$ Es dauert expontiell lange.
- Leider ist es in der Praxis so, dass die Antwort auf ähnliche Fragestellungen nicht immer so offensichtlich ist, und deshalt müssen wir es uns ein bisschen genauer betrachten.

Übung. Gegeben sei das folgende Strassennetz:
(Wildes Gekritzel)

[1] 4. September 2012

(a) Gibt es eine Möglichkeit, alle Strassen genau 1x zu durchlaufen und dann wieder am Ausgangspunkt anzulangen,

(b) Gibt es eine Möglichkeit, alle Kreuzungen genau 1x zu passieren und dann wieder am Ausgangspunkt anzulangen?

Die Antworten sind imfall:

(a) Einfache Aufgabe: Geht immer, wenn die Anzahl der Abzweigungen an jeder Kreuzung gerade ist.

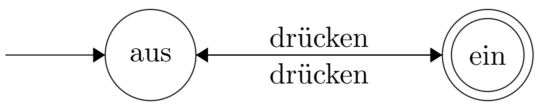
(b) Traveling salesman (TSP). Schwierig! Es ist keine wesentlich bessere Methode bekannt, als einfach alles durchzuprobieren.

Ziel für die Vorlesung: Um solche Fragestellungen systematisch beantworten zu können, brauchen wir ein exaktes mathematisches Modell eines Computers.

2 Automatentheorie

Endliche Automaten, Kontextfreie Grammatiken und Keller-Automaten. Zusätzliche Motivation: Das sind Konzepte, denen man auch häufig in der Praxis begegnet (Compilerbau, Textsuche, Textverarbeitung)

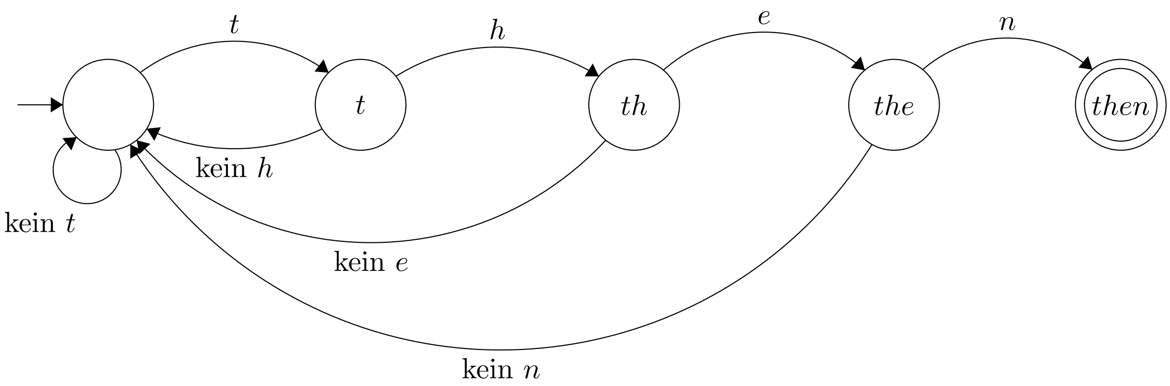
Noch nicht ganz so formal, bisschen intuitiv. Modellierung eines Kippschalters.



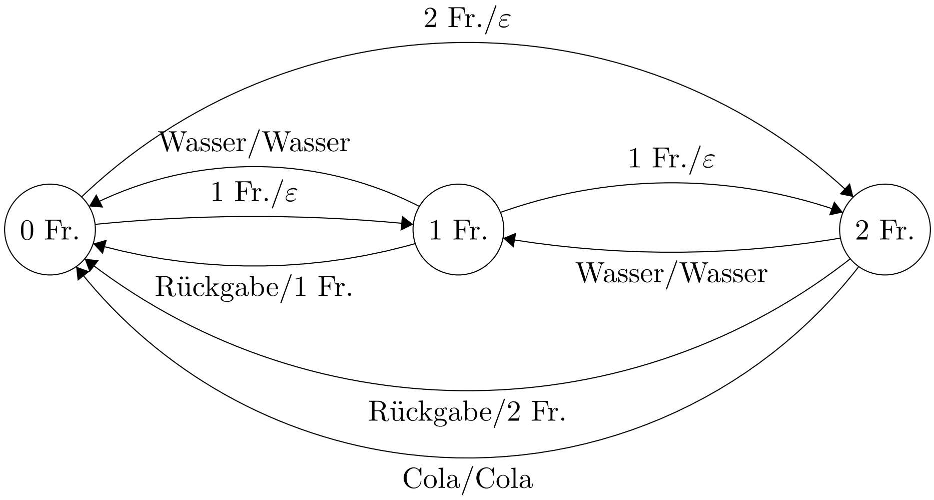
Definition 1 (Endlicher Automat)**.** Wir haben:

- Zwei Zustände, davon ein Startzustand
- und ein akzeptierender Zustand
- Transitionen (Zustandsübergänge): führen den Automaten anhand einer Eingabe von einem Zusand in den nächsten.

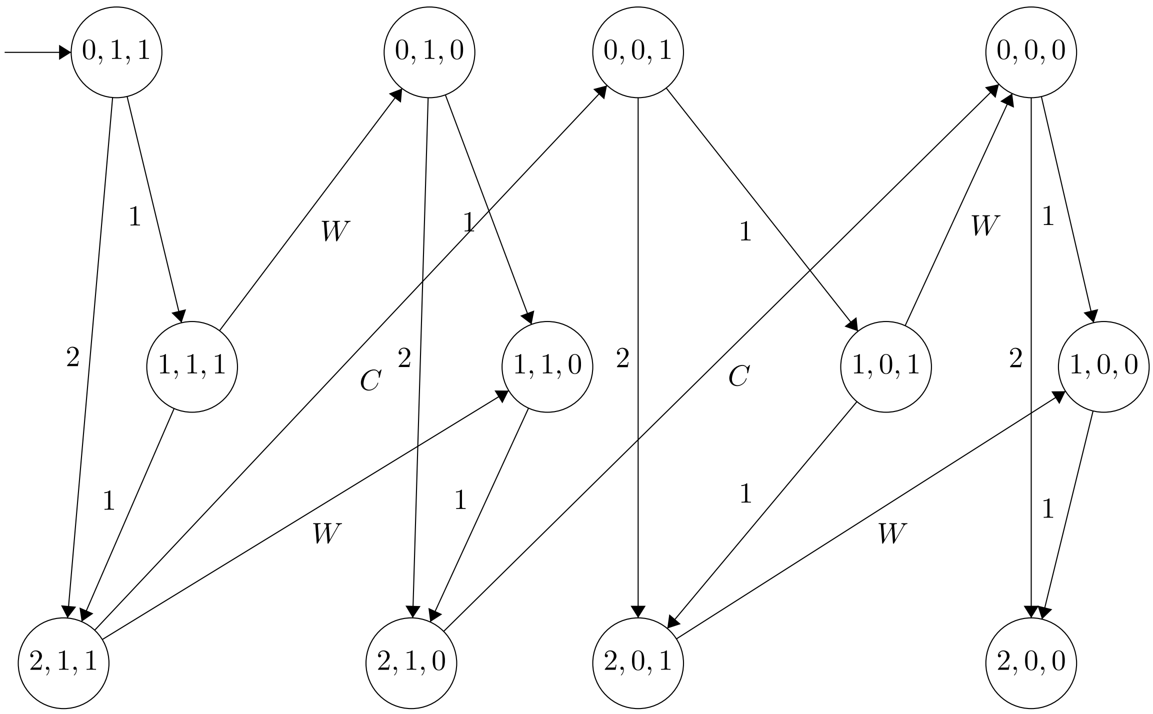
Beispiel: Mustererkennung in Texten: z.B. Suche das Wort "then"



Getränkeautomat (Beispiel für Automat mit Ausgabe): Cola für CHF 2, Wasser für CHF 1. Münzannahme: Münzen zu 1, 2 CHF.



Wenn der Automat nur eine Flasche Cola und eine Flasche Wasser enthält:



3 Formale Sprachen

Ziel: Genaue Beschreibung der Ein- und Ausgaben eines Automaten.

Definition 2 (Alphabet)**.** (endliche, nichtleere Menge von Symbolen). Bsp: Binäres Alphabet $\Sigma_{\text{bin}} = \{0, 1\}$, Tastaturalphabet $\Sigma_{\text{tast}} = \{a, \ldots, z, A, \ldots, Z, 0, \ldots 9, \ldots\}$

Definition 3 (Wort (= Zeichenkette, String))**.** Endliche Folge von Symbolen eines gegebenen Alphabets. Beispiel: 01110 ist ein Wort über Σ_{bin}.

Bemerkung (Eigenschaften von Wörtern)**.** Die da wären:

- Leeres Wort: ε (manchmal λ) = leere Folge von Symbolen (über einem beliebigen Alphabet)
- Länge eines Wortes: $|w|$ bezeichnet die Anzahl der Symbole im Wort w. $|\varepsilon| = 0$.

Bemerkung (Konventionen für Darstellung von Wörtern)**.** Wir sagen:

- $a, b, c, \ldots$ für Buchstaben, Symbole
- $v, w, x, \ldots$ für Wörter

Definition 4 (Potenzen von Wörtern)**.** Es gibt im Angebot:

- Menge aller Wörter einer bestimmten Länge:

$$\begin{aligned}
&\text{Sei } \Sigma \text{ Alphabet, so:} \\
\Sigma^0 &= \{\varepsilon\} \\
\Sigma^1 &= \Sigma \\
\Sigma^2 &= \{ab | a, b \in \Sigma\} \\
\Sigma^i &= \{a_1 \dots a_i | a_1 \dots a_i \in \Sigma\}
\end{aligned}$$

- Menge aller Wörter über Σ:

$$\Sigma^* = \bigcup_{i=0}^{\infty} \Sigma^i, \varepsilon \in \Sigma^*$$

- Menge aller nichtleeren Wörter über Σ:

$$\Sigma^+ = \bigcup_{i=1}^{\infty} \Sigma^i, \varepsilon \notin \Sigma^+$$

Beispiel.
Beispiel:

$$\begin{aligned}
\Sigma &= \{0, 1\} \\
\Sigma^2 &= \{00, 01, 10, 11\} \\
\Sigma^* &= \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, 001, \dots\}
\end{aligned}$$

Definition 5 (Konkatenation (Verkettung) von Wörtern)**.**

$$v = a_1, \dots a_k, w = b_1 \dots, b_k \text{ über } \Sigma \Rightarrow v \cdot w = a_1 \dots a_k b_1 \dots b_k = vw$$

Bemerkung (Rechenregeln für Konkatenation von Wörtern)**.**

$$\begin{aligned}
v \cdot (v \cdot w) &= (u \cdot v) \cdot w \\
|x \cdot y| &= |x| + |y| \\
x \cdot \varepsilon &= \varepsilon \cdot x = x
\end{aligned}$$

Bemerkung (Infixe, Präfixe, Suffixe)**.** Seien $v, w \in \Sigma^*$ für ein Alphabet Σ, dann ist v ein Teilwort (Infix) von w, falls es $x, y \in \Sigma^*$ gibt, so dass $w = x \cdot v \cdot y$; v ist ein Präfix von w, falls es $y \in \Sigma^*$ gibt, so dass $w = v \cdot y$. Suffix funktionert analog.

Beispiel: abc ist Teilwort von aabcc, aa ist Präfix und Suffix von aabcaa. ε ist Teilwort von jedem Wort.

Definition 6 (Sprache)**.** L über Alphabet Σ ist eine Teilmenge von Σ^*, $L \subseteq \Sigma^*$. Sprache ist Menge von Wörtern, kann unendlich gross sein. Leere Sprache: $\varnothing$ enthält keine Wörter, ist über jedem Alphabet definiert. Spezielle Sprache: L_ε ist die Sprache, die nur das leeere Wort enthält. $L_\varepsilon \neq \varnothing$.

Definition 7 (Konkatenation von Sprachen)**.** $L_1, L_2 \subseteq \Sigma^* \Rightarrow L_1 \cdot L_2 = L_1 L_2 = \{v \cdot w | v \in L_1, w \in L_2\}$

Definition 8 (Potenzen von Sprachen)**.**

$$\begin{aligned} L^0 &= L_\varepsilon = \{\varepsilon\} \\ L^{i+1} &= L^i \cdot L \text{ für } i \in \mathbb{N} \end{aligned}$$

Definition 9 (Kleene-Stern)**.**

$$\begin{aligned} L^* &= \bigcup_{i \in \mathbb{N}} L^i \\ L^+ &= \bigcup_{i \in \mathbb{N}} L^i - L_\varepsilon \end{aligned}$$

Beispiel. Sei $\Sigma = \{a, b\}, L_1 = \{a^i | i \geq 0\} = \{\varepsilon, a, aa, aaa, \dots\}, L_2 = \{b^i | i \geq 0\} = \{\varepsilon, b, bb, bbb, \dots\}$:

$L_1 \cdot L_2 = \{\varepsilon, a, b, ab, aab, abb, aaab, \dots\} = \{a^i b^j | i \geq 0, j \geq 0\}$ ist die Menge aller Wörter über $\{a, b\}$, in dem alle as vor allen bs vorkommen.

Übung. Seien $L_1, L_2, L_3 \subseteq \Sigma^*$.
(a) Gilt $(L_1 \cdot L_2) \cdot L_3 = L_1 \cdot (L_2 \cdot L_3)$? Ja.
(b) Gilt $(L_1 \cup L_2) \cdot L_3 = L_1 \cdot L_3 \cup L_2 \cdot L_3$? Ja.
(c) Gilt $(L_1 \cdot L_2) \cup L_3 = (L_1 \cup L_3) \cdot (L_2 \cup L_3)$? Quatsch.
(d) Gilt $L_1 \cdot L_2 = L_2 \cdot L_1$? Quatsch.

Beweis. (a)

$$\begin{aligned} (L_1 \cdot L_2) \cdot L_3 &= \{vw | v \in L_1 \cdot L_2, w \in L_3\} \\ &= \{xyw | x \in L_1, y \in L_2, w \in L_3\} \\ &= \{xz | x \in L_1, z \in L_2 L_3\} \\ &= L_1 \cdot (L_2 \cdot L_3) \end{aligned}$$

(b)

$$\begin{aligned} (L_1 \cup L_2) \cdot L_3 &= \{vw | v \in L_1 \cup L_2, w \in L_3\} \\ &= \{vw | v \in L_1, w \in L_3\} \cup \{vw | v \in L_2, w \in L_3\} \\ &= (L_1 \cdot L_3) \cup (L_2 \cdot L_3) \end{aligned}$$

(c) Gegenbeispiel: $L_1 = L_2 = \{a\}, L_3 = \{b\}$. Daraus folgt:

$$\begin{aligned} L_1 L_2 = \{aa\} &\Rightarrow (L_1 L_2) \cup L_3 = \{aa, b\} \\ L_1 \cup L_3 = \{a, b\} = L_2 \cup L_3 &= \Rightarrow ... \end{aligned}$$

$\square$

Bemerkung. (d) gilt aber für $|\Sigma| = 1$.

Definition 10 (Entscheidungsproblem)**.** Die Eingabe ist eine Sprache L über einem Alphabet Σ sowie ein Wort $w \in \Sigma^*$. Die Ausgabe soll lauten: JA falls $w \in L$ oder NEIN falls $w \notin L$.

Die Modellierung von vielen alltäglichen Berechnungsproblemen im Formalismus der formalen Sprachen.

Beispiel (Primzahltest)**.** Das Alphabet $\Sigma = \{0, 1\}$. Die Sprache

$$L = \{w \in \Sigma^* | w \text{ ist Binärdarstellung einer Primzahl}\}$$

Eine Zahl $p \in N$ ist Primzahl genau dann, wenn $Bin(p) \in L$.
Identifizierung von Sprachen als Probleme.

4 Kurze Einführung in formale Beweise

Behauptungen[2] mit Unanfechtbarkeitsanspruch wollen bewiesen werden. Insbesondere negative Aussagen der Form "Dieses Rechenmodell kann diese Aufgabe nicht lösen" brauchen eine präzise Formulierung und Begründung, um glaubwürdig zu sein[3]. Unkontrollierte Hausaufgabe

Als Beweismethoden haben wir im Angebot:

Deduktion (zum Beweis einer Implikation) Zum Beweis von Wenn-Dann-Aussagen, z.B.

$$\text{Wenn } \underbrace{x \geq 4}_{\text{Hypothese}}, \text{ dann } \underbrace{x^2 \geq 16}_{\text{Konklusion}}$$

Vorgehen: Hypothese als wahr annehmen, dann Folgerungen darauf anwenden, bis die Konklusion folgt.

Beispiel. Sei $x \geq 4$. Es gilt $4^2 = 16$ und die Quadratfunktion ist steigend. Daraus folgt: $x^2 \geq 4^2 = 16$

Hierfür ist es oft hilfreich, Definitionen von Begriffen einzusetzen.

Beispiel. Wenn $L = \{w \in \{a, b\}^* || w| \text{ gerade}\}$, dann gilt für alle $x \in L^2$, dass $|x|$ gerade ist.

Beweis. Sei $x \in L^2$.

Definition von L^2 einsetzen. Es existieren $u, v \in L$, so dass $x = uv$.

Definition von L einsetzen. $|u|$ ist gerade, $|v|$ ist gerade.

$\Rightarrow |x| = |u| + |v| \Rightarrow |x|$ ist gerade. □

[2] 11. September 2012
[3] Hopcraft et al., Abschnitt 1.2–1.4

Doppelte Deduktion (zum Beweis einer Äquivalenz) Zerlegung in zwei Wenn-Dann-Aussagen, wird getrennt bewiesen.

Beispiel (Gleichheit von Mengen)**.**

$$A = B \Rightarrow A \subseteq B \wedge B \subseteq A \Rightarrow A = B$$

Beispiel. Sprachen sind ja auch Mengen. Also: seien $L_1, L_2 \in \Sigma^*$. Dann gilt:

$$L_1 \cdot (L_1 \cup L_2) = L_1^2 \cup L_1 \cdot L_2$$

Beweis. Zweimal:

"$\subseteq$":
Sei $x \in L_1 \cdot (L_1 \cup L_2) \Rightarrow \exists x = uv, u \in L_1, v \in L_1 \cup L_2$.
Fallunterscheidung: (a) $v \in L_1 \Rightarrow x = uv, u \in L_1, v \in L_1 \Rightarrow x \in L_1^2$.
(b) $v \in L_2 \Rightarrow x = uv, u \in L_1, v \in L_2 \Rightarrow x \in L_1 \cdot L_2$.

- "$\supseteq$":
  Sei $x \in L_1^2 \cup L_1 \cdot L_2$.
  Fallunterscheidung: (a) $x \in L_1^2 \Rightarrow x = uv, u, v \in L_1$
  $\Rightarrow v \in L_1 \cup L_2$
  $\Rightarrow x \in L_1 \cdot (L_1 \cup L_2)$.
  (b) $x \in L_1 \cdot L_2 \Rightarrow x = uv, u \in L_1, v \in L_2$
  $\Rightarrow v \in (L_1 \cup L_2)$
  $\Rightarrow x \in L_1 \cdot (L_1 \cup L_2)$

$\square$

Widerspruchsbeweis Es wird eine Annahme getroffen und dann solange gefolgert, bis Quatsch herauskommt. Daraus folgt, dass die Annahme auch falsch sein muss.

Beispiel. Seien $a, b \in \mathbb{N}$ und $a, b \geq 2$. Dann gilt: a teilt nicht $ab + 1$.

Beweis. Seien $a, b \in \mathbb{N}$. Angenommen, a teilt $ab + 1$.

Ziel: Annahme zum Widerspruch führen, dann folgt daraus Negation, also die gewünschte Konklusion.

a teilt $ab \Rightarrow a$ teilt ab und $ab + 1$.

$\Rightarrow a$ teilt $(ab + 1) - (ab) = 1$

$\Rightarrow a = 1$. Bonk! Widerspruch zu $a \geq 2$. $\square$

Beispiel. Es gibt unendlich viele Primzahl (äquivalente Formulierung: Es gibt keine grösste Primzahl).

Beweis. Angenommen, es gäbe eine grösste Primzahl. Wir nennen die Anzahl der Primzahlen k.

Seien $p_1 < p_2 < \cdots < p_k$ die Primzahlen. Betrachten wir $x = p_1 \cdot p_2 \cdot \cdots \cdot p_k + 1$. Da p_k die grösste Primzahl ist und $x > p_k$, kann x keine Primzahl sein.

Also gibt es eine Primfaktorzerlegung von x, und p_l sei die kleinste Primzahl in dieser Zerlegung.

Dann gilt: p_l teilt $p_1 \cdot p_2 \cdot dots \cdot p_k$ und p_l teilt darum auch x.

Daraus folgt: p_l teilt $p_1 \cdot p_2 \cdot dots \cdot p_k + 1$. Aus dem vorhergehenden Beweis (a teilt $ab + 1$) folgt, dass das ein Widerspruch ist. □

Induktion Vor allem verwendet für Beweise über natürliche Zahlen. Ziel: Zeige, dass die Aussage $S(n)$ für alle $n \geq n_0$ gilt.

Methode:

1. Induktionsanfang (Anker). Zeige $S(n_0)$.
2. Induktionsschritt. Zeige $S(i) \Rightarrow S(i + 1)$.

Idee: Wenn $S(n_0)$ gilt und $S(i) \Rightarrow S(i + 1)$ für alle $i \geq n_0$, dann folgt aus $S(n_0)$ und $S(n_0) \Rightarrow S(n_0 + 1)$, dann gilt $S(n_0 + 1)$. Undsoweiter, undsofort.

Beispiel (Der kleine Gauss)**.** . Für alle $n \geq 1$ gilt: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

Beweis. Anker: $\sum_{i=1}^{1} = \frac{1+(1+1)}{2}$

Schritt: es gelte $\sum_{i=1}^{k} i = \frac{k(k+1)}{2}$ (Induktionsvoraussetzung).

Zeige: $\sum_{i=1}^{k+1} i = \frac{(k+1)(k+1+1)}{2}$ (Induktionsbehauptung).

$$\begin{aligned}
\sum_{i=1}^{k+1} i &= \sum_{i=1}^{k} i + (k+1) \\
&= \frac{k(k+1)}{2} + (k+1) \\
&= \frac{k(k+1) + 2(k+1)}{2} \\
&= \frac{(k+2)(k+1)}{2} \\
&= \frac{(k+1+1)(k+1)}{2}
\end{aligned}$$

□

Rekursive Definitionen und strukturelle Induktion

Beispiel (Rekursive Definition von arithmetischen Ausdrücken)**.** Basis: Jede Zahl und jede Variable ist ein arithmetischer Ausdruck: $(a+3)\cdot 5$ ist ein arithmetischer Ausdruck, $(a+3\cdot 5$ [sic] ist keiner (Klammerfehler).

Rekursion: Wenn E und F Ausdrücke sind, dann definieren wir, dass dann auch $E+F$, $E\cdot F$, (E) arithmetische Ausdrücke sind.

Behauptung: Jeder Ausdruck enthält gleichviele linke wie rechte Klammern.

Beweis. Mit struktureller Induktion:

1. I.A.: Basisausdrücke haben keine Klammern. Daraus folgt: gleichviele linke wie rechte Klammern (0).
2. Sei G ein arithmetischer Ausdruck ($G = E{+}F, E{\cdot}F, (G)$). Fallunterscheidung:
   (a) $G = E + F$: Induktionsvoraussetzung: E und F haben gleich viele linke wie rechte Klammern. Gilt also auch für G, da keine Klammern hinzukommen.
   (b) $G = E \cdot F$: analog.
   (c) (G): Induktionsvoraussetzung: E hat gleich viele linke wie rechte Klammern. Gilt also auch für G, da 1 linke und 1 rechte Klammer hinzukommt.

□

5 Unendlichkeit

Beweis für 3.7. Behauptung: $|A| < |B| \Leftarrow |A| = |C| \wedge C \subset B$
$A = B = \mathbb{N} = \{0,1,2,3,\dots\}$ $C = \{1,2,3,4,\dots\}$ $(0,1),(1,2),\dots,(k,k+1)$ □

Aufgaben: 3.7, 3.8, 3.11, 3.17, 3.18, 3.19

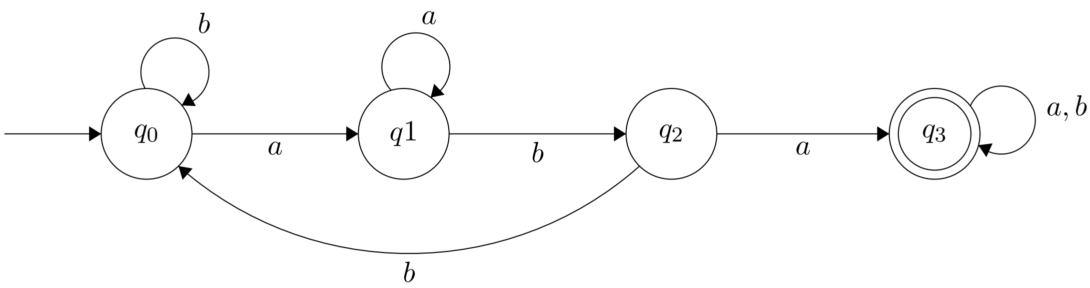
6 Endliche Automaten

Unterscheidung:

Deterministisch Automat kann zu einem Zeitpunkt nur einen Zustand annehmen.

Nichtdeterministisch Automat kann gleichzeitig mehrere Zustände besitzen.

Beispiel. DEA zum Erkennen des Musters aba:



Zustände speichern das bereits gelesene Teilmuster:

- q_0: noch nichts vom Teilwort aba gelesen.
- q_1: zumindest schon das Wort a gelesen.
- q_2: schon ab gelesen.
- q_3: das ganze Muster gelesen, es kann also akzeptiert werden.

Transitionen (Zustandsübergänge£) beschreiben die Änderung beim Lesen eines Zeichens des Wortes.

Definition 11 (Formale Definition (nochmal))**.** Ein DEA A wird beschrieben durch $A = (Q, \Sigma, \delta, q_0, F)$, wobei:

- Q ist eine endliche Menge von Zuständen,
- Σ ist das Alphabet für die Eingabe,
- q_0 ist der Startzustand, $q_0 \in Q$,
- F ist die Menge der akzeptierenden Zustände (Endzustände), $F \subseteq Q$,
- $\delta : Q \times \Sigma \mapsto Q$ ist die Übergangsfunktion (Transitionsfunktion), die für jeden Zustand und jedes Eingabesymbol einen eindeutigen Folgezustand definiert.

Beispiel (von vorher)**.**

$$A = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_3\}),$$

wobei δ gegeben ist durch:

$$
\begin{aligned}
\delta(q_0, a) &= q_1, \\
\delta(q_0, b) &= q_0, \\
\delta(q_1, a) &= q_1, \\
\delta(q_1, b) &= q_2, \\
\delta(q_2, a) &= q_3, \\
\delta(q_2, b) &= q_0, \\
\delta(q_3, a) &= q_3, \\
\delta(q_3, b) &= q_3
\end{aligned}
$$

(oder als Transitionstabelle aufgeschrieben):

	a	b
$\mapsto q_0$	q_1	q_0
q_1	q_1	q_2
q_2	q_3	q_0
$*q_3$	q_3	q_3

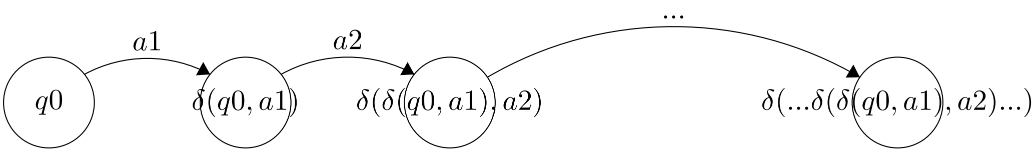
Die Transitionstabelle enthält alle Informationen über einen DEA, meistens ist aber die graphische Darstellung übersichtlicher.

Ziel: Automaten zur Beschreibung von Sprachen verwenden.

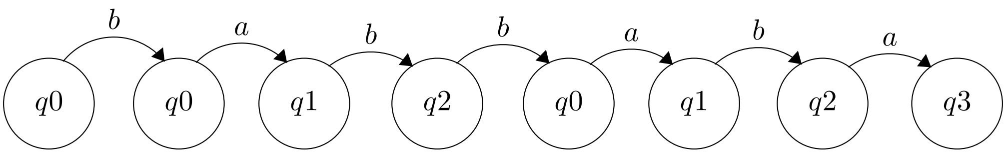
Definition 12 (Berechnung)**.** Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA, $w \in \Sigma^*$ ein Wort.

Die Berechnung von A auf w ist die Folge von Zuständen, die A beim Lesen von w durchläuft, also, gilt für $w = a_1 a_2 \dots a_k$:

$$q_0 \overset{a_1}{\curvearrowright} \delta(q_0, a_1) \overset{a_2}{\curvearrowright} \delta(\delta(q_0, a_1), a_2) \overset{\dots}{\curvearrowright} \delta(\dots \delta(\delta(q_0, a_1), a_2) \dots)$$



Beispiel. Berechnung von A auf $w = babbaba$:



Sinnvollerweise: Erweiterung von δ af Wörter (erweiterte Übergangsfunktion).

Definition 13 (Erweiterte Übergangsfunktion)**.** $\hat{\delta} : Q \times \Sigma^* \mapsto Q$ ist rekursiv definiert wie folgt:

- $\hat{\delta}(q, a) = \delta(q, a)$ für alle $q \in Q, a \in \Sigma$
- $\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$ für alle $q \in Q, x \in \Sigma^*, a \in \Sigma$.

Bemerkung. Anschaulich: $\hat{\delta}(q, w)$ betstimmt, in welchem Zustand A endet, wenn er vom Zustand q aus das Wort w liest.

Beispiel. $\hat{\delta}(q_0, babbaba)$

Definition 14. Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA. Die von A akzeptierte Sprache $L(A) \subseteq \Sigma^*$ ist die Menge aller $w \in \Sigma^*$, so dass $\hat{\delta}(q_0, w) \in F$.

Bemerkung. Anschaulich: Die Menge aller Wörter, die den DEA vom Anfangszustand aus in einen Endzustand führen.

Beispiel. $\hat{\delta}(q_0, babbaba) = q_3 \in F \Rightarrow w \in L(A)$ (w wird von A akzeptiert.)

Definition 15 (Reguläre Sprachen)**.** $\mathcal{L}(\text{DEA}) = \{L(A) | A \text{ ist ein DEA}\}$ ist die Klasse der Sprachen, die von DEAs akzeptiert werden. Man nennt sie die Klasse der regulären Sprachen und $L \in \mathcal{L}(\text{DEA})$ wird regulär genannt.

Beispiel. DEA A entwerfen, der die Sprache

$$L = \{w \in \{0, 1\}^* | w \text{ enthält eine gerade Anzahl Nullen und eine gerade Anzahl Einsen}\}$$

akzeptiert.

Idee: 4 Zustände, die speichern, wieviele Nullen und Einseln modulo 2 bereits gelesen wurden.

- q_{00}: gerade Anzahl Nullen, gerade Anzahl Einsen
- q_{01}: gerade Anzahl Nullen, ungerade Anzahl Einsen
- q_{10}: ungerade Anzahl Nullen, gerade Anzahl Einsen
- q_{11}: ungerade Anzahl Nullen, ungerade Anzahl Einsen

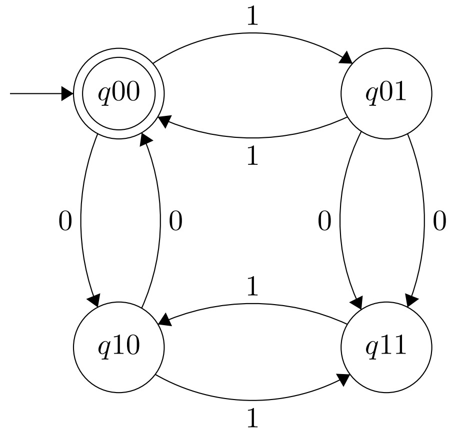
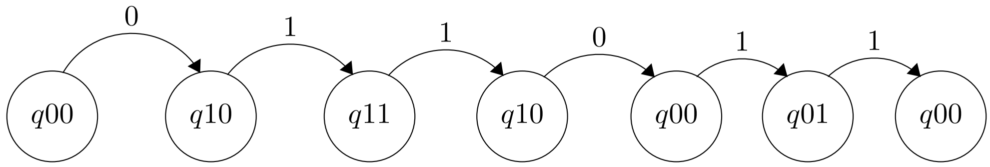


Tabelle:

	0	1
$\mapsto *q_{00}$	q_{10}	q_{01}
q_{01}	q_{11}	q_{00}
q_{10}	q_{00}	q_{11}
q_{11}	q_{01}	q_{10}

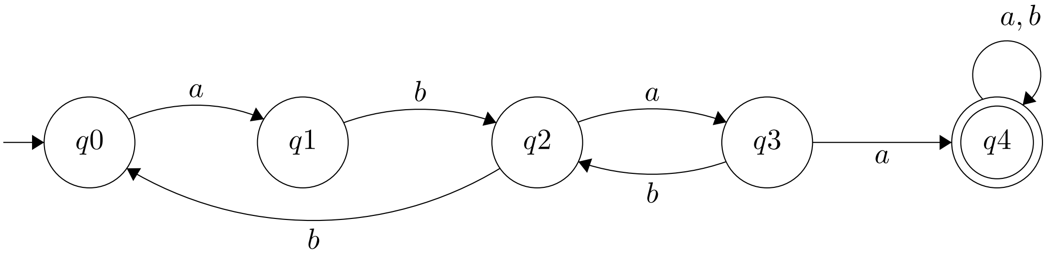
Berechnung von A auf $w = 011011$:



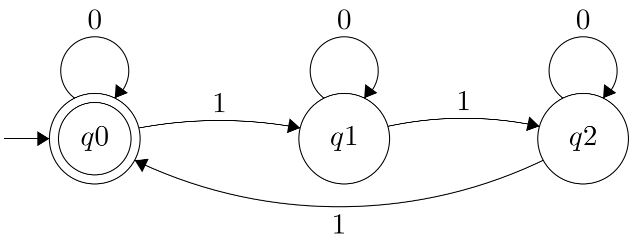
$\Rightarrow w \in L(A), \hat{\delta}(q_{00}, 011011) = q_0$

Lösung der Aufgaben

1a:



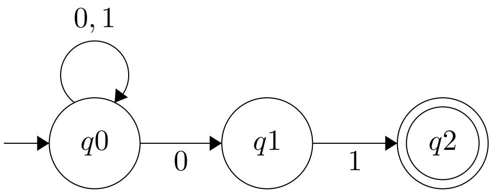
1b:



7 Nichtdeterministische erstaunliche Endliche Automaten (NEA)

Idee: Ein NEA kann sich nach dem Lesen eines Wortes in mehreren Zuständen befinden.
Vorteil: Sie sind einfacher zu entwerfen.

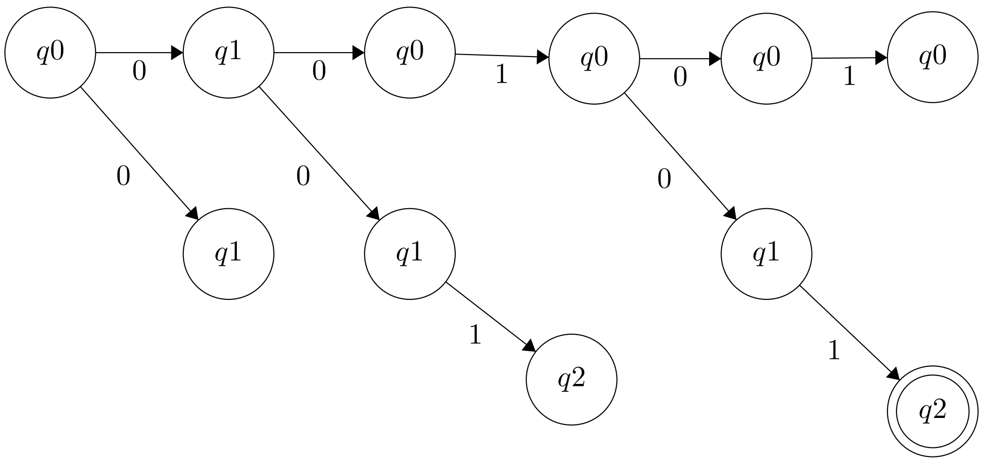
Beispiel. Akzeptiere alle Wörter, die mit 01 enden.



Transitionstabelle:

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

Mögliche Berenchungen auf dem Wort 00101:



Definition 16 (NEA)**.** Ein NEA A wird beschrieben durch $A = (Q, \Sigma, \delta, q_0, F)$, wobei

$$\delta : Q \times \Sigma \mapsto 2^Q = \mathcal{P}(Q)$$

Die Transitionsfunktion definiert für jeden Zustand und jedes Eingabesymbol eine Menge von Folgezuständen.

Definition 17 (Erweiterte NEA-Transitionsfunktion)**.** Sei $w \in \Sigma^*, q \in Q$. Dann definieren wir

$$\hat{\delta} : Q \times \Sigma^* \mapsto 2^Q$$

rekursiv wie folgt:

- $\hat{\delta}(q, \varepsilon) = \{q\}$
- Sei $w = xa$ und $\hat{\delta}(q, x) = \{p_1, \ldots p_k\}$ für $k \in \mathbb{N}$, dann ist

$$\hat{\delta}(q, w) = \bigcup_{i=1}^{k} \delta(p_i, a)$$

Beispiel.

$$\begin{aligned}
\hat{\delta}(q_0, \varepsilon) &= \{q_0\} \text{ (Grundregel)} \\
\hat{\delta}(q_0, 0) = \delta(q_0, 0) &= \{q_0, q_1\} \\
\hat{\delta}(q_0, 00) = \delta(q_0, 0) \cup \delta(q_1, 0) = \{q_0, q_1\} \cup \varnothing &= \{q_0, q_1\} \\
\hat{\delta}(q_0, 001) = \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_0\} \cup \{q_2\} &= \{q_0, q_2\} \\
\hat{\delta}(q_0, 0010) = \delta(q_0, 0) \cup \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_2\} &= \{q_0, q_2\} \text{ falsch} \\
\hat{\delta}(q_0, 00101) = \delta(q_0, 0) \cup \delta(q_0, 1) \cup \delta(q_1, 1) = \{q_1\} \cup \{q_2\} &= \{q_0, q_2\} \text{ falsch}
\end{aligned}$$

Definition 18. Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein NEA. Die von A akzeptierte Sprache $L(A)$ ist $L(A) = \{w \in \Sigma^* | \hat{\delta}(q_0, w) \cap F \neq \varnothing\}$, also die Menge aller Wörter, mit denen vom Startzustand aus ein Endzustand erreichbar ist.

Fragen/Probleme:

1. Wie kann man möglichst effizient entscheiden, ob ein gegebenes Wort in der Sprache eines gegebenen NEA enthalten ist?

2. Sind NEAs mächtiger als DEAs?

Antwort: DEAs und NEAs sind äquivalent.
Ziel: Automatische Umwandlung von NEAs in DEAs.

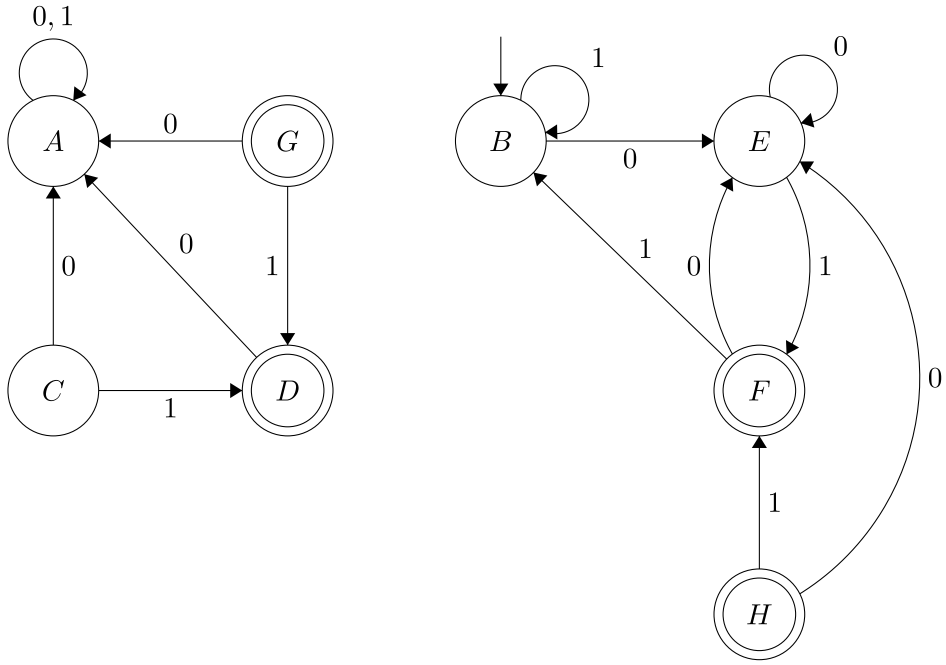
Bemerkung (Teilmengenkonstruktion (Potenzmengenkonstruktion))**.** Idee: Zustände des DEA sind Mengen erreichbarer Zustände des NEA.

Formal: Sei $N = (Q_N, \Sigma, \delta_N, q_0, F_N)$ ein NEA. Konstruiere einen äquivalenten DEA $D = (Q_D, \Sigma, \delta_D, \{q_0\}, F_D)$ mit

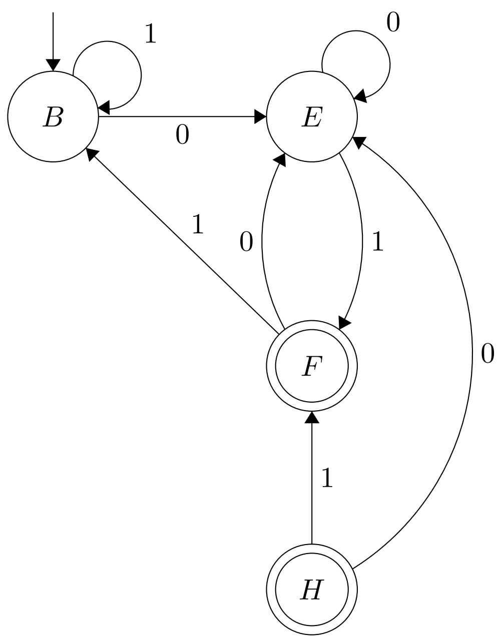
- $Q_D = 2^{Q_N}$
- $F = \{S \subseteq Q_D | S \cap F \neq \varnothing\}$
- $\delta_D(S, a) = \bigcup_{p \in S} \delta_N(p, a)$ für $S \in Q_D, a \in \Sigma$

Beispiel (von vorher)**.** Konstruktion des äquivalenten DEAs als Übergangstabelle.

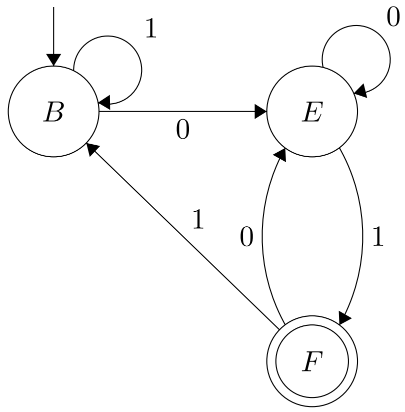
		0	1
A	$\varnothing$	$\varnothing = A$	$\varnothing = A$
B	$\mapsto \{q_0\}$	$\{q_0, q_1\} = E$	$\{q_0\} = B$
C	$\{q_1\}$	$\varnothing = A$	$\{q_2\} = D$
D	$*\{q_2\}$	$\varnothing$	$\varnothing$
E	$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
F	$*\{q_0, q_2\}$	$\{q_0, q_1\}$	$\{q_0\}$
G	$*\{q_1, q_2\}$	$\varnothing$	$\{q_2\}$
H	$*\{q_1, q_2, q_3\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$



Der linke Teil ist nicht erreichbar:



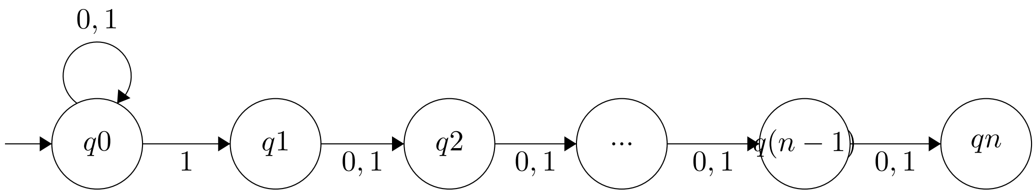
H ist ebenfalls nicht erreichbar:



Beispiel (Ein ungünstiger Fall für die Teilmengenkonstruktion)**.**

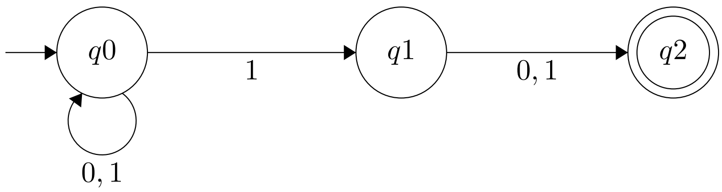
$$\begin{aligned} L_n &= \{w \in \{0,1\}^* | \text{ Das } n\text{-t-letzte Zeichen von } w \text{ ist eine } 1\} \\ &= \{w \in \{0,1\}^* | w = x1y \text{ mit } x \in \{0,1\}^*, y \in \{0,1\}^{n-1}\} \end{aligned}$$

NEA für L_n mit $n+1$ Zuständen:



Aber jeder DEA für L_n braucht mindestens 2^n Zustände, weil der DEA sich an die letzten n gelesenen Symbole erinnern muss, da er nicht weiss, wann das Wortende kommt. D.h. 2^n verschiedene Teilworte mpssen unterscheiden werden. D.h. 2^n Zustände nötig.

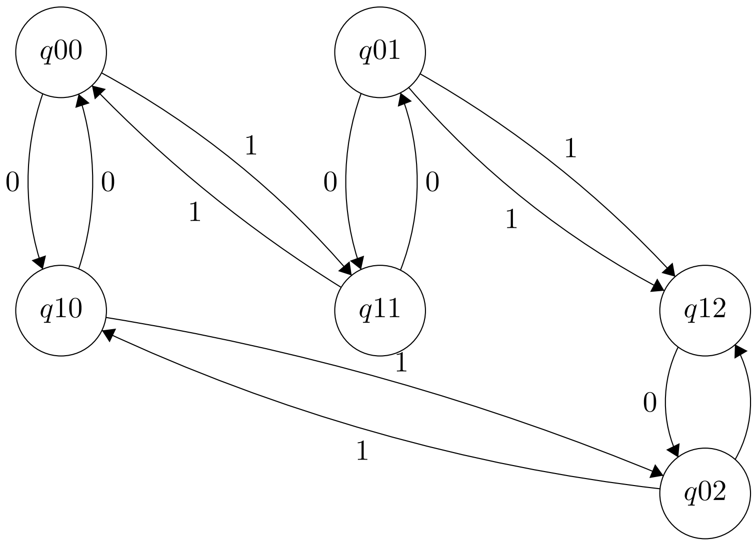
Beispiel mit $n = 2$:



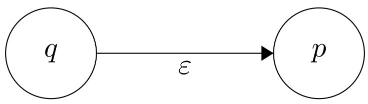
Beispiel (Lösung der Knobelaufgabe)**.** Wir machen uns zunutze:

$$2^k \mod 3 = \begin{cases} 1 \text{ falls } k \text{ gerade} \\ 1 \text{ falls } k \text{ gerade} \end{cases}$$

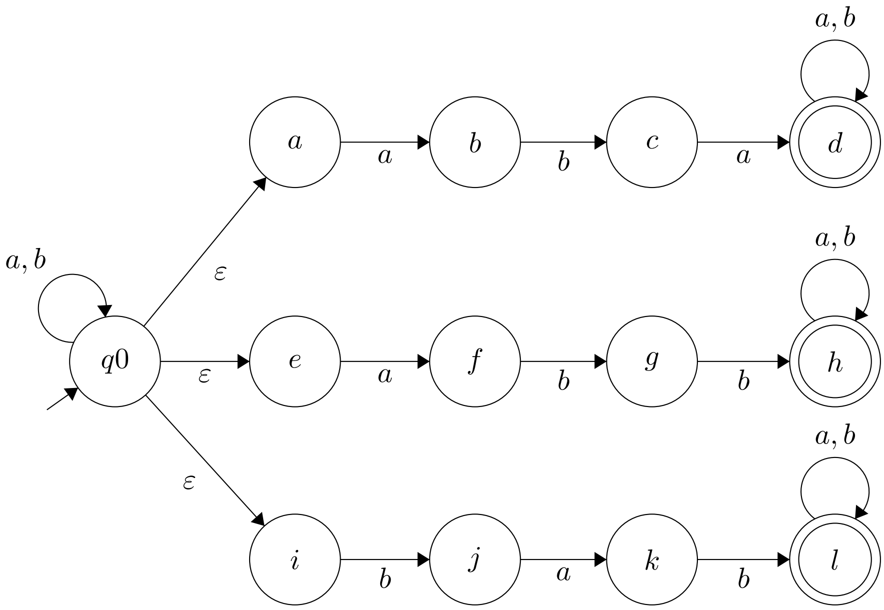
Wir haben Zustände q_{ij}. i ist der Exponent der nächsten 2er-Potenz modulo 2, j ist die Teilsumme.



8 NEAs mit ε-Übergängen



Beispiel. Suche nach mehreren Mustern. $L = \{w \in \{a,b\}^* |$ enthält eines der Teilwörter aba, abb, bab $\}$
Der NEA N mit $L(N) = L$ mit ε-Übergängen sieht aus:

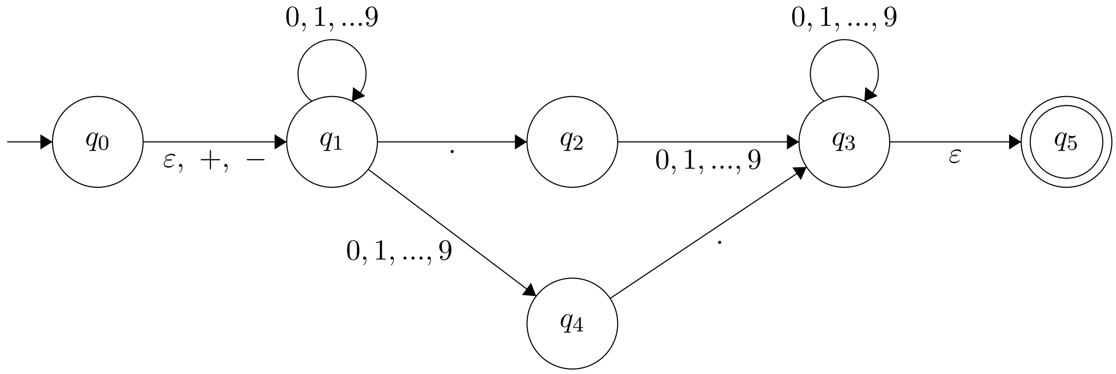


Beispiel. Ein ε-NEA, der Dezimalzahlen akzeptiert, die sich zusammensetzen aus:

- optionales $+$ oder $-$
- Zeichenreihe von Ziffern
- Dezimalpunkt
- Zeichenreihe von Ziffern

Eine der beiden Zeichenreihen von Ziffern darf leer sein.

Der Automat E sieht dann so aus:



Definition 19 (NEA mit ε-Übergängen)**.** Ein NEA mit ε-Übergängen A wird beschrieben durch $A = (Q, \Sigma, \delta, q_0, F)$, wobei Q, Σ, q_0, F wie beim DEA definiert sind, und

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \mapsto 2^Q$$

die Transitionsfunktion ist.

Bemerkung. Annahme: $\varepsilon \notin \Sigma$

Beispiel (von vorher)**.**

$$E = (\{q_0, q_1, \ldots, q_5\}, \{., +, -, 0, 1, \ldots, 9\}, \delta, q_0, \{q_5\})$$

Übergangstabelle:

	ε	$+, -$	$.$	$0, 1, \ldots, 9$
q_0	$\{q_1\}$	$\{q_1\}$	$\varnothing$	$\varnothing$
q_1	$\varnothing$	$\varnothing$	$\{q_2\}$	$\{q_1, q_4\}$
q_2	$\varnothing$	$\varnothing$	$\varnothing$	$\{q_3\}$
q_3	$\varnothing$	$\varnothing$	$\{q_3\}$	$\varnothing$
q_4	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
q_5	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

9 ε-Hüllen

Ziel: Zeigen, dass es zu jedem ε-NEA einen äquivalenten DEA gibt.

Idee: Finde für jeden Zustand die Menge aller Zustände, die von dort aus nur mit ε-Übergängen erreichbar sind.

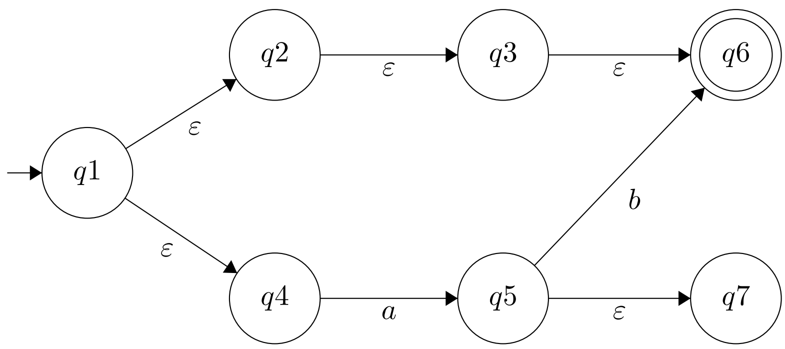
Definition 20 (Rekursive Definition)**.** Die ε-Hülle eines Zustands $q \in Q$ wird bezeichnet mit ε-Hülle(q) und ist definiert durch:

- $q \in \varepsilon$-Hülle(q)
- Falls $p \in \varepsilon$-Hülle(q), dann auch jeder Zustand r, so dass $r \in \delta(p, \varepsilon)$

Beispiel (Von vorher)**.**

$$\begin{array}{rcl} \varepsilon\text{-Hülle}(q_0) & = & \{q_0, q_1\} \\ \varepsilon\text{-Hülle}(q_1) & = & \{q_1\} \\ \varepsilon\text{-Hülle}(q_2) & = & \{q_2\} \\ \varepsilon\text{-Hülle}(q_3) & = & \{q_3, q_5\} \\ \varepsilon\text{-Hülle}(q_4) & = & \{q_4\} \\ \varepsilon\text{-Hülle}(q_5) & = & \{q_5\} \end{array}$$

Beispiel (Noch eines)**.** . Noch eines:



Bestimme E-Hülle(q1):

1. $q_1 \in \varepsilon - Hulle(q_1)$
2. Finde alle Zustände aus $\delta(q_1, \varepsilon) = \{q_2, q_4\} \Rightarrow \{q_2, q_4\} \subseteq \varepsilon - Hulle$

Definition 21 (ε-Hülle einer Menge S von Zuständen)**.**

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

10 Erweiterung der Übergangsfunktion auf Wörter

Finde alle Pfade im Graphen, die mit dem Wort w beschriftet sind (mit beliebig vielen ε dazwischen).

Definition 22 (Rekursive Definition)**.** Nun denn:

- $\hat{\delta}(q, \varepsilon) = \varepsilon\text{-Hülle}(q)$
- $\hat{\delta}(q, xa)$ wird für $x \in \Sigma^*$ und $a \in \Sigma$ definiert durch:
  1. Sei $\hat{\delta}(a, x) = \{p_1, \dots, p_k\}$
  2. Sei $\bigcup\limits_{i=1}^{k} \delta(p_i, a) = \{r_1, \dots, r_m\}$
  3. Dann gilt $\hat{\delta}(q, xa) = \varepsilon\text{-Hülle}(\{r_1, \dots, r_m\}) = \bigcup\limits_{i=1}^{m} \varepsilon\text{-Hülle}(r_i)$

Beispiel ($\hat{\delta}(q_0, 5{,}6)$)**.** hmm...

1. $\hat{\delta}(q_0, \varepsilon) = \varepsilon\text{-Hülle}(q_0) = \{q_0, q_1\}$
2. $\hat{\delta}(q_0, 5)$: $\delta(q_0, 5) \cup \delta(q_1, 5) = \varnothing \cup \{q_1, q_4\}$
   $\hat{\delta}(q_0, 5) = \varepsilon\text{-Hülle}(q_1) \cup \varepsilon\text{-Hülle}(q_4) = \{q_1, q_4\}$
3. $\hat{\delta}(q_0, 5.)$: $\delta(q_1, .) \cup \delta(q_4, .) = \{q_2\} \cup \{q_3\} = \{q_2, q_3\}$
   $\hat{\delta}(q_0, 5.) = \varepsilon\text{-Hülle}(q_2) \cup \varepsilon\text{-Hülle}(q_3) = \{q_2, q_3, q_5\}$
4. $\hat{\delta}(q_0, 5.6)$: $\delta(q_2, 6) \cup \delta(q_3, 6) \cup \delta(q_5, 6) = \{q_2\} \cup \{q_3\} = \{q_2, q_3\}$
   $\hat{\delta}(q_0, 5.) = \varepsilon\text{-Hülle}(q_2) \cup \varepsilon\text{-Hülle}(q_3) = \{q_2, q_3, q_5\}$ ...

Definition 23. Für einen ε-NEA $E = (Q, \Sigma, \delta, q_0, F)$ ist $L(E) = \{w | \hat{\delta}(q_0, w) \cap F \neq \varnothing\}$.

11 Umwandlung von ε-NEAs in DEAs

Schritt 1: ε-Übergänge eliminieren

Sei $E = (Q_E, \Sigma, \delta_E, q_0, F_E)$ ein ε-NEA. Konstruiere einen äquivalenten DEA $D = (Q_D, \Sigma, \delta_D, q_{0,D}, F_D)$.

- $Q_0 = 2^{Q_E}$. Alle erreichbaren Zustände sind ε-abgeschlossene Teilmengen, das heisst Mengen $S \subseteq Q_E$, so dass $S = \varepsilon\text{-Hülle}(S)$. D.h. jeder ε-Übergang von einem Zustand $q \in S$ führt zu einem Zustand, der auch in S liegt.
- $q_{0,D} = \varepsilon\text{-Hülle}(q_0)$
- $F_D = \{S \in Q_D | S \cap F_E \neq \varnothing\}$
- $\delta_D(S, a)$ wird für alle $S \in Q_D$ und $a \in \Sigma$ wie folgt berechnet:

1. Sei $S = \{p_1, \ldots, p_k\}$
2. Berechne $\bigcup\limits_{i=1}^{k} \delta(p_i, a) = \{r_1, \ldots, r_m\}$
3. Dann ist $\delta_D(S, a) = \bigcup\limits_{j=1}^{n} \varepsilon\text{-Hüllen}(r_j)$

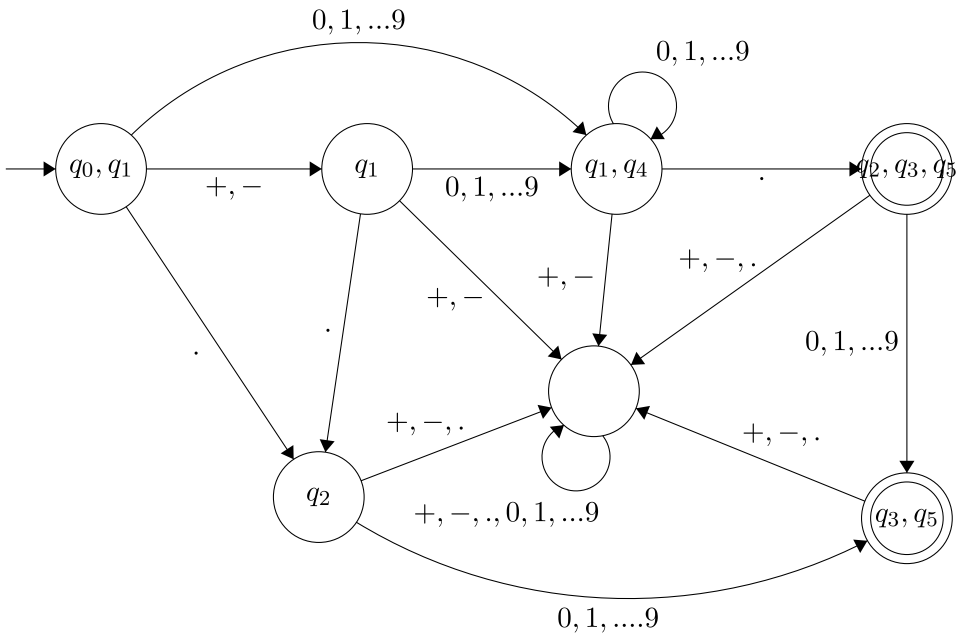
Beispiel (Von vorher)**.** Äquivalenter DEA D: Startzustand ist ε-Hülle$(q_0) = \{q_0, q_1\}$
Finde die Folgezustände von $\{q_0, q_1\}$ für alle Symbole des Alphabets:

$+, -$ Kein Übergang von q_1 aus, aber von q_0 nach q_1. Daraus folgt ε-Hülle$(q_1) = \{q_1\}$.

. Kein Übergang von q_0, aber von q_0 nach q_2. Daraus folgt ε-Hülle$(q_2) = \{q_2\}$.

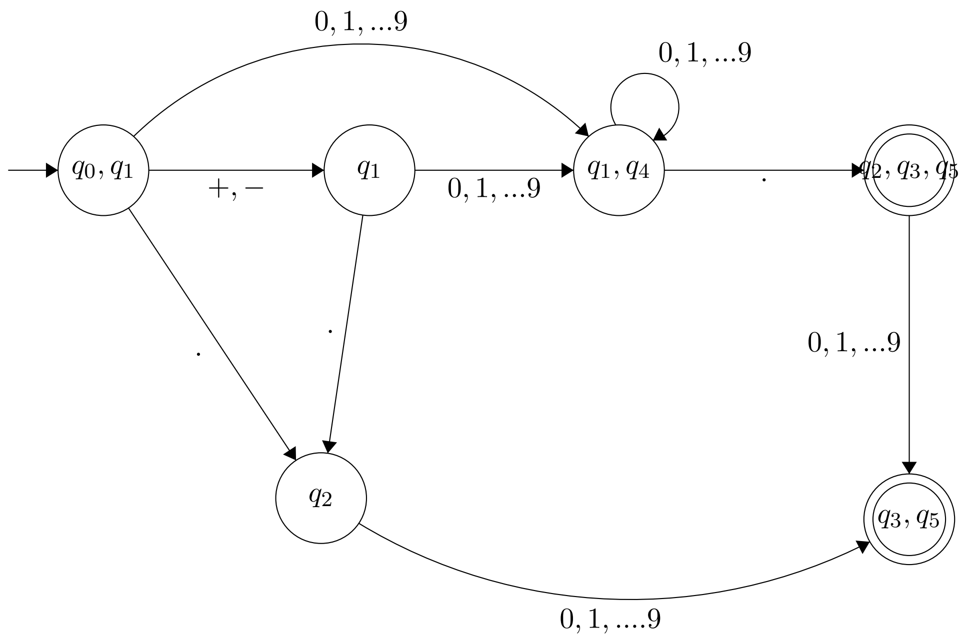
$0, 1, \ldots, 9$ Kein Übergang von q_0, aber von q_1 nach q_1 oder q_4. Daraus folgt ε-Hülle$(q_1, q_4) = \{q_1, q_4\}$

Transitionstabelle (siehe Excel =¿ transferieren)!

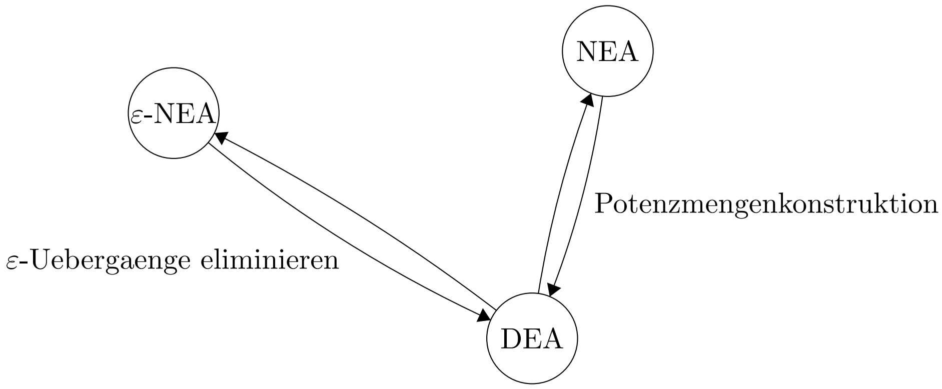


Bemerkung (Konvention)**.** Die da wären:

- Der Zustand $\varnothing$ wird Senkezustand (Abfallzustand) genannt und darf weggelassen werden.
- Darstellung eines DEA mit fehlenden Transitionen sind immer so zu verstehen, dass die fehlenden Transitionen in einen solchen Senkezustand führen:



Zusammenfassung: DEAs und ε-NEAs sind äquivalent.



12 Reguläre Ausdrücke

Besonders geeignet für die Mustersuche, anderer Formalismus zur Beschreibung von regulären Sprachen.

13 Operatoren für reguläre Ausdrücken

- Die Vereinigung von Sprachen L und M, $L \cup M$, ist die Menge aller Zeichenreihen, die entweder in L oder in M oder in beiden enthalten sind.

- Die Verkettung (Konkatenation) von L und M, $L \cdot M$ oder LM, ist die Menge aller Zeichenreihe, gebildet aus einer Verkettung einer Zeichenreihe aus L umit einer beliebigen aus M.

  Formal:
$$LM = \{w \in \Sigma^* | w = uv \text{ mit } u \in L \text{ und } v \in M\}$$

- Die Kleensche Hülle (Stern) von L, L^*, ist die Menge aller Zeichenreihen, die durch die Verkettung einer beliebigen Anzahl von Zeichenreihen aus L gebildet wird.

  Formal:
$$\begin{aligned} L^0 &= \{\varepsilon\} \\ L^1 &= L \\ L^n &= L \cdot L^{n-1} \\ L^* &= \bigcup_{i=0}^{\infty} L^i \end{aligned}$$

  Anschaulich: $w \in L^*$, wenn sich w zerlegen lässt in endlich viele Wörter $w = w_1 w_2 \ldots w_k$ für ein $k \in \mathbb{N}$, so dass $w_1 \in L_1, w_2 \in L_2 \ldots w_i L_i$ gilt.

  Spezialfälle:

  - $\varnothing^* = \{\varepsilon\}$, da $\varnothing^0 = \{\varepsilon\}$
  - $\{\varepsilon\}^* = \{\varepsilon\}$

  Aber: $\varnothing^i = \varnothing$ für $i > 1$.

Definition 24 (Reguläre Ausdrücke)**.** Reguläre Ausdrücke lassen sich wie folgt definieren:

1. ε und $\varnothing$ sind reguläre Ausdrücke.

2. Jedes Symbol $a \in \Sigma$ ist ein regulärer Ausdruck.

3. Wenn α und β reguläre Ausdrücke sind, dann sind:

   - $(\alpha \cdot \beta)$ (Konkatenation)
   - $(\alpha + \beta)$ (Vereinigung)

   reguläre Ausdrücke.

4. Wenn α ein regulärer Ausdruck ist, dan ist auch $(\alpha)^*$ ein regulärer Ausdruck. (Kleenscher Stern)

Bemerkung (Konvention)**.** Man tut:

- Überflüssige Klammern weglassen

- Stern bindet stärker als Verkettung

- Verkettung bindet stärker als Vereinigung

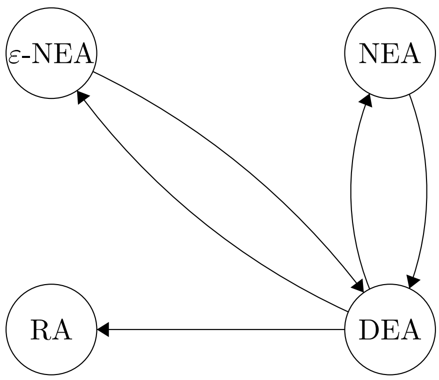
14 Bedeutung regulärer Ausdrücke

$L(\alpha)$ bezeichnet die durch den regulären Ausdruck beschriebene Sprache. Dabei gilt:

- $L(\varepsilon) = \{\varepsilon\}$
- $L(\varnothing) = \varnothing$
- $L(a) = \{a\}$
- $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$
- $L(\alpha + \beta) = L(\alpha) \cup L(\beta)$
- $L(\alpha^*) = (L(\alpha))^*$

Beispiel (Beschreibung von Sprachen durch reguläre Ausdrücke)**.** Nun:

1. $L_1 = \{w \in \{0,1\}^* | w$ beginnt mit 011 und enthält das Teilwort 00 $\}$. RA für L_1: $011 \cdot (0+1)^* \cdot 00 \cdot (0+1)^*$
2. $L_2 = \{w \in \{a,b,c\}^* || w|_a$ ist gerade $\}$.
   RA für L_2: $((b+c)^*a(b+c)^*a(b+c)^*)^*$
3. $L_3 = \{w \in \{0,1\}^* |$ In w kommen die Nullen und Einsen immer abwechselnd vor$\}$
   RA für L_3: $(01)^* + (10)^* + 1(01)^* + 0(10)^* = (\varepsilon + 1)(01)^*(\varepsilon + 0)$



Lösung der Aufgabe 6 (Aufgabenblatt 2):

a) $(a+b)^* : \{a,b\}^*$.
   $(a^*b^*) : \varepsilon \in \mathcal{L}(a^*)$ und $\varepsilon \in \mathcal{L}(b^*)$. $\Rightarrow a \in \mathcal{L}(a^*b^*)$ und $b \in \mathcal{L}(a^*b^*)$

15 Anwendungsbeispiel für RA

Spezifikation einer Eingabemaske für Kontostände. Bedingungen:

- Währungseingabe: CHF, EUR, USD
- optionales Vorzeichen vor dem Betrag: $\oplus$, $\ominus$
- Betrag ganzzahlig oder mit Dezimalpunkt und zwei Nachkommastellen
- keine führende Nullen

$\Sigma = \{C, D, E, F, H, R, S, U, \oplus, \ominus, ., 0, 1, \ldots, 9\}$
Idee: Setze den RA aus Teilen zusammen:
zahlung = waehrung vorzeichen betrag
waehrung = $(CHF + EUR + USD)$
vorzeichen = $(\varepsilon + \oplus + \ominus)$
betrag = ganzzahl (ε + .nachkomma)
ganzzahl = $(0 + (1 + 2 + \cdots + 9)(0 + 1 + \cdots + 9)^*$
nachkomma = $(0, 1, \ldots, 9) + (0, 1, \ldots, 9)$
$\Rightarrow$ zahlung = $(CHF+EUR+USD)$ $(\varepsilon+\oplus+\ominus)$ $(0+(1+2+\cdots+9)(0+1+\cdots+9)^*$ $(\varepsilon + .(0,1,\ldots,9)+(0,1,\ldots,9))$

16 Eine ganz neue Idee

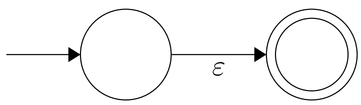
Theorem 1. Reguläre Ausdrücke und endliche Automaten sind äquivalent:

1. für jeden RA α existiert ein DEA A_α mit $\mathcal{L}(\alpha) = \mathcal{L}(A_\alpha)$
2. für jeden DEA A existiert ein RA α_A mit $\mathcal{L}(A) = \mathcal{L}(\alpha_A)$

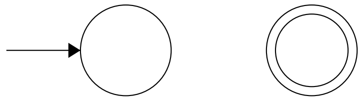
Beweis für 1. Nun. Umwandlung von RA in DEA. Vorgehensweise: RA $\mapsto$ ε-NEA$\mapsto$ DEA. Für einen gegebenen RA konstruiere einen ε-NEA induktiv entsprechend des regulären Ausdrucks des RA mit folgenden zusätzlichen Eigenschaften:

1. genau ein akzeptierender Zustand
2. keine Transitionen zum Startzustand
3. keine Transitionen vom akzeptierenden Zustand wegführende Transitionen

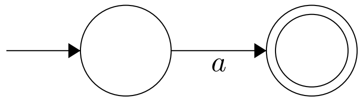
Induktionsanfang: ε-NEAfür ε:



ε-NEAfür $\varnothing$:



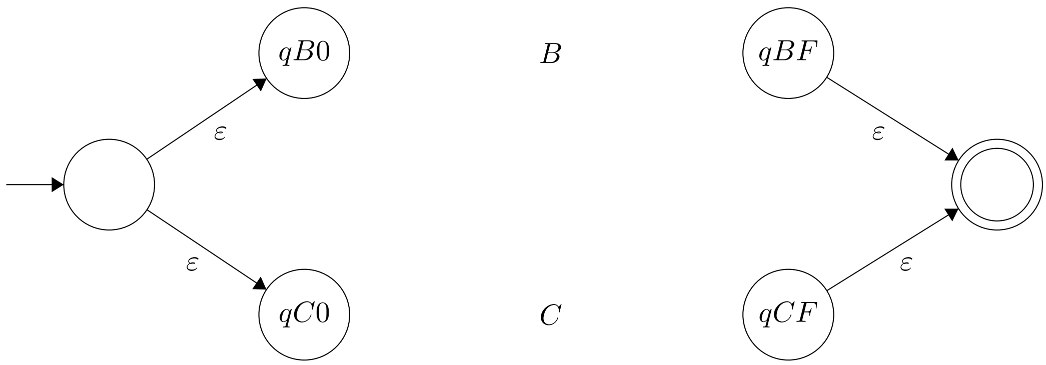
ε-NEAfür $a \in \Sigma$:



Induktionsschritt:

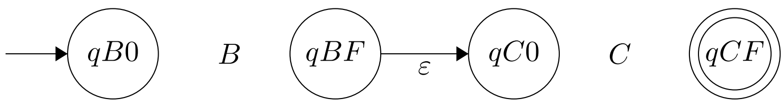
- Sei $\alpha = \beta + \gamma$ ein RA, seien B und C ε-NEAs für β und γ wie oben und mit Startzuständen q_{B0} und q_{C0} und akzeptierenden Zuständen q_{BF} und q_{CF}.

  Konstruiere hieraus einen ε-NEAfür α wie folgt:



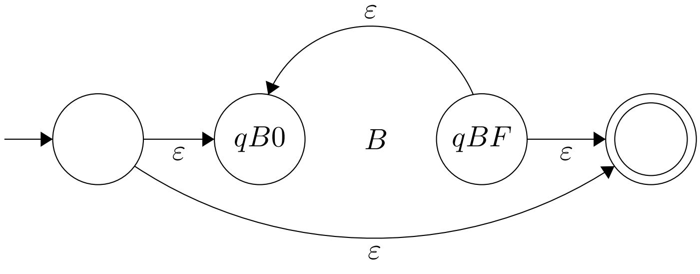
  erfüllt alle Bedingungen.

- Seien $\alpha = \beta \cdot \gamma$ und B, C ε-NEAs für β, γ. ε-NEAfür α:



  erfüllt alle Bedingungen.

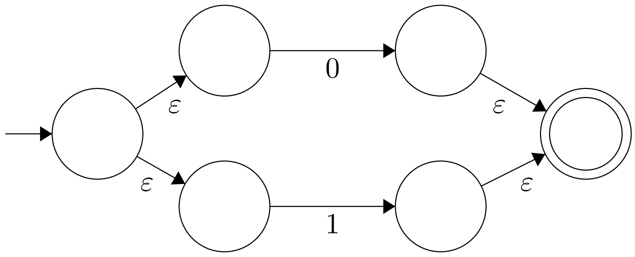
- Sei $\alpha = \beta^*$, B ein ε-NEAfür β. ε-NEAfür α:



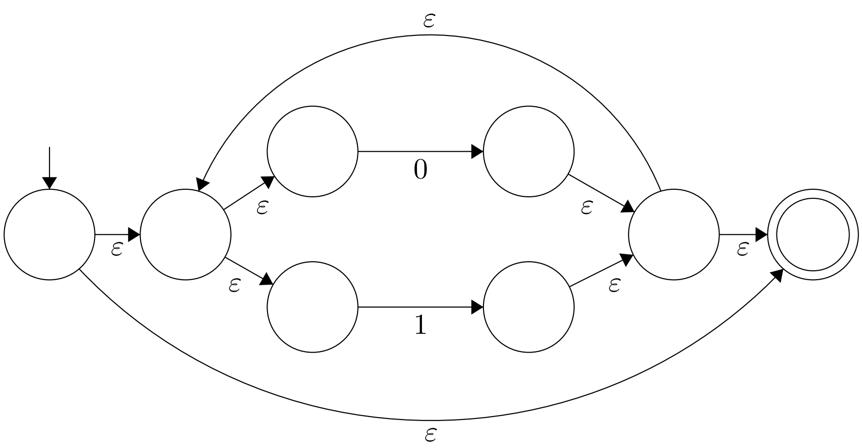
erfüllt alle Bedingungen.

$\square$

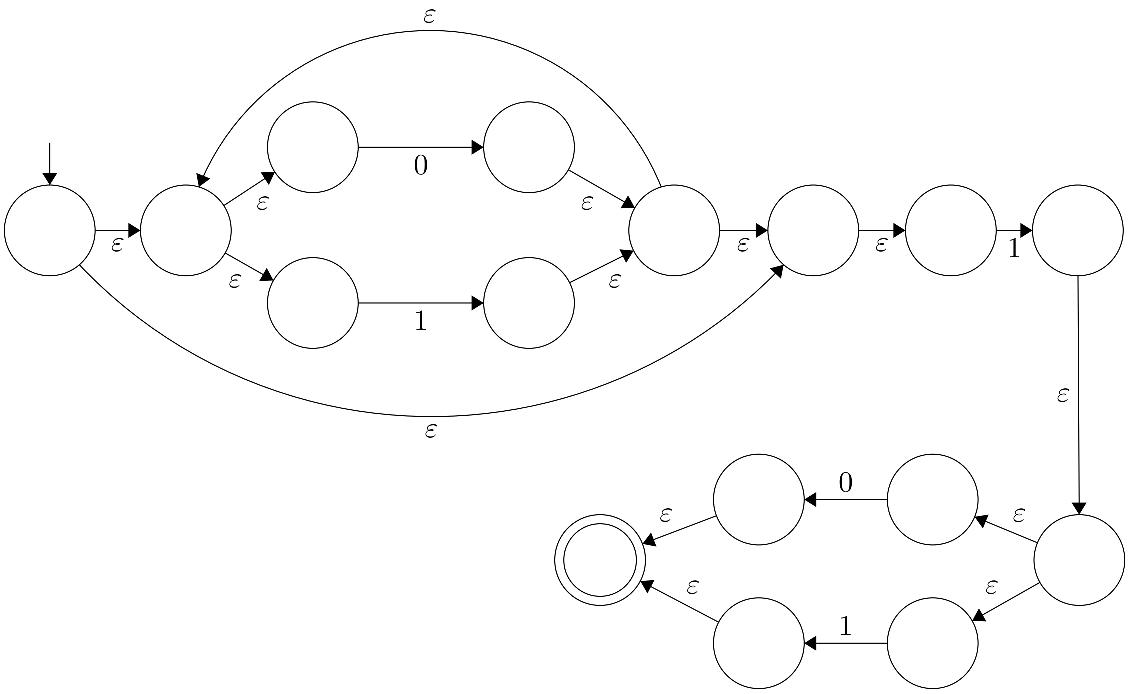
Beispiel. Beispiel: ε-NEAA für $\alpha = (0+1)^*1(0+1)$:
ε-NEAfür $(0+1)$:



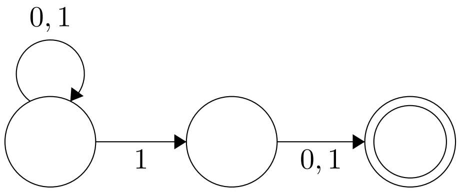
ε-NEAfür $(0+1)^*$:



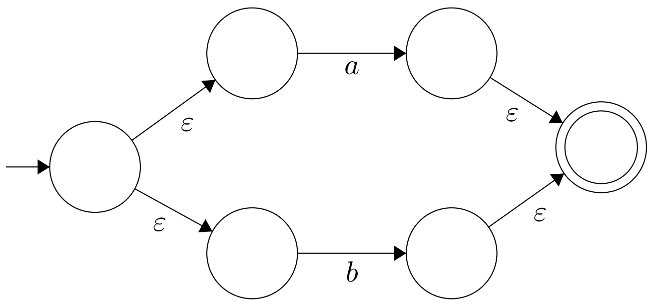
ε-NEAfür $(0+1)^*1(0+1)$:



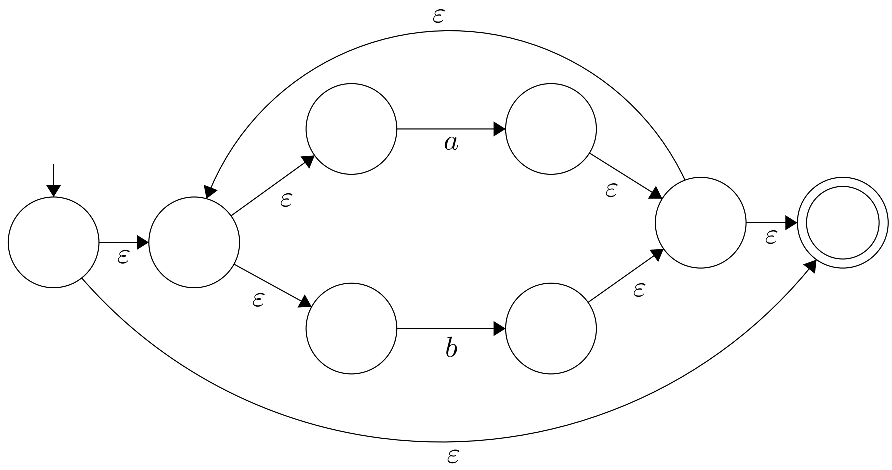
Vereinfachung:



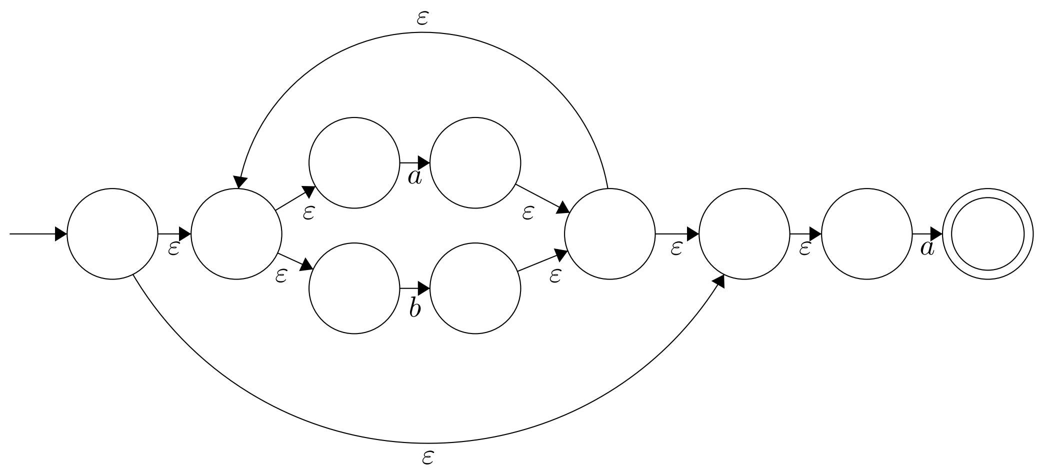
Übung (Aufgabe 1)**.** a) $a + b$:



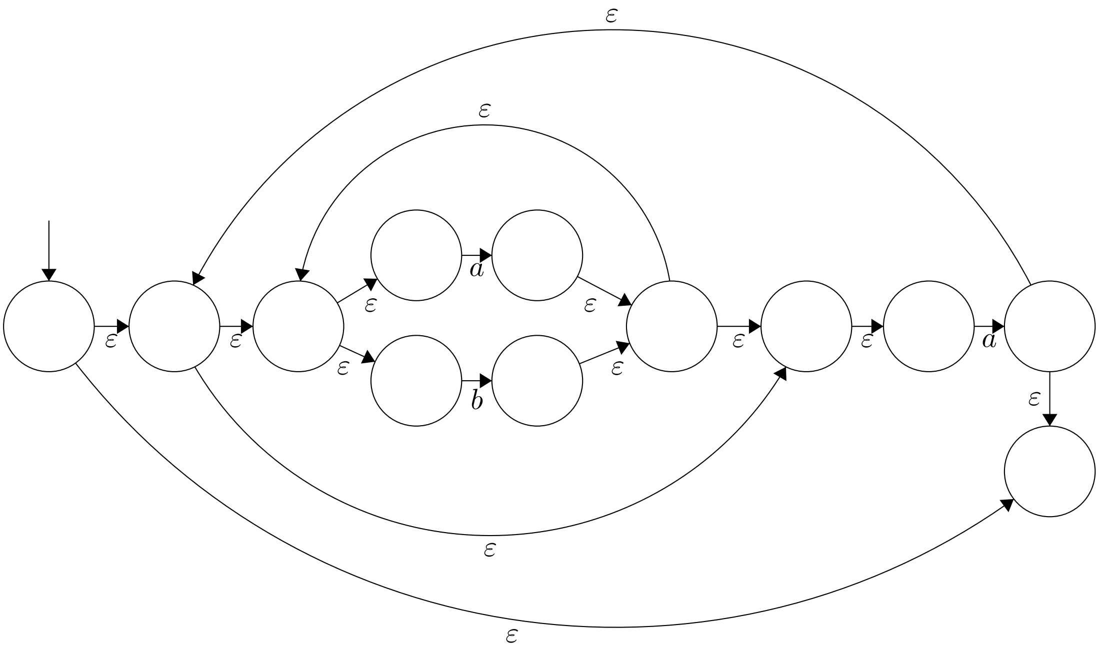
$(a + b)^*$:



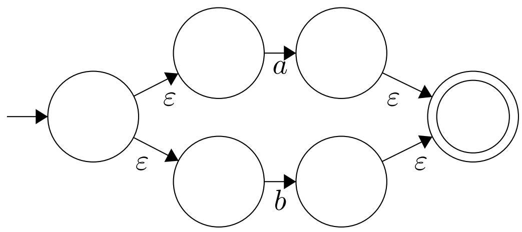
$(a+b)^*a$



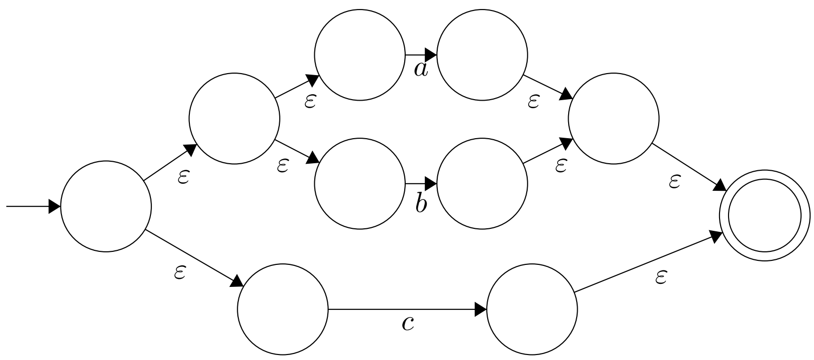
$((a+b)^*a)^*$



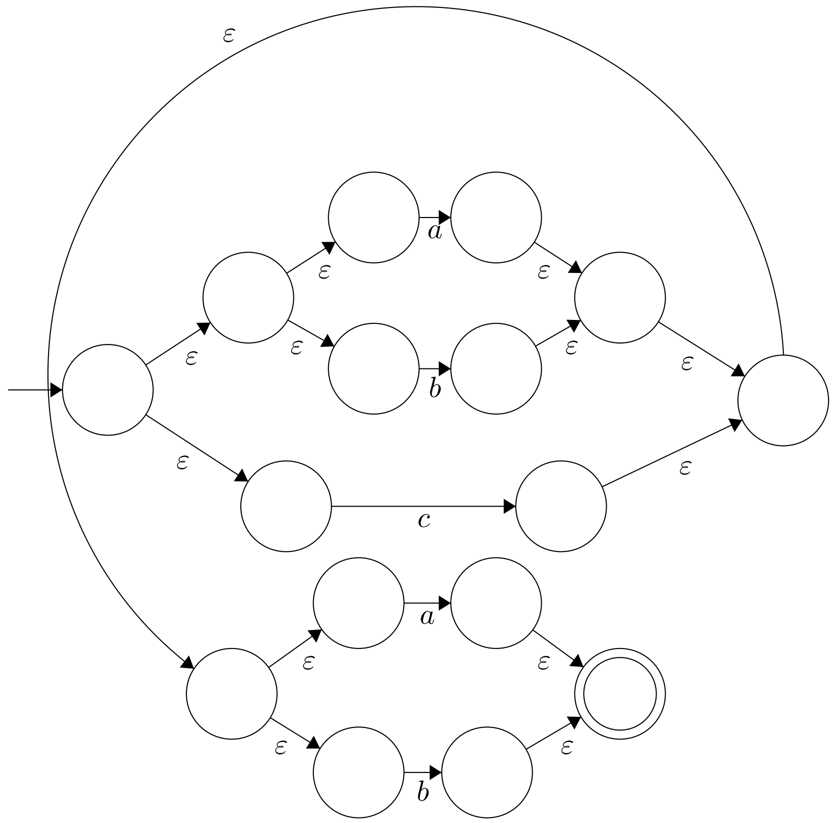
b) $a + b$



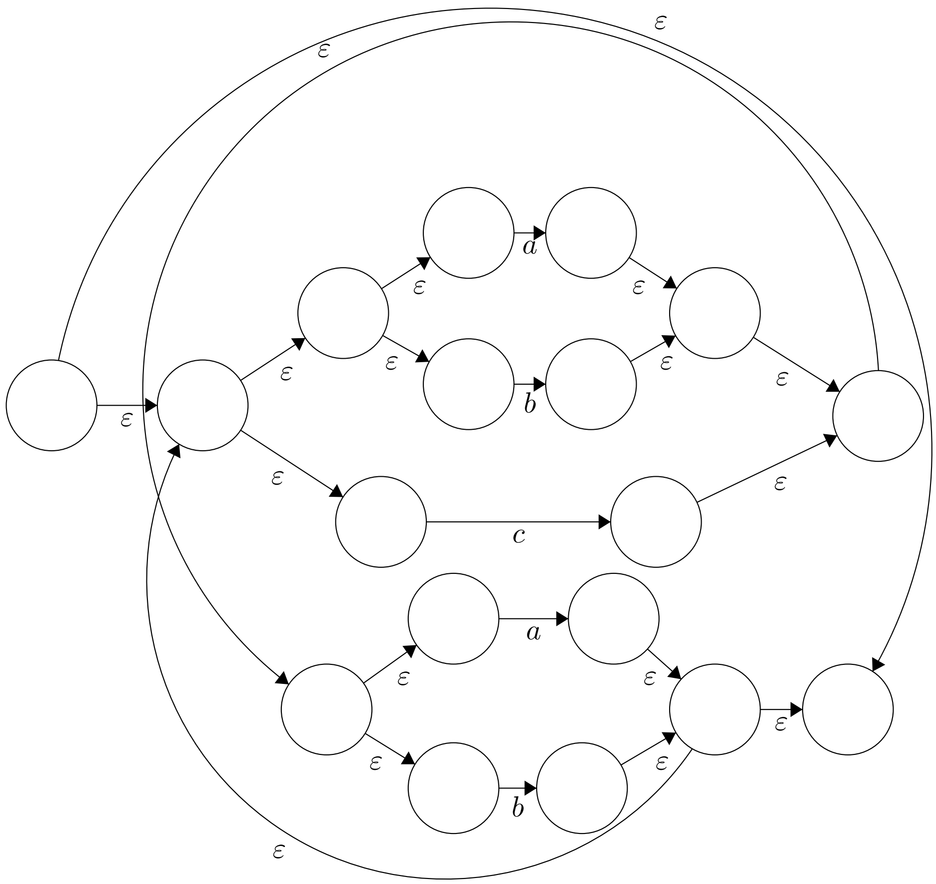
$a + b + c$



$(a + b + c) \cdot (a + b)$



Letzter Schritt:

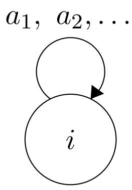


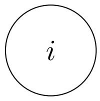
Beweis für 2. Idee: Dynamische Programmierung. Die Teilprobleme hier sind: Für jedes Paar (p, q) von Zuständen finde einen regulären Ausdruck, der alle Wörter beschreibt, die von p nach q führen.

Genauer: Sei $A = (Q, \Sigma, \delta, q_0, F)$ ein DEA, seien die Zustände durchnumeriert, d.h. $Q = \{1, 2, \ldots, n\}$, $q_0 = 1$.

1. Berechne für alle $i, j \in \{1, 2, \ldots n\}$ reguläre Ausdrücke $\alpha_{i,j}^{(0)}$, die die direkten Verbindungen von Zustand i zu Zustand j beschreiben:

   a) $i = j$:





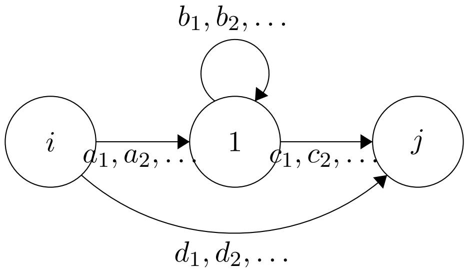
$\alpha_{i,i}^{(0)} = \varepsilon + a_1 + a_2 + \dots$ oder $\alpha_{i,i}^{(0)} = \varepsilon$

b) $i \neq j$

$\alpha_{i,i}^{(0)} = a_1 + a_2 + \dots$ oder $\alpha_{i,j}^{(0)} = \varnothing$

2. Nun werden nacheinander alle anderen Zustände als mögliche Zwischenstation auf dem Weg von i nach j hinzugenommen – zunächst den Zustand 1:

   $\alpha_{i,j}^{(1)}$ beschreibt die Wörter, mit denen man von i nach j kommt und zwischendurch nur den Zustand 1 besucht.



$\alpha_{i,j}^{(1)} = \alpha_{i,j}^{(0)} + \alpha_{i,1}^{(0)}(\alpha_{1,1}^{(0)})^*\alpha_{1,j}^{(0)}$

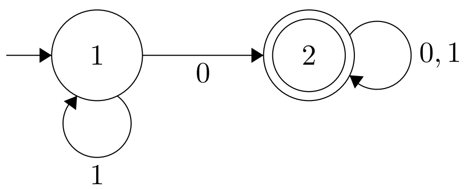
Induktion: Wir nehmen an, dass wir $\alpha_{i,j}^{(k-1)}$ für alle i, j schon berechnet haben. Dann gilt: $\alpha_{i,1}^{(k)} = \alpha_{i,j}^{(k-1)} + \alpha_{i,k}^{(k-1)}(\alpha_{k,k}^{(k-1)})^*\alpha_{k,j}^{(k-1)}$

Daraus folgt der Reguläre Ausdruck für den DEA A mit $q_0 = 1$ und $F = \{f_1, f_2, \dots f_m\}$:

$$\alpha_{1,f_1}^{(n)} + \alpha_{1,f_2}^{(n)} + \dots + \alpha_{1,f_m}^{(n)}$$

□

Beispiel. DEA A:



1. $\alpha_{1,1}^{(0)} = 1 + \varepsilon$

   $\alpha_{1,2}^{(0)} = 1$

   $\alpha_{2,1}^{(0)} = \varnothing$

   $\alpha_{2,2}^{(0)} = \varepsilon + 0 + 1$

2. $\alpha_{1,1}^{(1)} = \alpha_{1,1}^{(0)} + \alpha_{1,1}^{(0)}(\alpha_{1,1}^{(0)})^*\alpha_{1,1}^{(0)} = 1 + \varepsilon + (1+\varepsilon)(\alpha_{1,1}^{(0)})^*\alpha_{1,1}^{(0)} = (1+\varepsilon)^* = 1^*$

   $\alpha_{1,2}^{(1)} = \alpha_{1,2}^{(0)} + \alpha_{1,1}^{(0)}(\alpha_{1,1}^{(0)})^*\alpha_{1,2}^{(0)} = 0 + (1+\varepsilon)(1+\varepsilon)^*0 = 1^*0$

   $\alpha_{2,1}^{(1)} = \alpha_{2,1}^{(0)} + \alpha_{2,1}^{(1)}(\alpha_{1,1}^{(1)})^*\alpha_{1,1}^{(1)} = \varnothing + \varnothing(1+\varepsilon)^*(1+\varepsilon) = \varnothing$

   $\alpha_{2,2}^{(1)} = \alpha_{2,2}^{(0)} + \alpha_{2,1}^{(0)}(\alpha_{1,1}^{(0)})^*\alpha_{1,2}^{(0)} = (\varepsilon + 0 + 1) + \varnothing \cdots = \varepsilon + 0 + 1$

3. $\alpha_{1,1}^{(1)} + \alpha_{1,2}^{(1)}(\alpha_{2,2}^{(1)})^*\alpha_{2,1}^{(1)} = 1^* + 1^*0(\varepsilon+0+1)\varnothing = 1^*$

   $\alpha_{1,2}^{(2)} = \alpha_{1,2}^{(1)} + \alpha_{1,2}^{(1)}(\alpha_{2,2}^{(1)})^*\alpha_{2,2}^{(1)} = 1^*0 + 1^*0(\varepsilon+0+1)^*(\varepsilon+0+1) = 1^*0(0+1)^* = \mathcal{L}(A)$

17 Reguläre Sprachen und ihre Eigenschaften

Definition 25 (Reguläre Sprache)**.** Eine Sprache heisst regulär, wenn sie von einem DEA akzeptiert wird.

$$\mathcal{L}_{\text{reg}} = \mathcal{L}(\text{DEA}) = \{L | L \text{ist regulär}\}$$

Ziel: Charakterisierung von regulären Sprachen.

17.1 Abschlusseigenschaften regulärer Sprachen

Vereinigung Wenn L und M reguläre Sprachen sind, dann ist auch L vereinigt mit M regulär.

Beweis. Wenn L und M regulär sind, dann existieren reguläre Ausdrücke α und β für L und M, d.h. $L(\alpha) = L$ und $L(\beta) = M$.

Daraus folgt, dass $\alpha + \beta$ ebenfalls ein RA ist, daraus folgt $L(\alpha + \beta) = L \cup M$ (gemäss Definition der Vereinigung von RA). □

Verkettung Wenn L und M reguläre Sprachen sind, dann ist auch L verkettet mit M regulär.

Beweis. Analog der Vereinigung. □

Kleene'scher Stern Wenn L eine reguläre Sprache ist, dann ist auch L^* eine reguläre Sprache.

Beweis. Analog der Vereinigung. □

Komplement Wenn L eine reguläre Sprache ist, dann ist auch $\bar{L} = \Sigma^* - L$ regulär.

Beweis. Sei A ein DEA für L. Dann gibt es für jedes Wort w aus Σ^* einen eindeutigen Zustand, in dem der DEA endet, wenn er w liest. Alle Wörter aus $L(A) = L$ führen in Endzustände, alle Wörter aus $\Sigma^* - L = \bar{L}$ in nichtakzeptierende Zustände. Vertauschen von End- und Nicht-Endzuständen ergibt einen DEA für $\bar{L}$. □