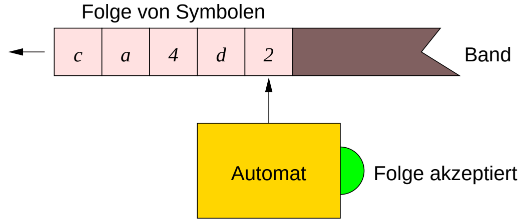
Berechenbarkeit und Komplexität
Endliche Automaten

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
http://www.risc.jku.at

Deterministische endliche Automaten

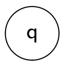


- Automat mit einer endlichen Menge von möglichen Zuständen.
  - Ein Anfangszustand, in dem der Automat die Ausführung beginnt.
  - Eine Menge von Endzuständen.
- Ein Band mit einer endlichen Folge (Wort) von Symbolen.
  - In jedem Schritt liest der Automat das nächste Symbol und bestimmt daraus und aus seinem aktuellen Zustand seinen neuen Zustand.
- Ist das Wort zu Ende, beendet der Automat seine Arbeit.
  - Der Automat signalisiert, ob er in einem der Endzustände ist.

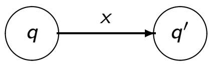
Ist nach Lesen des Worts der Automat in einem Endzustand, wurde das Wort vom Automaten *akzeptiert*.

Deterministische endliche Automaten

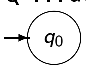
- (Deterministischer) endlicher Automat $M = (Q, \Sigma, \delta, q_0, F)$:
  - Q ... eine endliche Menge von Zuständen.

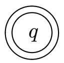
  - Σ ... eine endliche Menge von Symbolen (das Eingabealphabet).
  - $\delta : Q \times \Sigma \to Q$ ... die Überführungsfunktion.
    - $\delta(q, x) = q'$ ... der Automat liest im Zustand q das Symbol x und geht in den Zustand q' über.

  - $q_0 \in Q$ ... der Startzustand.

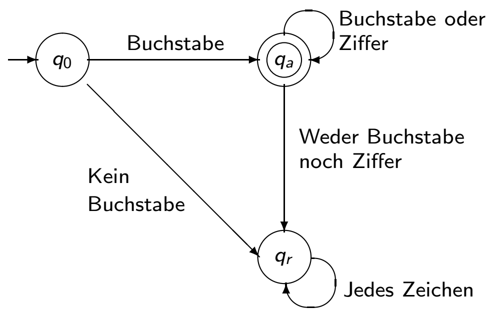
    

  - $F \subseteq Q$ ... die Menge der Endzustände.

Beispiel: Erkennen von Identifiern

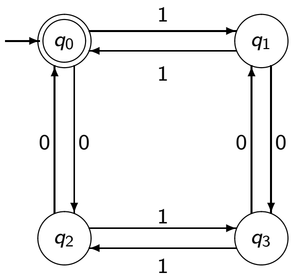
Wort aus Buchstaben und Ziffern, das mit einem Buchstaben beginnt.



- q_0 ... Anfangszustand.
- q_a ... akzeptierender Zustand (accept).
- q_r ... nicht akzeptierender Zustand (reject).

Beispiel: Zählen von Binärziffern

Wort aus 0en und 1en mit jeweils gerader Anzahl von 0en und 1en.



- q_0 ... gerade Anzahl 0en, gerade Anzahl 1en.
- q_1 ... gerade Anzahl 0en, ungerade Anzahl 1en.
- q_2 ... ungerade Anzahl 0en, gerade Anzahl 1en.
- q_3 ... ungerade Anzahl 0en, ungerade Anzahl 1en.

Die Sprache eines Automaten

- Anwendung von δ auf ein Wort:

  $\delta : Q \times \Sigma^* \to Q$
  $\delta(q, \epsilon) := q$
  $\delta(q, x\alpha) := \delta(\delta(q, x), \alpha)$
  - Σ^* ... die Menge der endlichen Wörter über Σ.
  - $\epsilon \in \Sigma^*$ ... das leere Wort
  - $x \in \Sigma$ ... ein Symbol, $\alpha \in \Sigma^*$ ... ein Wort.
  - $\delta(q, x_0x_1x_2 \ldots x_k) = \delta(\delta(\delta(\delta(\delta(q, x_0), x_1), x_2), \ldots), x_k)$
- x wird von M akzeptiert $:\Leftrightarrow \delta(q_0, x) \in F$.
  - Die Anwendung von δ auf den Anfangszustand und das Wort $x \in \Sigma^*$ führt zu einem Endzustand.
- Die von M akzeptierte Sprache $L(M) \subseteq \Sigma^*$:

  $L(M) := \{x \in \Sigma^* \mid x \text{ wird von } M \text{ akzeptiert}\}$
  - Die Menge der von M akzeptierten Wörter.

Sprachen über einem Alphabet

Seien Σ ein Alphabet und $L, L_1, L_2 \subseteq \Sigma^*$ (Sprachen über Σ).

- Komplement: $\overline{L} := \Sigma^* \backslash L$.
- Verkettung: $L_1 \circ L_2 := \{xy \mid x \in L_1, y \in L_2\}$
- Kleene Abschluss: $L^* := \bigcup_{i=0}^{\infty} L^i \; (= L^0 \cup L^1 \cup L^2 \cup \ldots)$
- Positiver Abschluss: $L^+ := \bigcup_{i=1}^{\infty} L^i \; (= L^1 \cup L^2 \cup \ldots)$
  - $L^0 := \{\epsilon\}$.
  - $L^i := L \circ L^{i-1}, i \geq 1$.
- Beispiel: $L = \{a, ba\}$.
  - $L^0 = \{\epsilon\}$
  - $L^1 = L \circ L^0 = L \circ \{\epsilon\} = L = \{a, ba\}$
  - $L^2 = L \circ L^1 = \{a, ba\} \circ \{a, ba\} = \{aa, aba, baa, baba\}$
  - $L^3 = L \circ L^2 = \{a, ba\} \circ \{aa, aba, baa, baba\} = \{aaa, aaba, abaa, ababa, baaa, baaba, babaa, bababa\}$

Reguläre Ausdrücke

Sei Σ ein Alphabet.

- Ein regulärer Ausdruck über Σ ist einer der folgenden Ausdrücke:
  - $\emptyset$, ϵ, α
    - für jedes Symbol $\alpha \in \Sigma$.
  - $(r + s)$, $(r \cdot s)$, (r^*)
    - für jeden regulären Ausdruck r und s über Σ.
- Beispiel: für $\Sigma = \{a, b, c\}$ sind die folgenden reguläre Ausdrücke:
  - $(a \cdot ((b^*) \cdot c))$
  - $(((a + b)^*) \cdot c)$
  - $((a + (b^*)) \cdot (c^*))$

Reguläre Sprachen

Sei Σ ein Alphabet und r ein regulärer Ausdruck über Σ.

- $L(r) \subseteq \Sigma^*$ ... die durch r bezeichnete reguläre Sprache:
  - $L(\emptyset) := \{\}$, $L(\epsilon) := \{\epsilon\}$, $L(\alpha) := \{\alpha\}$
    - für jedes Symbol $\alpha \in \Sigma$.
  - $L(r + s) := L(r) \cup L(s)$
  - $L(r \cdot s) := L(r) \circ L(s)$
  - $L(r^*) := L(r)^*$
  - für jeden regulären Ausdruck r und s über Σ.
- Es gilt z.B. $L((r + s) + t) = L(r + (s + t))$.
  - Wir können $r + s + t$ statt $((r + s) + t)$ schreiben.
  - Wir können auch $r \cdot s \cdot t$ statt $((r \cdot s) \cdot t)$ schreiben.
- Beispiel: sei $\Sigma = \{a, \ldots, z, 0, \ldots, 9\}$.
  - Sei $r := (a + \ldots + z) \cdot (a + \ldots + z + 0 + \ldots + 9)*$.
  - Dann ist $L(r)$ die Sprache der Identifier.

Das Linux `grep` Kommando

```
NAME
       grep, egrep, fgrep, rgrep - print lines matching a pattern

SYNOPSIS
       grep [options] PATTERN [FILE...]
...
REGULAR EXPRESSIONS
       A  regular  expression  is  a  pattern that describes a set of strings.
       ...
       The  fundamental building blocks are the regular expressions that match
       a single character.  Most characters, including all letters and digits,
       are  regular expressions that match themselves.
       ...
       A regular expression may be followed by the repetition operator *;
       the preceding item will be matched zero or more times.
       ...
       Two regular expressions may  be  concatenated;  the  resulting  regular
       expression  matches  any  string formed by concatenating two substrings
       that respectively match the concatenated subexpressions.
       ...
       Two regular expressions may be joined by  the  infix  operator  |;  the
       resulting  expression matches any string matching either subexpression.
```


Reguläre Ausdrücke und Automaten

Sei Σ ein Alphabet, $L \subseteq \Sigma^*$ eine Sprache über Σ.

- Satz:

  (i) L ist eine reguläre Sprache genau dann wenn
  (ii) L von einem endlichen Automaten akzeptiert wird.
- Beweis von (i) $\Rightarrow$ (ii):
  - Sei t ein regulärer Ausdruck über Σ, der die Sprache L bezeichnet.
  - Wir konstruieren einen endlichen Automaten M, der L akzeptiert.
- Beweis von (ii) $\Rightarrow$ (i):
  - Sei M ein endlicher Automat, der die Sprache L akzeptiert.
  - Wir konstruieren einen regulären Ausdruck r mit $L(r) = L$.

Reguläre Ausdrücke und endl. Automaten erzeugen dieselben Sprachen.

Erzeugung von Lexikalischen Analysatoren

Verschiedene Werkzeuge zur Erzeugung von lexikalischen Analysatoren.

- Eingabe: ein regulärer Ausdruck.

```
IDENT: LETTER (LETTER | DIGIT)* ;
```

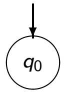
- Ausgabe: ein endlicher Automat (realisiert durch ein Programm):

```
public final void mIDENT(...) throws ... {
  ...
  mLETTER();
  _loop271: do {
    switch ( LA(1)) {
      case 'a':  ... case 'z': { mLETTER(); break; }
      case '0':  ... case '9': { mDIGIT(); break; }
      default: { break _loop271; }
    }
  } while (true);
  ...
}
```

Automat aus regulärem Ausdruck

Beispiel: $t = a \cdot a + b^* \cdot a \cdot b^*$, $\Sigma = \{a, b\}$

- Wir konstruieren t' über Alphabet Σ', in dem jedes Vorkommen eines Symbols $\alpha \in \Sigma$ in t durch ein eindeutig indiziertes α_i ersetzt wird:
  $$t' := a_1 \cdot a_2 + b_1^* \cdot a_3 \cdot b_2^*,\ \Sigma' = \{a_1, a_2, a_3, b_1, b_2\}.$$
- Wir beginnen die Konstruktion von M mit einem Startzustand q_0.



- Jeder weitere Zustand q von M entspricht einer Menge $S \subseteq \Sigma'$.
  - M hat q nach Lesen eines Teils des Eingabeworts erreicht.
  - Die Elemente von S sind die Symbole in Σ', von deren Nachfolgern ausgehend t' den Rest des Eingabeworts beschreiben *könnte*.

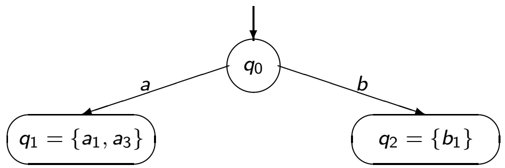
Wir brauchen zusätzliche Informationen über die Sprache $L(t')$.

Automat aus regulärem Ausdruck

$t' := a_1 \cdot a_2 + b_1^* \cdot a_3 \cdot b_2^*$, $\Sigma' = \{a_1, a_2, a_3, b_1, b_2\}$.

- Wir bestimmen für die Sprache $L(t')$:
  - B' …die Menge aller Anfangssymbole.
  - E' …die Menge aller Endsymbole.
  - S' …die Menge aller Paare aufeinanderfolgender Symbole.

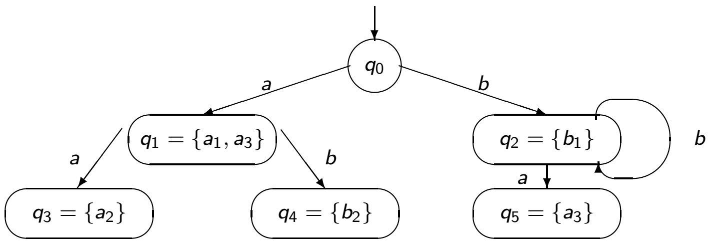
  $B' = \{a_1, b_1, a_3\}$
  $E' = \{a_2, a_3, b_2\}$
  $S' = \{(a_1, a_2), (b_1, b_1), (b_1, a_3), (a_3, b_2), (b_2, b_2)\}$
- Wir bestimmen die Nachfolger von q_0.
  - Für jedes $\alpha \in \Sigma$ die Menge der α_i, die Anfangssymbole sind.



Automat aus regulärem Ausdruck

$t' := a_1 \cdot a_2 + b_1^* \cdot a_3 \cdot b_2^*$, $\Sigma' = \{a_1, a_2, a_3, b_1, b_2\}$.
$S' = \{(a_1, a_2), (b_1, b_1), (b_1, a_3), (a_3, b_2), (b_2, b_2)\}$

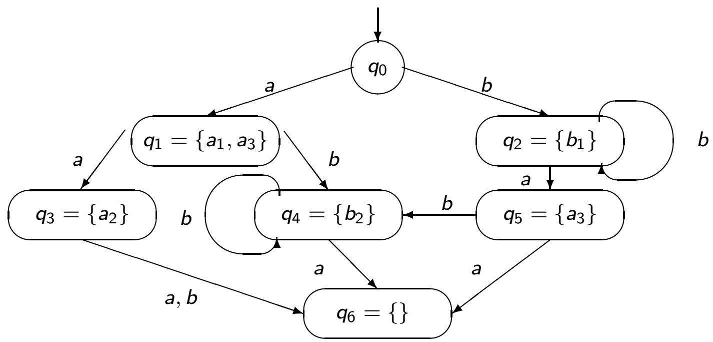
- Wir bestimmen die Nachfolger eines jeden im letzten Schritt erzeugten Zustands q.
  - Für jedes $\alpha \in \Sigma$ die Menge der α_i, die einem Symbol in q nachfolgen.



Automat aus regulärem Ausdruck

$t' := a_1 \cdot a_2 + b_1^* \cdot a_3 \cdot b_2^*$, $\Sigma' = \{a_1, a_2, a_3, b_1, b_2\}$.
$S' = \{(a_1, a_2), (b_1, b_1), (b_1, a_3), (a_3, b_2), (b_2, b_2)\}$

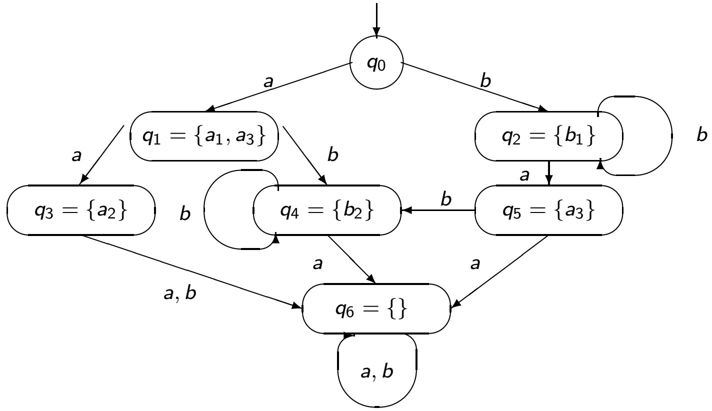
- Wir bestimmen die Nachfolger eines jeden neuen Zustands.



Automat aus regulärem Ausdruck

$t' := a_1 \cdot a_2 + b_1^* \cdot a_3 \cdot b_2^*$, $\Sigma' = \{a_1, a_2, a_3, b_1, b_2\}$.
$S' = \{(a_1, a_2), (b_1, b_1), (b_1, a_3), (a_3, b_2), (b_2, b_2)\}$

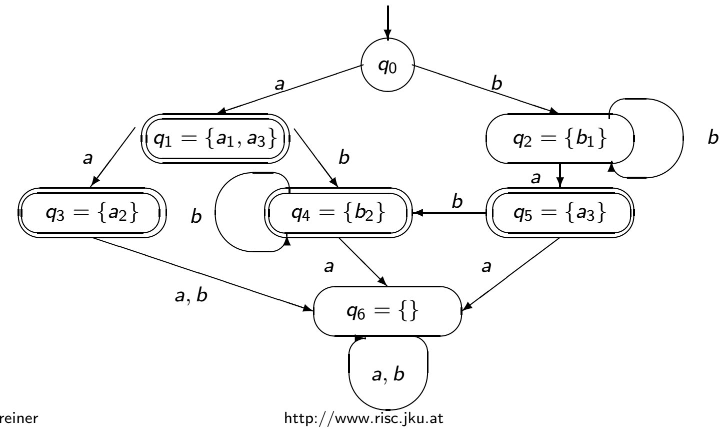
- Wir bestimmen die Nachfolger eines jeden neuen Zustands.



Automat aus regulärem Ausdruck

$t' := a_1 \cdot a_2 + b_1^* \cdot a_3 \cdot b_2^*$, $\Sigma' = \{a_1, a_2, a_3, b_1, b_2\}$.
$E' = \{a_2, a_3, b_2\}$

- Algorithmus terminiert, wenn keine neuen Zustände erzeugt werden.
  - Endzustände sind alle Zustände, die ein Endsymbol enthalten.



Automat aus regulärem Ausdruck

Automat($\downarrow \Sigma, \downarrow L, \downarrow B', \downarrow E', \downarrow S', \uparrow Q, \uparrow \delta, \uparrow q_0, \uparrow F$):
  $q_0 \leftarrow$ "q0" -- beliebiger Anfangszustand
  $Q_0 \leftarrow \{q_0\}$
  $\delta \leftarrow \emptyset$ -- Funktion am Anfang leer
  for $\alpha \in \Sigma$ **do** -- erweitere Funktion um Abbildungen
    $\delta \leftarrow \delta[(q_0, \alpha) \mapsto \{\alpha_i \mid \alpha_i \in B'\}]$
  $Q_1 \leftarrow Q_0 \cup \{\delta(q_0, \alpha) \mid \alpha \in \Sigma\}$
  $k \leftarrow 0$
  while $Q_{k+1} \neq Q_k$ **do**
    for $q \in Q_{k+1} \backslash Q_k$ **do**
      for $\alpha \in \Sigma$ **do** -- erweitere Funktion um Abbildungen
        $\delta \leftarrow \delta[(q, \alpha) \mapsto \{\alpha_i \mid \exists \beta_j \in q : (\beta_j, \alpha_i) \in S'\}]$
    $Q_{k+2} \leftarrow Q_{k+1} \cup \{\delta(q, \alpha) \mid q \in Q_{k+1} \backslash Q_k, \alpha \in \Sigma\}$
    $k \leftarrow k + 1$
  $Q \leftarrow Q_k$
  $F \leftarrow \{q \in Q \mid (q = q_0 \wedge \epsilon \in L) \vee (q \neq q_0 \wedge q \cap E' \neq \emptyset)\}$
end Automat.

Wir verzichten auf die Formalisierung der Berechnung von B', E', S'.

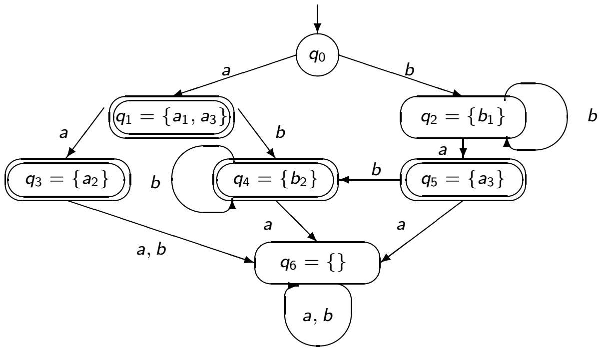
Regulärer Ausdruck aus Automaten

Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein endlicher Automat.

- Wir konstruieren eine regulären Ausdruck t über Σ sodass $L(t)$ die von M akzeptierte Sprache ist.
- Wir definieren $R_{q,p} := \{x \in \Sigma^* | \delta(q, x) = p\}$
  - Die Menge der Wörter, die M von Zustand q in Zustand p überführen.
- Es gilt $L = \bigcup_{p \in F} R_{q_0,p}$.
  - Die von M akzeptierte Sprache besteht aus allen Wörtern, die M vom Anfangszustand in einen der Endzustände überführen.
  - Es genügt also zu zeigen, dass $R_{q,p}$ durch einen regulären Ausdruck $t_{q,p}$ beschrieben ist (für beliebige Zustände q, p).
    - Wir wissen $F = \{p_0, p_1, \ldots, p_k\}$ für irgendwelche $p_0, p_1, \ldots, p_k$.
    - Daher gilt $t := t_{q_0,p_0} + t_{q_0,p_1} + \ldots + t_{q_0,p_k}$.

Es bleibt zu zeigen, dass $R_{q,p}$ eine reguläre Sprache ist.

Beispiel



$$t = t_{q_0,q_1} + t_{q_0,q_3} + t_{q_0,q_4} + t_{q_0,q_5} = a + a \cdot a + (a \cdot b \cdot b^* + b \cdot b^* \cdot a \cdot b \cdot b^*) + b \cdot b^* \cdot a$$

Regulärer Ausdruck aus Automaten

Wir zeigen, dass $R_{q,p}$ eine reguläre Sprache ist.

- Wir wissen $Q = \{q_0, q_1, \ldots, q_r\}$ für irgendwelche $q_0, q_q, \ldots, q_r$.
- Wir definieren $R^j_{q,p}, 1 \leq j \leq r+1$.
  - Die Menge der Wörter, die M von q so nach p überführen, dass dabei nur die Zustände $q_0, \ldots, q_{j-1}$ durchlaufen werden.

$$R^j_{q,p} := \{x \in R_{q,p} \mid \delta(q,y) \in \{q_0, \ldots, q_{j-1}\} \text{ für jedes Prefix } y \text{ von } x\}.$$

- Es gilt $R_{q,p} = R^{r+1}_{q,p}$.
  - Es gibt nur die Zustände $q_0, q_q, \ldots, q_r$.
  - Es genügt also zu zeigen, dass $R^j_{q,p} (1 \leq j \leq r+1)$ durch einen regulären Ausdruck $t^j_{q,p}$ beschrieben ist.

Es bleibt zu zeigen, dass $R^j_{q,p}$ eine reguläre Sprache ist.

Regulärer Ausdruck aus Automaten

Wir zeigen, dass $R^j_{q,p} (1 \leq j \leq r+1)$ eine reguläre Sprache ist.

- Wir definieren $R^0_{q,p}$.
  - Die Menge der Wörter, die M von q nach p überführen und aus maximal einem Symbol bestehen:

$$R^0_{q,p} := \begin{cases} \{\alpha \in \Sigma \mid \delta(q,\alpha) = p\}, & \text{wenn } q \neq p \\ \{\alpha \in \Sigma \mid \delta(q,\alpha) = p\} \cup \{\epsilon\}, & \text{wenn } q = p \end{cases}$$

  - $R^0_{q,p}$ wird durch einen regulären Ausdruck $t^0_{q,p}$ beschrieben.
    - Wir wissen $\{\alpha \in \Sigma \mid \delta(q,\alpha) = p\} = \{\alpha_0, \alpha_1, \ldots, \alpha_k\}$. für irgendwelche $\alpha_0, \alpha_1, \ldots, \alpha_k$.
    - $t^0_{q,p} := \alpha_0 + \alpha_1 + \ldots + \alpha_k$, wenn $q \neq p$.
    - $t^0_{q,p} := \alpha_0 + \alpha_1 + \ldots + \alpha_k + \epsilon$, wenn $q = p$.

Wir benötigen $t^0_{q,p}$ als die Basis für eine induktive Konstruktion des regulären Ausdrucks $t^j_{q,p}$.

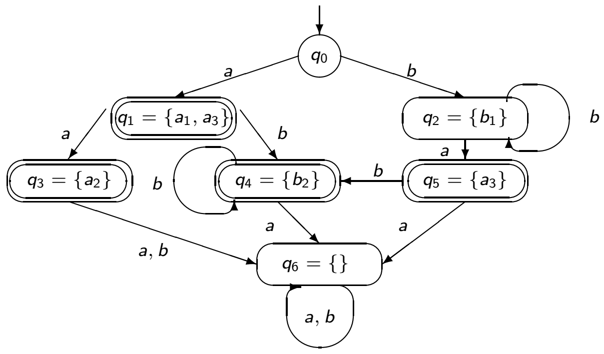
Regulärer Ausdruck aus Automaten

Wir zeigen, dass $R^j_{q,p} (1 \leq j \leq r+1)$ eine reguläre Sprache ist.

- Es gilt $R^{i+1}_{q,p} = R^i_{q,p} \cup R^i_{q,q_i} \circ (R^i_{q_i,q_i})^* \circ R^i_{q_i,p}$.
  - Alle Wege von q nach p, die nur über $q_0, \ldots, q_i$ verlaufen,
    - verlaufen entweder nur über $q_0, \ldots, q_{i-1}$ oder
    - verlaufen von q nach q_i über $q_0, \ldots, q_{i-1}$, haben dann endlich viele Schleifen von q_i zu q_i über $q_0, \ldots, q_{i-1}$ und führen dann von q_i zu p über $q_0, \ldots, q_{i-1}$.
  - $R^{i+1}_{q,p}$ wird durch einen regulären Ausdruck $t^{i+1}_{q,p}$ beschrieben.
    - $t^{i+1}_{q,p} := t^i_{q,p} + t^i_{q,q_i} \cdot (t^i_{q_i,q_i})^* \cdot t^i_{q_i,p}$

Der reguläre Ausdruck, der die von M akzeptierte Sprache bezeichnet, ist also eine "Summe" derartiger Ausdrücken $t^{r+1}_{q_0,p}$ für alle Endzustände p.

Beispiel



$$
\begin{aligned}
t_{q_0,q_1} &= t^7_{q_0,q_1}\\
t^0_{q_0,q_1} &= a\\
t^1_{q_0,q_1} &= a + t^0_{q_0,q_0}\cdot(t^0_{q_0,q_0})^*\cdot t^0_{q_0,q_1} = a+\epsilon\cdot\epsilon^*\cdot a = a+a=a\\
t^2_{q_0,q_1} &= \ldots = t^7_{q_0,q_1} = a
\end{aligned}
$$

Die Charakterisierung regulärer Sprachen

Sei L eine reguläre Sprache.

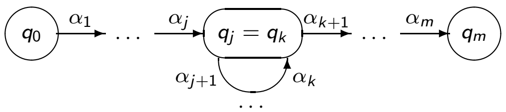
- Pumping Lemma: es gibt eine natürliche Zahl n
  (wobei n kleiner oder gleich der Anzahl der Zustände des kleinsten Automaten ist, der L akzeptiert),
  - sodass jedes Wort z der Sprache L, das aus mindestens n Zeichen besteht, in drei Teile u, v, w zerlegt werden kann, d.h.
    $z = uvw$
    (wobei uv aus maximal n Zeichen und v aus mindestens einem Zeichen besteht),
  - und auch das entsprechende Wort mit beliebig vielen Kopien von v in der Sprache enthalten ist, d.h.
    $uv^iw \in L$
    (für alle $i \geq 0$).

Jedes genügend lange Wort kann zu einem beliebig langen Wort der Sprache "aufgepumpt" werden.

Die Charakterisierung regulärer Sprachen

Beweis des "Pumping Lemma".

- Sei $M = (Q, \Sigma, \delta, q_0, F)$ ein endlicher Automat, der L akzeptiert.
  - Sei n die Anzahl der Zustände in Q.
- Sei z ein Wort der Länge $m \geq n$, d.h.
  $z = \alpha_1 \ldots \alpha_m \in L$
  und sei q_i der Zustand, den M nach dem Lesen des Teilworts $\alpha_1 \ldots \alpha_i$ angenommen hat.
- Da $m \geq n$, müssen zwei Zustände q_j und q_k gleich sein:



- Damit kann z zerlegt werden in $z = uvw$ mit
  - $u = \alpha_1 \ldots \alpha_j$
  - $v = \alpha_{j+1} \ldots \alpha_k$
  - $w = \alpha_{k+1} \ldots \alpha_m$

Beispiel

Das "Pumping Lemma" kann zeigen, dass eine Sprache nicht regulär ist.

- Sei $L = \{a^m b^m \mid m \in \mathbb{N}\}$.
  - $L = \{\epsilon, ab, aabb, aaabbb, \ldots\}$
- Angenommen L wäre regulär.
  - Sei n die durch das Pumping Lemma gegebene Konstante.
- Dann ist $a^n b^n$ ein Wort der Länge $2n \geq n$ in L.
  - Das Wort lässt sich also zerlegen in $a^n b^n = uvw$.
  - uv enthält maximal n zeichen, v mindestens ein Zeichen.
  - $uvw = (a^{n_1})(a^{n_2})(a^{n_3}b^n)$, $n_1 + n_2 + n_3 = n, n_2 \geq 1$
- Dann ist $uv^2w = (a^{n_1})(a^{n_2})^2(a^{n_3}b^n)$ nicht in L.
  - $n_1 + 2n_2 + n_3 = n + n_2 \neq n$.
- Also ist L nicht regulär.

Beispiel

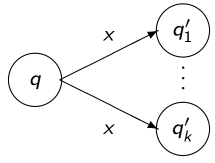
Das "Pumping Lemma" kann zeigen, dass eine Sprache nicht regulär ist.

- Sei $L = \{0^{i^2} \mid i \in \mathbb{N}\}$.
  - $L = \{\epsilon, 0, 0000, 000000000, \ldots\}$
- Angenommen L wäre regulär.
  - Sei n die durch das Pumping Lemma gegebene Konstante.
- Dann ist 0^{n^2} ein Wort der Länge $n^2 \geq n$ in L.
  - Das Wort lässt sich also zerlegen in $0^{n^2} = uvw$.
  - uv enthält maximal n zeichen, v mindestens ein Zeichen.
- Dann ist das Wort uv^2w nicht in L.
  - $1 \leq |v| \leq n$ ($|v| \ldots$ Länge von v)
  - $n^2 = |uvw| < |uvw| + 1 \leq |uvw| + |v| = |uv^2w| \leq n^2 + n < (n+1)^2$.
  - Die Länge von uv^2w ist also keine Quadratzahl.
- Also ist L nicht regulär.

Nichtdeterministische endliche Automaten

Ein Automat kann bei gegebenem Zustand und Eingabesymbol mehr als einen (oder auch keinen) möglichen Nachfolgezustand haben.

- Nichtdeterministischer endlicher Automat $M = (Q, \Sigma, \delta, S, F)$:
  - $Q \ldots$ eine endliche Menge von Zuständen.
  - $\Sigma \ldots$ eine endliche Menge von Symbolen (das Eingabealphabet).
  - $\delta : Q \times \Sigma \to \mathcal{P}(Q) \ldots$ die Überführungsfunktion.
    - $\mathcal{P}(Q) \ldots$ die Menge aller Teilmengen (die *Potenzmenge*) von Q.
    - $\delta(q, x) = \{q'_1, \ldots, q'_k\} \ldots$ der Automat liest im Zustand q das Symbol x und geht in *einen* der Zustände $q'_1, \ldots, q'_k$ über.

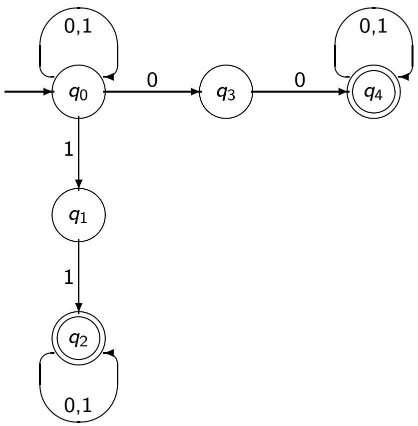


  - $S \subseteq Q \ldots$ die Menge der Startzustände.
  - $F \subseteq Q \ldots$ die Menge der Endzustände.

Beispiel

$M = (Q, \Sigma, \delta, S, F)$
$Q = \{q_0, q_1, q_2, q_3, q_4\}, \Sigma = \{0, 1\}, S = \{q_0\}, F = \{q_2, q_4\}$

δ	0	1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	$\emptyset$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\emptyset$
q_4	$\{q_4\}$	$\{q_4\}$

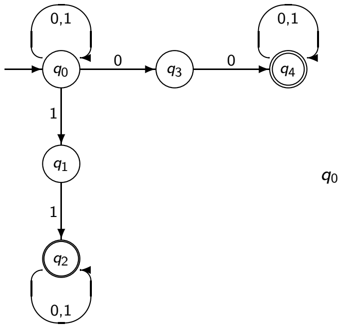


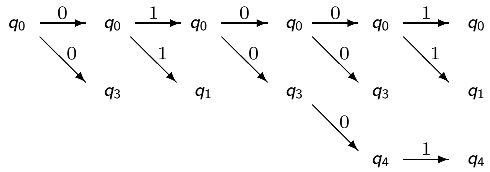
Die Sprache eines nichtdeterm. Automaten

- Anwendung von δ auf ein Wort:
  $\delta : Q \times \Sigma^* \to \mathcal{P}(Q)$
  $\delta(q, \epsilon) := \{q\}$
  $\delta(q, x\alpha) := \{p \mid \exists r \in \delta(q, x) : p \in \delta(r, \alpha)\}$
  $x \in \Sigma \ldots$ ein Symbol, $\alpha \in \Sigma^* \ldots$ ein Wort.
  - Die Menge der möglichen Zustände, die M von q ausgehend nach Lesen des Wortes einnehmen kann.
- Anwendung von δ auf eine Menge von Zuständen:
  $\delta : \mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$
  $\delta(P, x) = \bigcup_{p \in P} \delta(p, x)$
  - Die Menge der möglichen Zustände, die M von einem der Zustände in P ausgehend nach Lesen des Wortes x einnehmen kann.
- x wird von M akzeptiert $:\Leftrightarrow \delta(S, x) \cap F \neq \emptyset$.
  - Die Anwendung von δ auf das Wort $x \in \Sigma^*$ kann von einem der Anfangszustände zu einem der Endzustände führen.

Beispiel

Der Berechnungsbaum des Automaten für Eingabe 01001.





Der Automat akzeptiert alle Wörter, die 00 oder 11 enthalten.

Vergleich mit deterministischen Automaten

- Offensichtlich: jeder deterministische Automat ist der Spezialfall eines nichtdeterministischen Automaten.
  - Jede Sprache, die von einem deterministischen Automaten akzeptiert wird, kann auch von einem nichtdeterministischen akzeptiert werden.
  - Aber gilt auch das umgekehrte Prinzip?
- Satz: Für jeden nichtdeterministischen endlichen Automaten M gibt es einen deterministischen endlichen Automaten M' sodass

  $$L(M) = L(M')$$

  - Der deterministische Automat akzeptiert die gleiche Sprache wie der nichtdeterministische.

Deterministische Automaten und nichtdeterminisische Automaten akzeptieren die gleichen Sprachen, sind also gleichmächtig.

Konstruktion eines determin. Automaten

Sei $M = (Q, \Sigma, \delta, S, F)$ ein nichtdeterministischer endlicher Automat.

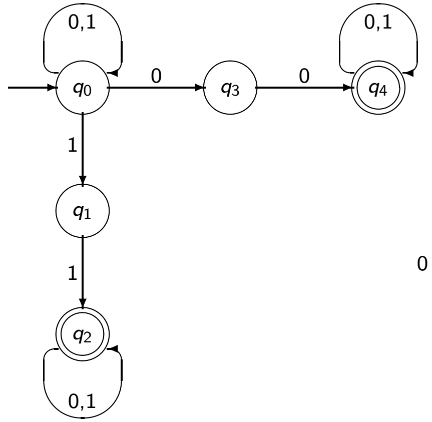
- Wir konstruieren einen deterministischen endlichen Automaten M'
  - Zustände von M' sind Mengen von Zuständen von M.
- $M' = (Q', \Sigma, \delta', q'_0, F')$.
  - $Q' = \mathcal{P}(Q)$
  - $\delta'(q', \alpha) = \bigcup_{p \in q'} \delta(p, \alpha) \; (= \delta(q', \alpha))$
  - $q'_0 = S$
  - $F' = \{q' \in Q' \mid q' \cap F \neq \emptyset\}$
- Es gilt $L(M') = L(M)$, da für jedes Wort $x = \alpha_1 \ldots \alpha_n \in \Sigma^*$ gilt:
  - $x \in L(M) \Leftrightarrow \delta(S, x) \cap F \neq \emptyset$
  - Für gewisse Zustände $q_0, \ldots, q_n$ in Q gilt daher

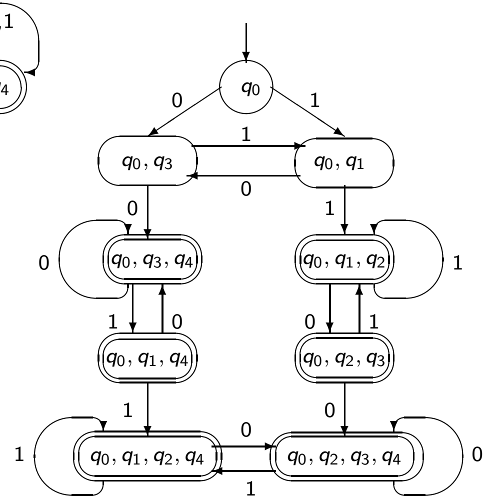
    $\delta(q_0, \alpha_1) = q_1, \ldots, \delta(q_{n-1}, \alpha_n) = q_n, q_n \in F$.
  - Für gewisse Zustände $q'_0, \ldots, q'_n$ in Q' (Teilmengen von Q) gilt daher

    $q_0 \in q'_0, \ldots, q_n \in q'_n$

    $\delta'(q'_0, \alpha_1) = q'_1, \ldots, \delta'(q'_{n-1}, \alpha_n) = q'_n, q'_n \cap F \neq \emptyset$.
  - $\delta'(q'_0, x) \in F' \Leftrightarrow x \in L(M')$

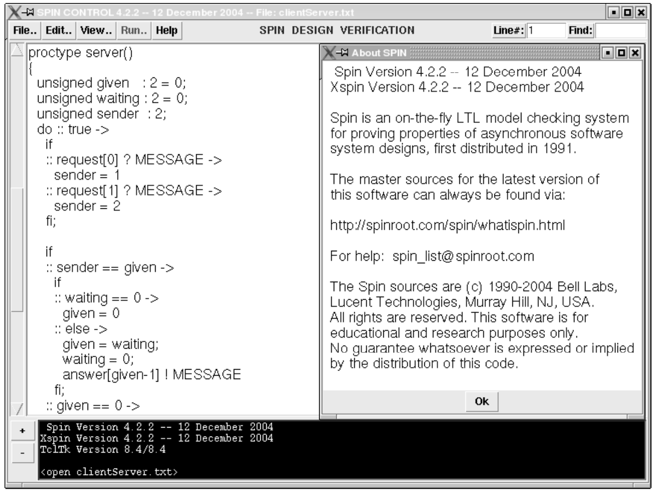
Beispiel





Anwendung nichtdetermin. Automaten

Die Modellierung und Verifikation nebenläufiger Systeme.



Anwendung nichtdetermin. Automaten

Systemkomponente als nichtdeterministischer Automat.