TECHNISCHE UNIVERSITÄT DRESDEN

Fakultät Informatik, Institut für Künstliche Intelligenz, Professur für Computational Logic

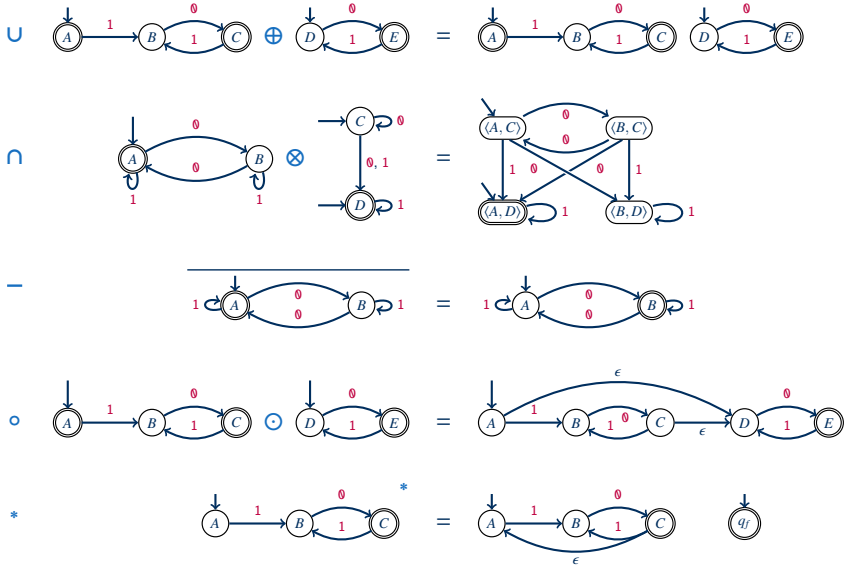
FORMALE SYSTEME

6. Vorlesung: Reguläre Ausdrücke

Hannes Straß

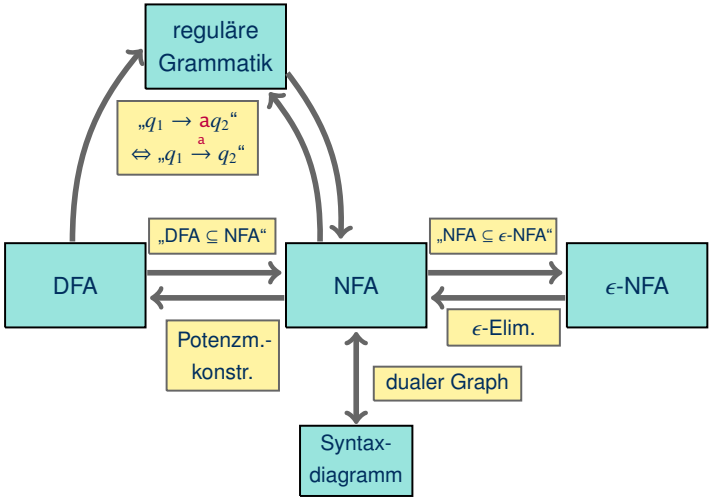
TU Dresden, 28. Oktober 2021

Wiederholung: Operationen auf Automaten



NFAs mit Wortübergängen

Darstellungen von Typ-3-Sprachen



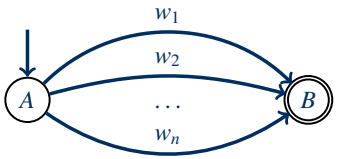
Reguläre Ausdrücke

Endliche Sprachen

Eine einfache Beobachtung:

Satz: Jede endliche Sprache ist regulär.

Beweis: Man kann eine beliebige endliche Sprache $\{w_1, \ldots, w_n\}$ durch einen NFA mit Wortübergängen erkennen:



Wie in der letzten Vorlesung gezeigt, kann dieser in einen NFA umgeformt werden.
Jeder NFA akzeptiert eine reguläre Sprache. □

Sprachen konstruieren?

Wir haben gesehen:

- Jede endliche Sprache ist regulär.
- Durch Anwendung von $\cap$, $\cup$, $\overline{}$, $\circ$ und * entstehen aus regulären Sprachen immer wieder reguläre Sprachen.

Eine natürliche Frage ist also:

Welche regulären Sprachen kann man durch Anwendung von $\cap$, $\cup$, $\overline{}$, $\circ$ und * aus endlichen Sprachen konstruieren?

Alle!

Die kleinste $(\cup, \circ, ^*)$-abgeschlossene Klasse

Überraschender Weise sind $\cap$ und $\overline{}$ nicht einmal nötig!

Satz: Alle regulären Sprachen können durch Anwendung von $\cup$, $\circ$ und * aus endlichen Sprachen konstruiert werden.

Mit $\circ$ und $\cup$ kann man jede endliche Sprache über einem Alphabet Σ leicht aus den Sprachen $\emptyset$, $\{\epsilon\}$ und $\{a\}$ (für jedes $a \in \Sigma$) konstruieren.

Mit den bekannten Abschlusseigenschaften erhält man also:

Satz: Die Klasse der regulären Sprachen ist die kleinste Klasse von Sprachen mit den folgenden Eigenschaften:

- Sie enthält die Sprachen $\emptyset$, $\{\epsilon\}$ und $\{a\}$ für alle $a \in \Sigma$.
- Sie ist abgeschlossen unter den Operatoren $\cup$, $\circ$ und *.

Beweisplan:

- Definiere diese Klasse syntaktisch: reguläre Ausdrücke
- Zeige, dass diese genau die regulären Sprachen darstellen

Reguläre Ausdrücke

Das motiviert die Einführung einer eigenen Syntax:

Die Menge regulärer Ausdrücke über einem Alphabet Σ ist induktiv[a] wie folgt definiert:

- $\emptyset$ ist ein regulärer Ausdruck;
- ϵ ist ein regulärer Ausdruck;
- für jedes $\mathsf{a} \in \Sigma$ ist a ein regulärer Ausdruck;
- wenn α und β reguläre Ausdrücke sind, dann sind auch
  - $(\alpha\beta)$ (Konkatenation)
  - $(\alpha \mid \beta)$ (Alternative)
  - $(\alpha)^*$ (Kleene-Stern)

  reguläre Ausdrücke.

[a] D.h. die Menge der regulären Ausdrücke ist die kleinste Menge, welche die Bedingungen erfüllt.

Anmerkung: Manchmal werden $(\alpha + \beta)$ statt $(\alpha \mid \beta)$ und $(\alpha \circ \beta)$ oder $(\alpha \cdot \beta)$ statt $(\alpha\beta)$ verwendet.

Beispiele regulärer Ausdrücke sind $(\mathsf{a}(\mathsf{b})^*)$ oder auch $(((\epsilon \mid \epsilon))^*)^*$.

Reguläre Ausdrücke als formale Sprache

Man kann die Menge der regulären Ausdrücke über dem Alphabet $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ auch als kontextfreie Grammatik über dem Alphabet $\Sigma \cup \{\emptyset, \epsilon, (,), |, ^*\}$ beschreiben:

$$\mathsf{S} \to \emptyset \mid \epsilon \mid \mathsf{A} \mid (\mathsf{SS}) \mid (\mathsf{S}|\mathsf{S}) \mid (\mathsf{S})^*$$
$$\mathsf{A} \to \sigma_1 \mid \ldots \mid \sigma_n$$

Solche Notationen werden in der Praxis oft weiter vereinfacht:

- Endliche Mengen als Nichtterminale:

$$\mathsf{S} \to \emptyset \mid \epsilon \mid \Sigma \mid (\mathsf{SS}) \mid (\mathsf{S}|\mathsf{S}) \mid (\mathsf{S})^*$$

- Mehrere Nichtterminale als Hinweis auf möglicherweise unterschiedliche Ausdrücke:

$$\alpha \to \emptyset \mid \epsilon \mid \Sigma \mid (\alpha_1\alpha_2) \mid (\alpha_1 \mid \alpha_2) \mid (\alpha)^*$$

Semantik regulärer Ausdrücke

Reguläre Ausdrücke beschreiben die erwarteten Sprachen:

Die Sprache eines regulären Ausdrucks α wird mit $\mathbf{L}(\alpha)$ bezeichnet und induktiv an Hand der Form von α definiert:

- $\mathbf{L}(\emptyset) = \emptyset$
- $\mathbf{L}(\epsilon) = \{\epsilon\}$
- $\mathbf{L}(\mathsf{a}) = \{\mathsf{a}\}$ für jedes $\mathsf{a} \in \Sigma$
- $\mathbf{L}((\alpha\beta)) = \mathbf{L}(\alpha) \circ \mathbf{L}(\beta)$
- $\mathbf{L}((\alpha \mid \beta)) = \mathbf{L}(\alpha) \cup \mathbf{L}(\beta)$
- $\mathbf{L}((\alpha)^*) = \mathbf{L}(\alpha)^*$

Beispiel: $\mathbf{L}(((\mathsf{ab}))^*) = (\mathbf{L}((\mathsf{ab})))^* = (\mathbf{L}(\mathsf{a}) \circ \mathbf{L}(\mathsf{b}))^* = (\{\mathsf{a}\} \circ \{\mathsf{b}\})^* = \{\mathsf{ab}\}^*$

Quiz: Sprache eines regulären Ausdrucks

Die Sprache $\mathbf{L}(\alpha)$ eines regulären Ausdrucks α ist induktiv definiert:

$$\begin{aligned}
\mathbf{L}(\emptyset) &= \emptyset \\
\mathbf{L}(\epsilon) &= \{\epsilon\} \\
\mathbf{L}(\mathsf{a}) &= \{\mathsf{a}\} \text{ für jedes } \mathsf{a} \in \Sigma \\
\mathbf{L}((\alpha\beta)) &= \mathbf{L}(\alpha) \circ \mathbf{L}(\beta) \\
\mathbf{L}((\alpha \mid \beta)) &= \mathbf{L}(\alpha) \cup \mathbf{L}(\beta) \\
\mathbf{L}((\alpha)^*) &= \mathbf{L}(\alpha)^*
\end{aligned}$$

Quiz: Wir betrachten den regulären Ausdruck α über $\Sigma = \{\mathsf{0}, \mathsf{1}\}$:

...

Äquivalenz regulärer Ausdrücke

Zwei reguläre Ausdrücke α und β sind genau dann äquivalent, in Symbolen $\alpha \equiv \beta$, wenn $\mathbf{L}(\alpha) = \mathbf{L}(\beta)$.

Typische Rechenregeln der Sprachoperationen gelten analog:

$$\alpha \mid \beta \equiv \beta \mid \alpha \qquad \epsilon\alpha \equiv \alpha\epsilon \equiv \alpha$$
$$\alpha(\beta \mid \gamma) \equiv \alpha\beta \mid \alpha\gamma \qquad \emptyset\alpha \equiv \alpha\emptyset \equiv \emptyset$$
$$(\beta \mid \gamma)\alpha \equiv \beta\alpha \mid \gamma\alpha \qquad \emptyset \mid \alpha \equiv \alpha \mid \emptyset \equiv \alpha$$
$$\epsilon^* \equiv \epsilon \qquad \emptyset^* \equiv \epsilon$$

Vereinfachte Klammerung

Reguläre Ausdrücke können durch Klammerungsregeln vereinfacht werden:

* bindet stärker als Konkatenation bindet stärker als $|$.

Beispiel: $\texttt{ab}^* \mid \texttt{bc}$ ist kurz für $((\texttt{a}(\texttt{b})^*) \mid (\texttt{bc}))$.

Konkatenation und Alternative sind assoziativ:

$$\mathbf{L}((\alpha(\beta\gamma))) = \mathbf{L}(((\alpha\beta)\gamma)) \qquad \mathbf{L}((\alpha \mid (\beta \mid \gamma))) = \mathbf{L}(((\alpha \mid \beta) \mid \gamma))$$

Daher ist es unproblematisch, dass die Klammerregeln die Reihenfolge nicht spezifizieren.

Beispiel: $\texttt{101010}$ könnte für genau 42 verschiedene reguläre Ausdrücke stehen, unter anderem für $(((((\texttt{10})\texttt{1})\texttt{0})\texttt{1})\texttt{0})$, $((\texttt{10})((\texttt{10})(\texttt{10})))$ und $(\texttt{1}((\texttt{0}((\texttt{10})\texttt{1}))\texttt{0}))$.

Kurzschreibweisen für reguläre Ausdrücke

Man kann eine Reihe von Kurzschreibweisen definieren:

- α^+ ist kurz für $\alpha(\alpha)^*$
- $\alpha?$ ist kurz für $(\alpha \mid \epsilon)$
- $\alpha\{n, m\}$ mit $0 \leq n \leq m$ ist kurz für $(\underbrace{\alpha \ldots \alpha}_{n\text{-mal}} \mid \ldots \mid \underbrace{\alpha \ldots \alpha}_{m\text{-mal}})$

Implementierungen regulärer Ausdrücke bieten auch Kurzformen für Alternativen einzelner Symbole („Character Classes“):

- `.`: beliebiges Symbol, d.h. $\sigma_1 \mid \ldots \mid \sigma_n$ falls $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$
- `[`$\theta_1 \cdots \theta_m$`]`: beliebiges Symbol aus einer Liste, d.h. $\theta_1 \mid \ldots \mid \theta_m$
- `[^`$\theta_1 \cdots \theta_m$`]`: beliebiges Symbol nicht aus einer Liste, d.h. $\sigma_1 \mid \ldots \mid \sigma_\ell$ mit $\{\sigma_1, \ldots, \sigma_\ell\} = \Sigma \setminus \{\theta_1, \ldots, \theta_m\}$
- Weitere Kurzformen für praktisch wichtige Fälle, z.B. `\s` oder `[:blank:]` für Leerzeichen, `\d` oder `[:digit:]` für Ziffern

Theoretiker:innen meiden Kurzformen (mehr Formen $\leadsto$ mehr Fälle in Beweisen und Definitionen)

Regexps in der Praxis

Reguläre Ausdrücke sind von großer praktischer Bedeutung:

- Mustererkennung in Texten (Pattern Matching)
- Lexer/Tokenizer
- Suche nach Mustern in Datenbanken
- ...

Unterschiede zur reinen Lehre:

- Pattern Matching: (1) spezifiziere eine Sprache (Suchwörter) und (2) finde deren Vorkommen (Matches) in einem längeren Wort (Text) $\leadsto$ Möglichkeiten zur Steuerung des zweiten Teils nötig (z.B. **greedy** vs. **lazy** matching)
- Referenzen: praktische Implementierungen erlauben es meist, Teile eines Matches im Muster wieder zu verwenden $\leadsto$ keine reguläre Sprache mehr
- Escaping: Unterscheidung von Steuerzeichen (Metazeichen) und Alphabetssymbolen ist praktisch aufwändig

Konkreter regulärer Ausdruck zur Erkennung gültiger E-Mail-Adressen in JavaScript:

```
/^(([^<>()\\[\]\\.,;:\s@"]+(\.[^<>()\[\]\\.,;:\s@"]+)*)|(".+"))@
((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/
```

Von Regulären Ausdrücken zu Automaten

Zielstellung

Behauptung: Eine Sprache **L** ist genau dann regulär, wenn es einen regulären Ausdruck α gibt, für den $\mathbf{L}(\alpha) = \mathbf{L}$ gilt.

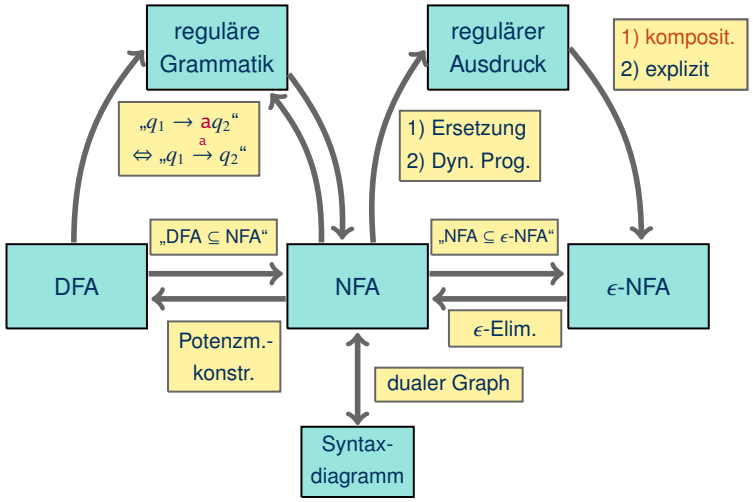
Beweis (Plan):

- Teilbehauptung 1: Für jeden regulären Ausdruck α gibt es einen NFA $\mathcal{M}$, so dass $\mathbf{L}(\alpha) = \mathbf{L}(\mathcal{M})$.

  Zwei mögliche Beweismethoden:
  (1) Kompositionelle Methode
  (2) Explizite Konstruktion
- Teilbehauptung 2: Für jeden NFA $\mathcal{M}$ gibt es einen regulären Ausdruck α, so dass $\mathbf{L}(\alpha) = \mathbf{L}(\mathcal{M})$.

  Zwei mögliche Beweismethoden:
  (1) Ersetzungsmethode
  (2) Dynamische Programmierung

Darstellungen von Typ-3-Sprachen



Rekursive Komposition von ϵ-NFA

Die Struktur regulärer Ausdrücke kann mit Operationen auf Automaten direkt abgebildet werden.

Für einen Ausdruck α definieren wir induktiv den ϵ-NFA $\mathcal{M}(\alpha)$:

Grundfälle:

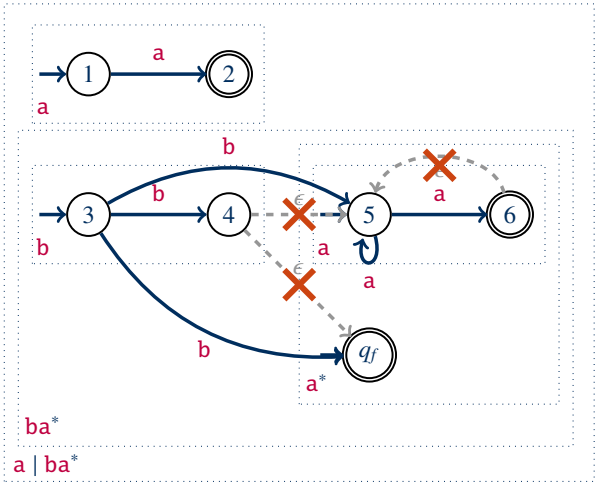
- Wenn $\alpha = \emptyset$, dann $\mathcal{M}(\alpha) =$
- Wenn $\alpha = \epsilon$, dann $\mathcal{M}(\alpha) =$
- Wenn $\alpha = a$, dann $\mathcal{M}(\alpha) =$

Rekursive Fälle: Wir bezeichnen mit $\text{elim}_\epsilon(\mathcal{M})$ den NFA, der aus einem ϵ-NFA $\mathcal{M}$ durch Eliminierung der ϵ-Übergänge entsteht.

- Wenn $\alpha = (\beta\gamma)$, dann $\mathcal{M}(\alpha) = \text{elim}_\epsilon(\mathcal{M}(\beta) \odot \mathcal{M}(\gamma))$.
- Wenn $\alpha = (\beta \mid \gamma)$, dann $\mathcal{M}(\alpha) = \mathcal{M}(\beta) \oplus \mathcal{M}(\gamma)$.
- Wenn $\alpha = (\beta)^*$, dann $\mathcal{M}(\alpha) = \text{elim}_\epsilon(\mathcal{M}(\beta)^*)$.

Beispiel

Regulärer Ausdruck: a | ba*



Korrektheit der Kompositionsmethode

Satz: Für jeden regulären Ausdruck α gilt $\mathbf{L}(\alpha) = \mathbf{L}(\mathcal{M}(\alpha))$.

Beweis: Die Gleichheit folgt aus der Definition von $\mathbf{L}(\alpha)$ und der Korrektheit der Operationen auf Automaten und der ϵ-Eliminierung.

Formal ist der Beweis eine strukturelle Induktion: Wir konstruieren unsere Argumentation entlang der (induktiv definierten) Struktur regulärer Ausdrücke.

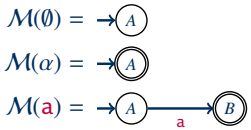
- Induktionsanfang: Für die Grundfälle $\alpha = \emptyset$, $\alpha = \epsilon$ und $\alpha = a$ ist die Behauptung leicht zu sehen.
- Induktionsvoraussetzung (IV): Die Behauptung wurde bereits für β und γ gezeigt, d.h. $\mathbf{L}(\beta) = \mathbf{L}(\mathcal{M}(\beta))$ und $\mathbf{L}(\gamma) = \mathbf{L}(\mathcal{M}(\gamma))$.
- Induktionsschritt: Im Fall $\alpha = (\beta\gamma)$ gilt:

$$\mathbf{L}(\alpha) \overset{\text{Def.}}{=} \mathbf{L}(\beta) \circ \mathbf{L}(\gamma) \overset{\text{IV}}{=} \mathbf{L}(\mathcal{M}(\beta)) \circ \mathbf{L}(\mathcal{M}(\gamma))$$
$$\overset{(1)}{=} \mathbf{L}(\mathcal{M}(\beta) \odot \mathcal{M}(\gamma)) \overset{(2)}{=} \mathbf{L}(\text{elim}_\epsilon(\mathcal{M}(\beta) \odot \mathcal{M}(\gamma))) \overset{\text{Def.}}{=} \mathbf{L}(\mathcal{M}(\alpha)).$$

(1) Korrektheit der Operation $\odot$; (2) Korrektheit der ϵ-Eliminierung

Die Fälle $\alpha = (\beta \mid \gamma)$ und $\alpha = (\beta)^*$ sind analog. □

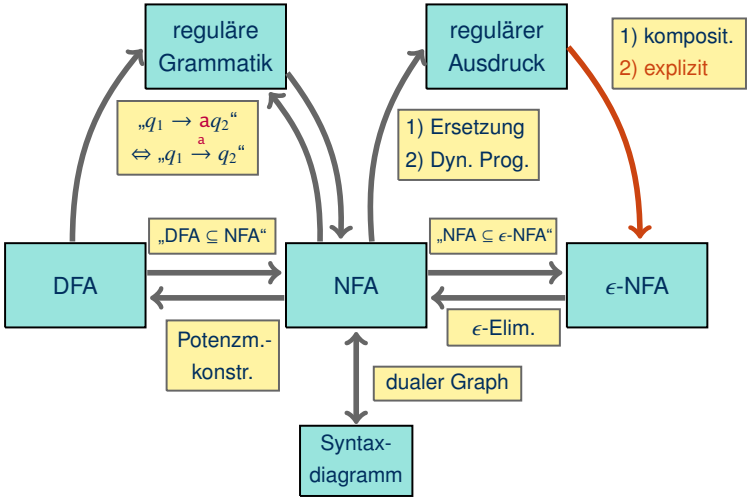
Quiz: Kompositionsmethode



$$\mathcal{M}(\beta\gamma) = \text{elim}_\epsilon(\mathcal{M}(\beta) \odot \mathcal{M}(\gamma))$$
$$\mathcal{M}(\beta \mid \gamma) = \mathcal{M}(\beta) \oplus \mathcal{M}(\gamma)$$
$$\mathcal{M}((\beta)^*) = \text{elim}_\epsilon(\mathcal{M}(\beta)^*)$$

Quiz: Wir betrachten den Automaten $\mathcal{M}(\alpha)$, der mittels Kompositionsmethode aus dem regulären Ausdruck α hervorgegangen ist.
...

Darstellungen von Typ-3-Sprachen



Explizite Konstruktion des NFA

Idee:

- Beginne mit einem „NFA mit RegExp-Übergängen“, bei dem Kanten mit regulären Ausdrücken beschriftet sind.
- Eliminiere diese Übergänge schrittweise, ähnlich wie beim Eliminieren von Wortübergängen.

Es ist einfacher, das für reguläre Ausdrücke zu tun, die kein $\emptyset$ enthalten.

Eliminierung von $\emptyset$

Der folgende einfache Algorithmus erzeugt reguläre Ausdrücke ohne innere Vorkommen von $\emptyset$:

Eingabe: regulärer Ausdruck α
Ausgabe: äquivalenter regulärer Ausdruck β ohne $\emptyset$ (falls $\mathbf{L}(\alpha) \neq \emptyset$, sonst Ausgabe $\emptyset$)

Wende die folgenden Ersetzungsregeln erschöpfend auf Teilausdrücke von α an:

- $(\gamma \mid \emptyset) \mapsto \gamma$ und $(\emptyset \mid \gamma) \mapsto \gamma$
- $(\gamma\emptyset) \mapsto \emptyset$ und $(\emptyset\gamma) \mapsto \emptyset$
- $(\emptyset)^* \mapsto \epsilon$

Gib das Ergebnis dieser Ersetzungen aus.

- Die Korrektheit des Algorithmus folgt aus der Korrektheit der angewendeten Ersetzungsregeln.
- Der Algorithmus terminiert, da in jedem Schritt die Größe (z.B. Anzahl echter Teilausdrücke) des verarbeiteten Ausdrucks verringert wird.

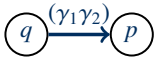
Explizite Konstruktion von NFAs

Für den Ausdruck $\emptyset$ können wir einen NFA direkt angeben (wie vorn); andernfalls gehen wir wie folgt vor:

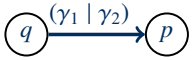
Gegeben: Ein regulärer Ausdruck α ohne $\emptyset$.

Initialisierung: $\mathcal{M}_\alpha =$ 

Solange es in $\mathcal{M}_\alpha$ Übergänge $q \xrightarrow{\beta} p$ gibt, die mit einem Ausdruck $\beta \notin \{\epsilon\} \cup \Sigma$ beschriftet sind, wende eine der folgenden Regeln an:

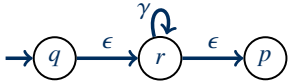
- Ersetze 

durch 

- Ersetze 

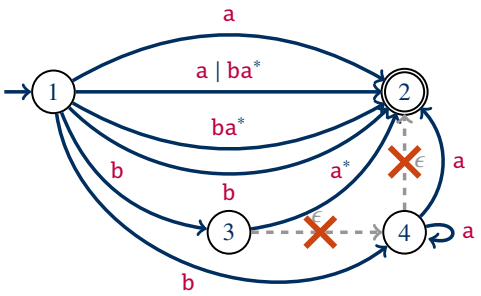
durch 

- Ersetze 

durch 

Beispiel

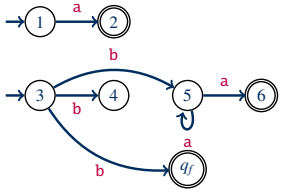
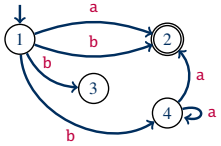
Regulärer Ausdruck: a | ba*



ϵ-Übergänge können wie gewohnt eliminiert werden, um einen NFA zu erhalten

Vergleich NFA-Konstruktionen

Ausdruck: a | ba*

Kompositioneller Ansatz	Expliziter Ansatz
	
Linear viele Zustände[1]	Linear viele Zustände[1]
Meist etwas größer	Meist etwas kleiner
Korrektheitsbeweis aus Abschlusseigenschaften	Korrektheitsbeweis erfordert neue Argumentation
Fast ohne Löschen (außer ϵ-Übergänge)	Übergänge werden oft gelöscht

[1] Bezüglich der Länge des regulären Ausdrucks bzw. bezüglich der Anzahl seiner Operationen.

Zusammenfassung und Ausblick

Reguläre Ausdrücke sind eine praktisch wichtige Methode zur Beschreibung (beliebiger) regulärer Sprachen.

Die kompositionelle Methode wendet Automaten-Operationen an, um aus einem regulären Ausdruck einen NFA zu erzeugen.

Die explizite Methode verwendet Übergänge mit regulären Ausdrücken, die schrittweise expandiert werden.

Offene Fragen:

- Wie kommt man von NFA („zurück“) zu regulärem Ausdruck?
- Welche Sprachen sind nicht regulär?
- Wie kann man Automaten systematisch vereinfachen?