Technische Universität Dresden

Formale Systeme

6. Vorlesung: Reguläre Ausdrücke

Markus Krötzsch

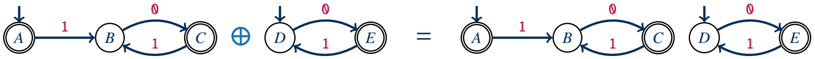
Professur für Wissensbasierte Systeme

TU Dresden, 30. Oktober 2025

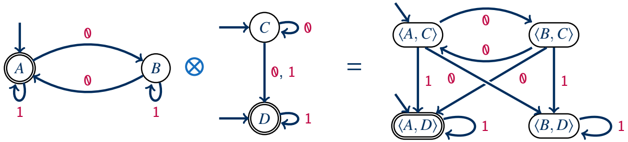
Rückblick

Wiederholung: Operationen auf Automaten

$\cup$



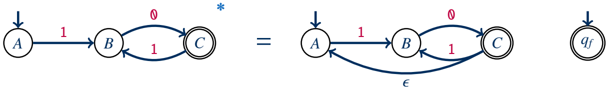
$\cap$



$-$

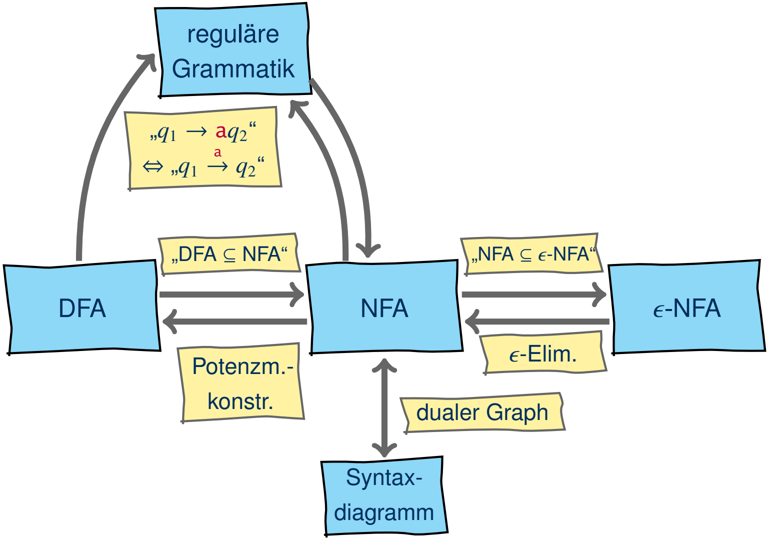
$\circ$

$*$



NFAs mit Wortübergängen

Darstellungen von Typ-3-Sprachen



Reguläre Ausdrücke

Endliche Sprachen

Eine einfache Beobachtung:

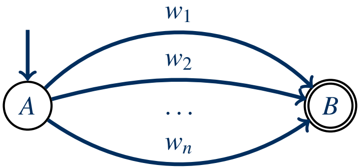
Satz: Jede endliche Sprache ist regulär.

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?

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 leicht aus den Sprachen $\emptyset$, $\{\epsilon\}$ und $\{\mathtt{a}\}$ ($\mathtt{a} \in \Sigma$) konstruieren.

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 leicht aus den Sprachen $\emptyset$, $\{\epsilon\}$ und $\{\mathtt{a}\}$ ($\mathtt{a} \in \Sigma$) 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 $\{\mathtt{a}\}$ für alle $\mathtt{a} \in \Sigma$
- Sie ist abgeschlossen unter den Operatoren $\cup$, $\circ$ und *

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 leicht aus den Sprachen $\emptyset$, $\{\epsilon\}$ und $\{\mathtt{a}\}$ $(\mathtt{a} \in \Sigma)$ 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 $\{\mathtt{a}\}$ für alle $\mathtt{a} \in \Sigma$
- 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 der regulärer Ausdrücke über einem Alphabet Σ ist induktiv[a] wie folgt definiert:

- $\emptyset$ ist ein regulärer Ausdruck
- ϵ ist ein regulärer Ausdruck
- $\mathtt{a}$ ist ein regulärer Ausdruck für jedes $\mathtt{a} \in \Sigma$
- Wenn α und β reguläre Ausdrücke sind,
  dann sind auch $(\alpha\beta)$, $(\alpha \mid \beta)$ und $(\alpha)^*$ 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 $(\mathtt{a}(\mathtt{b})^*)$ 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:

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

$$\mathsf{A} \to \sigma_1 \mid \ldots \mid \sigma_n$$

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 unterschiedliche Ausdrücke:

$$\alpha, \beta \to \emptyset \mid \epsilon \mid \Sigma \mid (\alpha\beta) \mid (\alpha \mid \beta) \mid (\alpha)^*$$

Semantik regulärer Ausdrücke

Reguläre Ausdrücke beschreiben die erwarteten Sprachen:

Die Sprache eines regulären Ausdrucks α wird mit $\mathbf{L}(\alpha)$ bezeichnet und rekursiv definiert:

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

Semantik regulärer Ausdrücke

Reguläre Ausdrücke beschreiben die erwarteten Sprachen:

Die Sprache eines regulären Ausdrucks α wird mit $\mathbf{L}(\alpha)$ bezeichnet und rekursiv definiert:

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

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

Äquivalenz regulärer Ausdrücke

Zwei reguläre Ausdrücke α und β sind 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: $\mathtt{ab}^* \mid \mathtt{bc}$ ist kurz für $((\mathtt{a}(\mathtt{b})^*) \mid (\mathtt{bc}))$

Vereinfachte Klammerung

Reguläre Ausdrücke können durch Klammerungsregeln vereinfacht werden: * bindet stärker als Konkatenation bindet stärker als |

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

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

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 $(\underbrace{\alpha \ldots \alpha}_{n\text{-mal}} \mid \ldots \mid \underbrace{\alpha \ldots \alpha}_{m\text{-mal}})$

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

Theoretiker:innen meiden Kurzformen (mehr Formen = 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 Muster in Datenbanken
- ...

Regexps in der Praxis

Reguläre Ausdrücke sind von großer praktischer Bedeutung

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

Unterschiede zur reinen Lehre:

- Pattern Matching: (1) spezifiziere eine Sprache (Suchwörter) und (2) finde deren Vorkommen (Matches) in einem längeren Wort (Text) $\rightsquigarrow$ 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 $\rightsquigarrow$ keine reguläre Sprache mehr
- Escaping: Unterscheidung von Steuerzeichen (Metazeichen) und Alphabetssymbolen ist praktisch aufwändig


reg·ex /^[a-z0-9]/
noun, plural regrets
Have you tried using an XML parser instead?

Von Regulären Ausdrücken zu Automaten

Zielstellung

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

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:

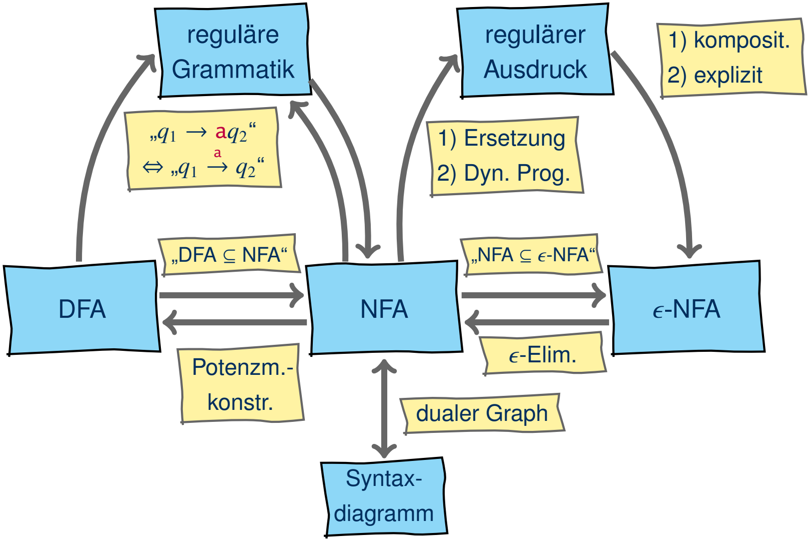
  (1) Kompositionelle Methode
  (2) Explizite Konstruktion

- Teilbehauptung 2: Für jeden NFA $\mathcal{M}$ gibt es einen regulären Ausdruck α, so dass $\mathbf{L}(\alpha) = \mathbf{L}(\mathcal{M})$

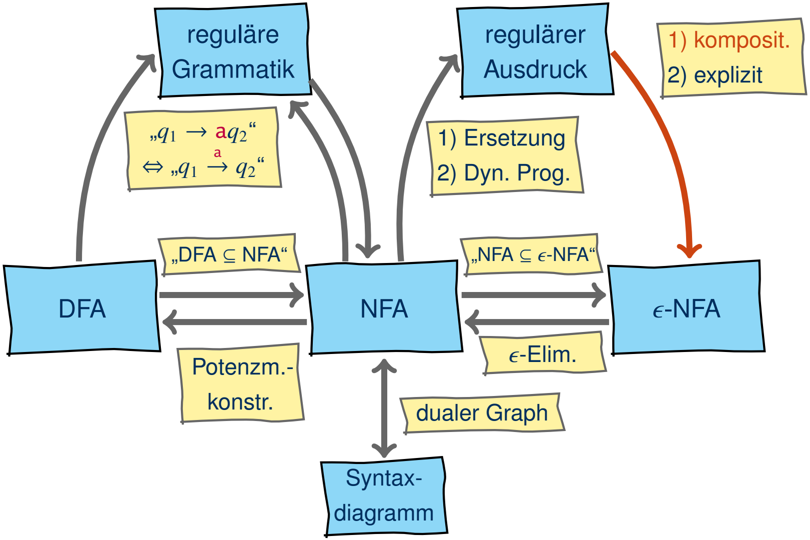
  Zwei mögliche Beweismethoden:

  (1) Ersetzungsmethode
  (2) Dynamische Programmierung

Darstellungen von Typ-3-Sprachen



Darstellungen von Typ-3-Sprachen



Rekursive Komposition von ϵ-NFA

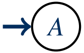
Die Struktur regulärer Ausdrücke kann mit Operationen auf Automaten direkt abgebildet werden.

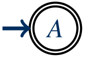
Rekursive Komposition von ϵ-NFA

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

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

Grundfälle:

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

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

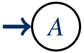
- Wenn $\alpha = \mathtt{a}$ dann $\mathcal{M}(\alpha) =$ 

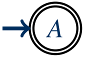
Rekursive Komposition von ϵ-NFA

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

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

Grundfälle:

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

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

- Wenn $\alpha = \mathtt{a}$ 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.

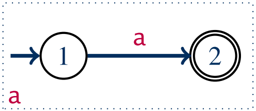
- Wenn $\alpha = (\beta\gamma)$ dann $\mathcal{M}(\alpha) = \text{elim}_\epsilon(\mathcal{M}(\beta) \odot \mathcal{M}(\gamma))$
- Wenn $\alpha = (\beta \mid \gamma)$ dann $\mathcal{M}(\alpha) = \mathcal{M}(\beta) \oplus \mathcal{M}(\gamma)$
- Wenn $\alpha = (\beta)^*$ dann $\mathcal{M}(\alpha) = \text{elim}_\epsilon(\mathcal{M}(\beta)^*)$

Beispiel

Regulärer Ausdruck: $\mathtt{a} \mid \mathtt{ba}^*$

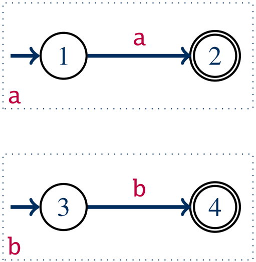
Beispiel

Regulärer Ausdruck: $\texttt{a} \mid \texttt{b}\texttt{a}^*$



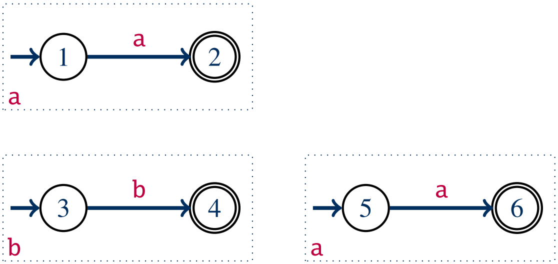
Beispiel

Regulärer Ausdruck: a | ba*



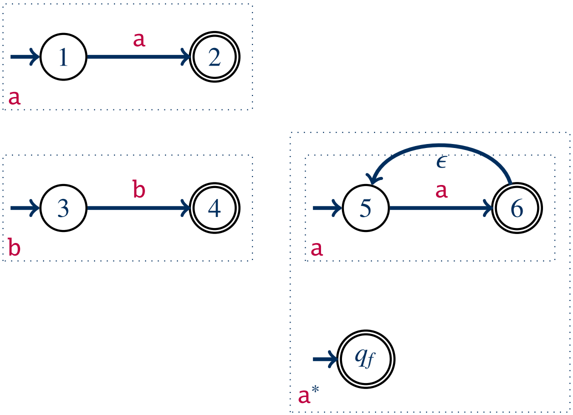
Beispiel

Regulärer Ausdruck: a | ba*



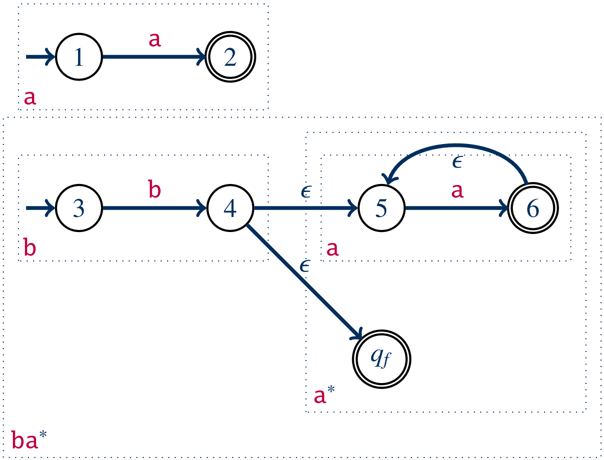
Beispiel

Regulärer Ausdruck: a | ba*



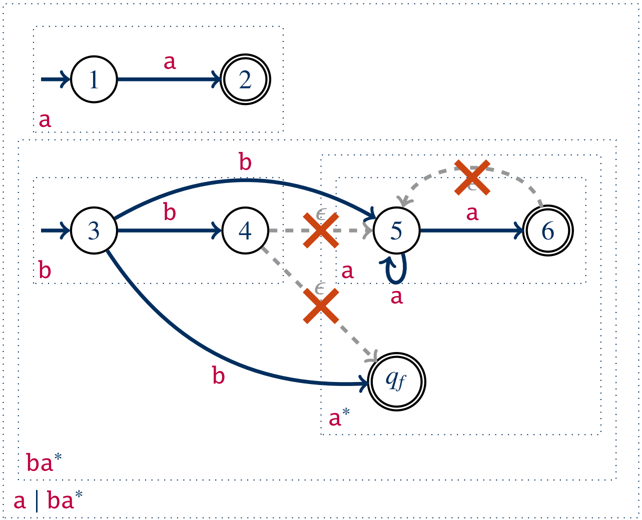
Beispiel

Regulärer Ausdruck: a | ba*



Beispiel

Regulärer Ausdruck: a | ba*



Korrektheit der Kompositionsmethode

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

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.

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 Struktur regulärer Ausdrücke.

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 Struktur regulärer Ausdrücke.

- Induktionsanfang: Für die Grundfälle $\alpha = \emptyset$, $\alpha = \epsilon$ und $\alpha = \mathtt{a}$ ist die Behauptung leicht zu sehen

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 Struktur regulärer Ausdrücke.

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

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 Struktur regulärer Ausdrücke.

- Induktionsanfang: Für die Grundfälle $\alpha = \emptyset$, $\alpha = \epsilon$ und $\alpha = \mathtt{a}$ ist die Behauptung leicht zu sehen
- Induktionshypothese (IH): 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:

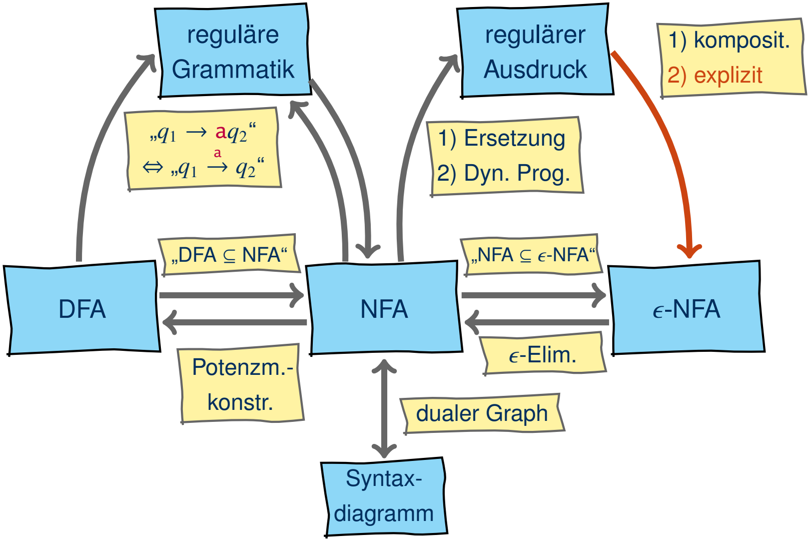
$$\mathbf{L}(\alpha) \stackrel{\text{Def.}}{=} \mathbf{L}(\beta) \circ \mathbf{L}(\gamma) \stackrel{\text{IH}}{=} \mathbf{L}(\mathcal{M}(\beta)) \circ \mathbf{L}(\mathcal{M}(\gamma))$$

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

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

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

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 $\mathbf{L}(\alpha) \neq \emptyset$) oder $\emptyset$ (falls $\mathbf{L}(\alpha) = \emptyset$)

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

Explizite Konstruktion von NFAs

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

Gegeben: regulärer Ausdruck α ohne $\emptyset$

Explizite Konstruktion von NFAs

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

Gegeben: 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:

Explizite Konstruktion von NFAs

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

Gegeben: regulärer Ausdruck α ohne $\emptyset$

Initialisierung: $\mathcal{M}_\alpha = \;\to A \xrightarrow{\alpha} B$

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 $q \xrightarrow{(\gamma_1\gamma_2)} p$ durch $q \xrightarrow{\gamma_1} r \xrightarrow{\gamma_2} p$

Explizite Konstruktion von NFAs

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

Gegeben: regulärer Ausdruck α ohne $\emptyset$

Initialisierung: $\mathcal{M}_\alpha = \ \to A \xrightarrow{\alpha} B$

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 $q \xrightarrow{(\gamma_1\gamma_2)} p$ durch $q \xrightarrow{\gamma_1} r \xrightarrow{\gamma_2} p$
- Ersetze $q \xrightarrow{(\gamma_1 \mid \gamma_2)} p$ durch
$$\begin{tikzcd} q \arrow[r, bend left, "\gamma_1"] \arrow[r, bend right, "\gamma_2"'] & p \end{tikzcd}$$

Explizite Konstruktion von NFAs

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

Gegeben: regulärer Ausdruck α ohne $\emptyset$

Initialisierung: $\mathcal{M}_\alpha = \;\to A \xrightarrow{\alpha} B$

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 $q \xrightarrow{(\gamma_1\gamma_2)} p$ durch $q \xrightarrow{\gamma_1} r \xrightarrow{\gamma_2} p$
- Ersetze $q \xrightarrow{(\gamma_1 \mid \gamma_2)} p$ durch $\begin{tikzcd} q \arrow[r, bend left, "\gamma_1"] \arrow[r, bend right, "\gamma_2"'] & p \end{tikzcd}$
- Ersetze $q \xrightarrow{(\gamma)^*} p$ durch $\begin{tikzcd} \arrow[r] & q \arrow[r, "\epsilon"] & r \arrow[loop above, "\gamma"] \arrow[r, "\epsilon"] & p \end{tikzcd}$

Beispiel

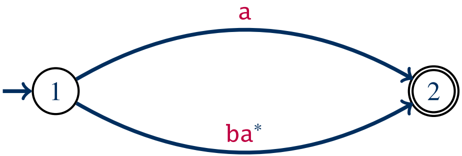
Regulärer Ausdruck: $\mathtt{a} \mid \mathtt{ba}^*$

Beispiel

Regulärer Ausdruck: $\mathtt{a} \mid \mathtt{ba}^*$

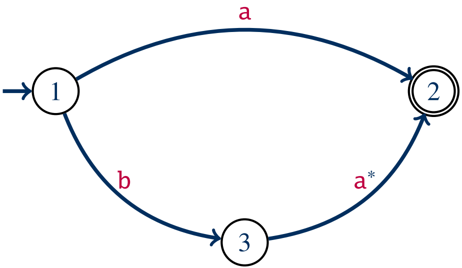
Beispiel

Regulärer Ausdruck: $\mathtt{a} \mid \mathtt{ba}^*$



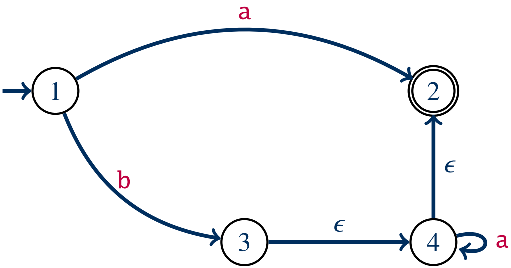
Beispiel

Regulärer Ausdruck: a | ba*



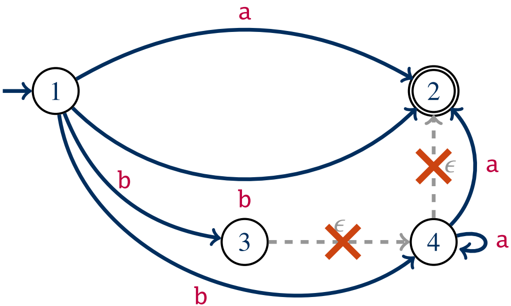
Beispiel

Regulärer Ausdruck: $\mathtt{a} \mid \mathtt{ba}^*$



Beispiel

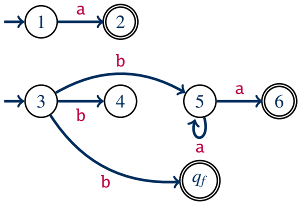
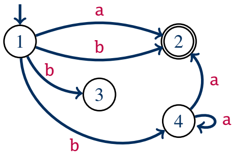
Regulärer Ausdruck: $\mathtt{a} \mid \mathtt{ba}^*$



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*	Linear viele Zustände*
Meist etwas größer	Meist etwas kleiner
Korrektheitsbeweis aus Abschlusseigenschaften	Korrektheitsbeweis erfordert neue Argumentation
Fast ohne Löschen (außer ϵ-Übergänge)	Übergänge werden oft gelöscht

*) Bzgl. Länge des reg. Ausdrucks bzw. bzgl. 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 zurück von NFA zu regulärem Ausdruck?
- Welche Sprachen sind nicht regulär?
- Wie kann man Automaten systematisch vereinfachen?