Technische Universität Dresden

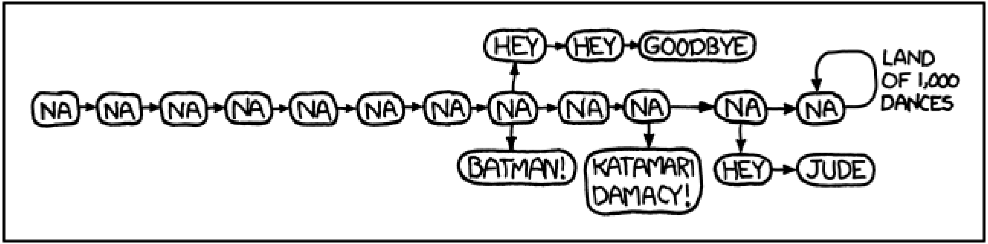
Formale Systeme

7. Vorlesung: Reguläre Ausdrücke

Markus Krötzsch

Professur für Wissensbasierte Systeme

TU Dresden, 3. November 2025



Randall Munroe, `https://xkcd.com/851_make_it_better/`, CC-BY-NC 2.5

Wiederholung: Reguläre Ausdrücke

- Reguläre Ausdrücke als Syntax für Sprachen, die durch Operationen aus endlichen Sprachen gebildet werden
- Grundformen: $\emptyset$, ϵ, $\mathtt{a}$ für alle $\mathtt{a} \in \Sigma$
- Operationen: Konkatenation, Alternative (|), Kleene-Stern (*)
- Viele weitere Ausdrucksmittel in praktischen „RegExps“

Kleene's Theorem

Satz („Kleene's Theorem"): Eine Sprache wird genau dann von einem regulären Ausdruck beschrieben, wenn sie von einem endlichen Automaten erkannt wird.

Letzte Vorlesung: „regulärer Ausdruck $\rightsquigarrow$ endlicher Automat"

- kompositionelle Methode
- explizite Methode

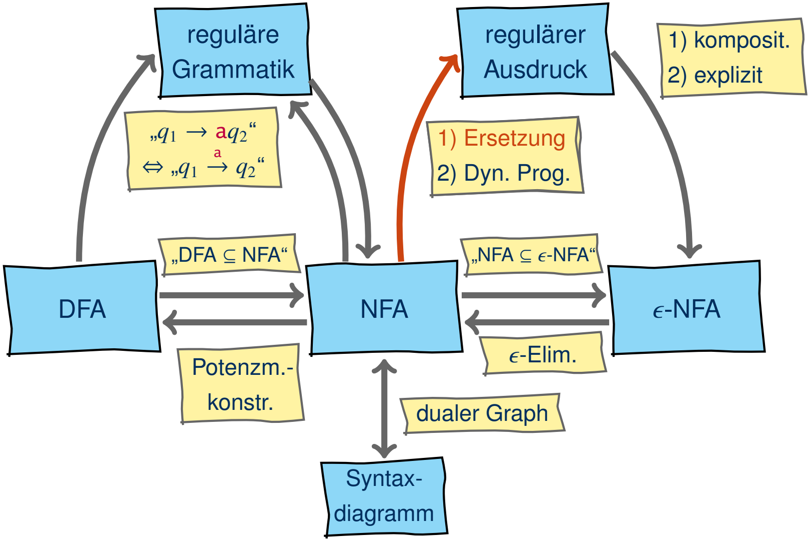
Heute: „endlicher Automat $\rightsquigarrow$ regulärer Ausdruck"

- Ersetzungsmethode
- Dynamische Programmierung

Stephen Cole Kleene 1978 *

Die Ersetzungsmethode

Darstellungen von Typ-3-Sprachen



NFA $\rightsquigarrow$ regulärer Ausdruck: Vorbereitung

Wir vereinfachen den NFA zunächst wie folgt:

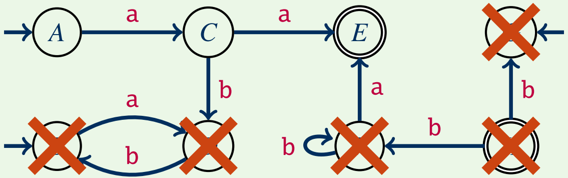
Gegeben: NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F \rangle$

- Entferne alle Zustände, die von keinem Zustand in Q_0 erreichbar sind
- Entferne alle Zustände, von denen kein Zustand in F erreichbar ist

Die Menge der von einem Zustand aus erreichbaren Zustände kann mit Graphalgorithmen berechnet werden, z.B. Breitensuche.

Offensichtlich verändert diese Vereinfachung die Sprache nicht

Beispiel:



Die Ersetzungsmethode

Gegeben: NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$

Gesucht: regulärer Ausdruck α mit $\mathbf{L}(\alpha) = \mathbf{L}(\mathcal{M})$

Ansatz:

Für jeden Zustand $q \in Q$, berechne einen regulären Ausdruck α_q für die Sprache

$$\begin{aligned}
\mathbf{L}(\alpha_q) &= \{w \in \Sigma^* \mid \text{es gibt ein } q_f \in F \text{ mit } q \xrightarrow{w} q_f\} \\
&= \{w \in \Sigma^* \mid \delta(q, w) \cap F \neq \emptyset\} \\
&= \mathbf{L}(\mathcal{M}_q) \qquad \text{mit } \mathcal{M}_q = \langle Q, \Sigma, \delta, \{q\}, F\rangle
\end{aligned}$$

Dann gilt:

$$\begin{aligned}
\mathbf{L}(\mathcal{M}) &= \bigcup_{q \in Q_0} \mathbf{L}(\alpha_q) \\
&= \mathbf{L}(\alpha_{q_1} \mid \alpha_{q_2} \mid \ldots \mid \alpha_{q_n}) \text{ mit } Q_0 = \{q_1, q_2, \ldots, q_n\}
\end{aligned}$$

Notation

Wir verwenden $\sum$ als verallgemeinerte Alternative:

Für eine endliche Menge $A = \{\alpha_1, \ldots, \alpha_n\}$ von regulären Ausdrücken schreiben wir $\sum_{\alpha \in A} \alpha$ als Abkürzung für $\alpha_1 \mid \ldots \mid \alpha_n$.

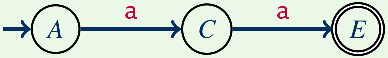
Wir wenden diese Notation auch in anderen ähnlichen Fällen an, zum Beispiel für den vorigen Ausdruck:

$$\sum_{q \in Q_0} \alpha_q = \alpha_{q_1} \mid \alpha_{q_2} \mid \ldots \mid \alpha_{q_n}$$

Ersetzungsmethode – Schwierigkeit

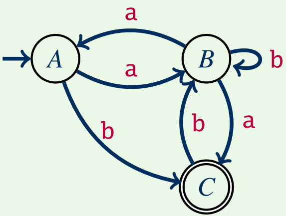
Wie kann man die regulären Ausdrücke α_q finden?

Beispiel: ohne Rekursion ist es einfach . . .



$$\alpha_A = \mathtt{aa},$$
$$\alpha_C = \mathtt{a},$$
$$\alpha_E = \epsilon$$

Beispiel: mit Rekursion ist es weniger klar . . .

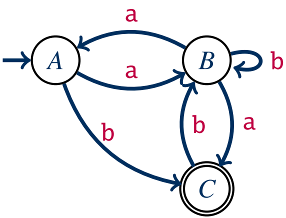


$$\alpha_A = ?$$

Ersetzungsmethode – Rekursion

Idee: rekursiver Automat $\rightsquigarrow$ rekursive Definition von α_q

Beispiel:



$$\alpha_A \equiv \mathtt{a}\alpha_B \mid \mathtt{b}\alpha_C$$

$$\alpha_B \equiv \mathtt{a}\alpha_A \mid \mathtt{a}\alpha_C \mid \mathtt{b}\alpha_B$$

$$\alpha_C \equiv \mathtt{b}\alpha_B \mid \epsilon$$

$\rightsquigarrow$ Ein Gleichungssystem von regulären Ausdrücken!

Ersetzungsmethode – Gleichungen

Allgemein kann man das Gleichungssystem wie folgt beschreiben:

Für einen NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$ betrachten wir die folgenden Gleichungen für Ausdrücke α_q mit $q \in Q$.

- Für jeden Zustand $q \in Q \setminus F$:

$$\alpha_q \equiv \sum_{\mathtt{a}\in\Sigma} \sum_{p\in\delta(q,\mathtt{a})} \mathtt{a}\alpha_p$$

- Für jeden Zustand $q \in F$:

$$\alpha_q \equiv \epsilon \mid \sum_{\mathtt{a}\in\Sigma} \sum_{p\in\delta(q,\mathtt{a})} \mathtt{a}\alpha_p$$

Jetzt müssen wir diese Gleichungen lediglich lösen ...

Gleichungssysteme Lösen

$$\alpha_A \equiv \mathtt{a}\alpha_B \mid \mathtt{b}\alpha_C \qquad \alpha_B \equiv \mathtt{a}\alpha_A \mid \mathtt{a}\alpha_C \mid \mathtt{b}\alpha_B \qquad \alpha_C \equiv \mathtt{b}\alpha_B \mid \epsilon$$

Wie können wir solche Gleichungssysteme lösen?

- **Methode 1:** Gleichungen ineinander Einsetzen und das Ergebnis vereinfachen

  Beispiel: Setzen wir die Definition von α_C in die Gleichung für α_A ein, so erhalten wir

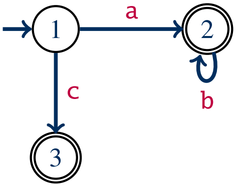
  $$\alpha_A \equiv \mathtt{a}\alpha_B \mid \mathtt{b}(\mathtt{b}\alpha_B \mid \epsilon) \equiv \mathtt{a}\alpha_B \mid \mathtt{bb}\alpha_B \mid \mathtt{b} \equiv (\mathtt{a} \mid \mathtt{bb})\alpha_B \mid \mathtt{b}.$$

  Problem: rekursive Gleichungen lassen sich so nicht vereinfachen …

- **Methode 2:** Rekursive Gleichungen direkt Lösen

  Regel von Arden: Aus $\alpha \equiv \beta\alpha \mid \gamma$ mit $\epsilon \notin \mathbf{L}(\beta)$ folgt $\alpha \equiv \beta^*\gamma$.

Beispiel: Gleichungssysteme Lösen



(1) $\alpha_1 \equiv \mathtt{a}\alpha_2 \mid \mathtt{c}\alpha_3$

(2) $\alpha_2 \equiv \mathtt{b}\alpha_2 \mid \epsilon$

(3) $\alpha_3 \equiv \epsilon$

(4) $\alpha_2 \equiv \mathtt{b}^*\epsilon \equiv \mathtt{b}^*$ Arden (2)

(5) $\alpha_1 \equiv \mathtt{ab}^* \mid \mathtt{c}\alpha_3$ (4) in (1)

(6) $\alpha_1 \equiv \mathtt{ab}^* \mid \mathtt{c}$ (3) in (5)

Regel von Arden:
Aus $\alpha \equiv \beta\alpha \mid \gamma$ mit $\epsilon \notin \mathbf{L}(\beta)$ folgt $\alpha \equiv \beta^*\gamma$.

$\leadsto$ regulärer Ausdruck für NFA ist $\sum_{q \in Q_0} \alpha_q = \alpha_1$, also $\mathtt{ab}^* \mid \mathtt{c}$

Korrektheit der Ersetzungsregel (1)

Regel von Arden*: Aus $\alpha \equiv \beta\alpha \mid \gamma$ mit $\epsilon \notin \mathbf{L}(\beta)$ folgt $\alpha \equiv \beta^*\gamma$.

Beweis: Wir behaupten: Wenn $\mathbf{L}(\alpha) = \mathbf{L}(\beta) \circ \mathbf{L}(\alpha) \cup \mathbf{L}(\gamma)$ mit $\epsilon \notin \mathbf{L}(\beta)$ dann gilt $\mathbf{L}(\alpha) = \mathbf{L}(\beta)^* \circ \mathbf{L}(\gamma)$.

Wir zeigen: dies gilt nicht nur für $\mathbf{L}(\alpha)$, $\mathbf{L}(\beta)$ und $\mathbf{L}(\gamma)$, sondern für beliebige Sprachen $\mathbf{L}$, $\mathbf{K}$ und $\mathbf{H}$:

$$\text{Wenn } \mathbf{L} = \mathbf{K}\mathbf{L} \cup \mathbf{H} \text{ und } \epsilon \notin \mathbf{K} \text{ dann } \mathbf{L} = \mathbf{K}^*\mathbf{H}$$

Wir zeigen die beiden Richtungen der geforderten Gleichheit einzeln.

* nach Dean N. Arden der das Resultat 1961 publizierte; auch bekannt als „Lemma von Arden“

Korrektheit der Ersetzungsregel (2)

Annahme: $\mathbf{L} = \mathbf{K}\mathbf{L} \cup \mathbf{H}$ und $\epsilon \notin \mathbf{K}$

Teilbehauptung 1: $\mathbf{K}^*\mathbf{H} \subseteq \mathbf{L}$

- Sei $w \in \mathbf{K}^*\mathbf{H}$ beliebig
- Dann hat w die Form $u_1 \cdots u_n v$ mit $n \geq 0$, $u_1, \ldots, u_n \in \mathbf{K}$ und $v \in \mathbf{H}$
- Wegen $\mathbf{L} = \mathbf{K}\mathbf{L} \cup \mathbf{H}$ gilt $\mathbf{K}\mathbf{L} \subseteq \mathbf{L}$ und $\mathbf{H} \subseteq \mathbf{L}$
- Wegen $v \in \mathbf{H}$ und $\mathbf{H} \subseteq \mathbf{L}$ gilt $v \in \mathbf{L}$
- Wegen $v \in \mathbf{L}$, $u_n \in \mathbf{K}$ und $\mathbf{K}\mathbf{L} \subseteq \mathbf{L}$ gilt $u_n v \in \mathbf{L}$
- Wegen $u_n v \in \mathbf{L}$, $u_{n-1} \in \mathbf{K}$ und $\mathbf{K}\mathbf{L} \subseteq \mathbf{L}$ gilt $u_{n-1} u_n v \in \mathbf{L}$

  ...
- Wegen $u_2 \cdots u_n v \in \mathbf{L}$, $u_1 \in \mathbf{K}$ und $\mathbf{K}\mathbf{L} \subseteq \mathbf{L}$ gilt $\underbrace{u_1 \cdots u_n v}_{w} \in \mathbf{L}$

Korrektheit der Ersetzungsregel (3)

Annahme: $\mathbf{L} = \mathbf{K}\mathbf{L} \cup \mathbf{H}$ und $\epsilon \notin \mathbf{K}$

Teilbehauptung 2: $\mathbf{L} \subseteq \mathbf{K}^*\mathbf{H}$

Sei $w \in \mathbf{L}$ beliebig. Wir beweisen $w \in \mathbf{K}^*\mathbf{H}$ durch Induktion über $n = |w|$.

Induktionsanfang: sei $n = 0$

- Dann ist $w = \epsilon$
- Weil $\epsilon \notin \mathbf{K}$ gilt $\epsilon \notin \mathbf{K}\mathbf{L}$
- Da $w = \epsilon \in \mathbf{L}$ und $\mathbf{L} = \mathbf{K}\mathbf{L} \cup \mathbf{H}$ gilt also $\epsilon \in \mathbf{H}$
- Also gilt $w \in \mathbf{K}^*\mathbf{H}$.

Korrektheit der Ersetzungsregel (4)

Annahme: $\mathbf{L} = \mathbf{K}\mathbf{L} \cup \mathbf{H}$ und $\epsilon \notin \mathbf{K}$

Teilbehauptung 2: $\mathbf{L} \subseteq \mathbf{K}^*\mathbf{H}$

Induktionshypothese: Die Aussage gilt für alle Wörter v mit $|v| < n$, d.h., für jedes solches $v \in \mathbf{L}$ gilt auch $v \in \mathbf{K}^*\mathbf{H}$

Induktionsschritt: sei $|w| = n$

- Wegen $\mathbf{L} = \mathbf{K}\mathbf{L} \cup \mathbf{H}$ gilt (1) $w \in \mathbf{H}$ oder (2) $w \in \mathbf{K}\mathbf{L}$
- Fall 1 $w \in \mathbf{H}$: dann ist $w = \epsilon w \in \mathbf{K}^*\mathbf{H}$
- Fall 2 $w \in \mathbf{K}\mathbf{L}$:
  - Dann gibt es $u \in \mathbf{K}$ und $v \in \mathbf{L}$ mit $w = uv$
  - Wegen $\epsilon \notin \mathbf{K}$ ist $u \neq \epsilon$ und also $|v| < |w| = n$
  - Laut IH gilt also $v \in \mathbf{K}^*\mathbf{H}$
  - Wegen $u \in \mathbf{K}$ gilt also auch $uv = w \in \mathbf{K}^*\mathbf{H}$

Damit ist der Beweis abgeschlossen. $\square$

Zusammenfassung Ersetzungsmethode

Die Umwandlung NFA $\rightsquigarrow$ regulärer Ausdruck ist also wie folgt:

(1) Vereinfache den Automaten (entferne offensichtlich unnötige Zustände)

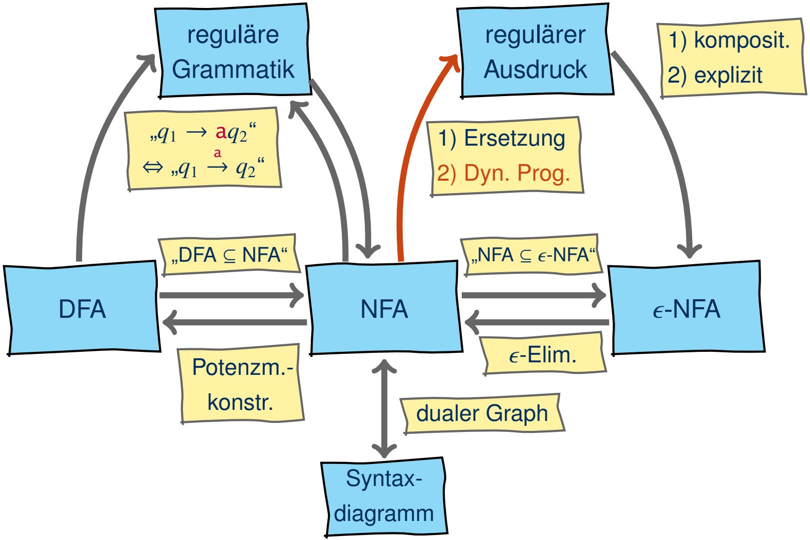
(2) Bestimme das Gleichungssystem (eine Gleichung pro Zustand)

(3) Löse das Gleichungssystem (durch Einsetzen und Ardens Regel)

(4) Gib den Ausdruck für die Sprache des NFA an (Alternative der Ausdrücke für alle Anfangszustände)

Ermittlung regulärer Ausdrücke durch dynamische Programmierung

Darstellungen von Typ-3-Sprachen



Idee

Gegeben: NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$

Gesucht: regulärer Ausdruck α mit $\mathbf{L}(\alpha) = \mathbf{L}(\mathcal{M})$

Ansatz:

Für jedes Paar von Zuständen $q, p \in Q$, berechne einen regulären Ausdruck $\alpha_{q,p}$ für die Sprache

$$\begin{aligned}\mathbf{L}(\alpha_{q,p}) &= \{w \in \Sigma^* \mid q \xrightarrow{w} p\} \\ &= \{w \in \Sigma^* \mid p \in \delta(q, w)\} \\ &= \mathbf{L}(\mathcal{M}_{q,p}) \qquad \text{mit } \mathcal{M}_{q,p} = \langle Q, \Sigma, \delta, \{q\}, \{p\}\rangle\end{aligned}$$

Dann gilt:

$$\mathbf{L}(\mathcal{M}) = \bigcup_{q \in Q_0} \bigcup_{p \in F} \mathbf{L}(\alpha_{q,p}) = \mathbf{L}\left(\sum_{q \in Q_0} \sum_{p \in F} \alpha_{q,p}\right)$$

Dynamische Ermittlung von $\alpha_{q,p}$

Gegeben: NFA $\mathcal{M} = \langle Q, \Sigma, \delta, Q_0, F\rangle$

Annahme: Zustände sind nummeriert: $Q = \{q_1, \ldots, q_n\}$ (o.B.d.A.)

Gegeben $\mathcal{M}$, Zahlen $i, j \in \{1, \ldots, n\}$ und eine Zahl $k \in \{0, 1, \ldots, n\}$ definieren wir die Sprache $\mathbf{L}^k[i, j]$ als die Menge aller Wörter $w = \mathtt{a}_1 \cdots \mathtt{a}_\ell$ für die gilt:

- es gibt einen Lauf $q_i \xrightarrow{\mathtt{a}_1} p_1 \xrightarrow{\mathtt{a}_2} \ldots \xrightarrow{\mathtt{a}_{\ell-1}} p_{\ell-1} \xrightarrow{\mathtt{a}_\ell} q_j$, wobei
- für jeden Zwischenzustand p_z mit $z \in \{1, \ldots, \ell - 1\}$ gilt $p_z \in \{q_1, \ldots, q_k\}$

Gesucht: Reguläre Ausdrücke $\alpha^k[i, j]$ mit $\mathbf{L}(\alpha^k[i, j]) = \mathbf{L}^k[i, j]$.

Wir wollen dynamische Programmierung anwenden, um $\alpha^k[i, j]$ für immer größere Werte k zu berechnen.

Der Fall $k = n$

Gegeben $\mathcal{M}$, Zahlen $i, j \in \{1, \ldots, n\}$ und eine Zahl $k \in \{0, 1, \ldots, n\}$ definieren wir die Sprache $\mathbf{L}^k[i, j]$ als die Menge aller Wörter $w = \mathtt{a}_1 \cdots \mathtt{a}_\ell$ für die gilt:

- es gibt einen Lauf $q_i \xrightarrow{\mathtt{a}_1} p_1 \xrightarrow{\mathtt{a}_2} \ldots \xrightarrow{\mathtt{a}_{\ell-1}} p_{\ell-1} \xrightarrow{\mathtt{a}_\ell} q_j$, wobei
- für jeden Zwischenzustand p_z mit $z \in \{1, \ldots, \ell - 1\}$ gilt $p_z \in \{q_1, \ldots, q_k\}$

Für $k = n$ ist die zweite Bedingung immer erfüllt, da $\{q_1, \ldots, q_n\} = Q$

$\leadsto$ $\mathbf{L}^n[i, j]$ ist die Menge aller Wörter „zwischen“ q_i und q_j

$\leadsto$ $\alpha^n[i, j] = \alpha_{q_i, q_j}$ sind die regulären Ausdrücke, aus denen wir letztlich die Lösung ermitteln wollen

Der Fall $k = 0$

Gegeben $\mathcal{M}$, Zahlen $i, j \in \{1, \ldots, n\}$ und eine Zahl $k \in \{0, 1, \ldots, n\}$ definieren wir die Sprache $\mathbf{L}^k[i, j]$ als die Menge aller Wörter $w = \mathtt{a}_1 \cdots \mathtt{a}_\ell$ für die gilt:

- es gibt einen Lauf $q_i \xrightarrow{\mathtt{a}_1} p_1 \xrightarrow{\mathtt{a}_2} \ldots \xrightarrow{\mathtt{a}_{\ell-1}} p_{\ell-1} \xrightarrow{\mathtt{a}_\ell} q_j$, wobei
- für jeden Zwischenzustand p_z mit $z \in \{1, \ldots, \ell - 1\}$ gilt $p_z \in \{q_1, \ldots, q_k\}$

Für $k = 0$ kann die zweite Bedingung für keinen Zustand p_i erfüllt werden

$\leadsto$ $\mathbf{L}^0[i, j]$ beruht nur auf Läufen ohne Zwischenzustände

- Falls $i \neq j$, dann kommen nur Läufe $q_i \xrightarrow{\mathtt{a}} q_j$ in Frage
- Falls $i = j$, dann kommen Läufe $q_i \xrightarrow{\mathtt{a}} q_i$ ($w = \mathtt{a}$) oder q_i ($w = \epsilon$) in Frage

$\leadsto$ reguläre Ausdrücke $\alpha^0[i, j]$ können direkt aus $\mathcal{M}$ abgelesen werden

Die regulären Ausdrücke $\alpha^0[i,j]$

Für $k = 0$ können wir $\alpha^0[i,j]$ direkt aus $\mathcal{M}$ ablesen:

Sei $\{\mathtt{a}_1, \ldots, \mathtt{a}_m\} = \{\mathtt{a} \in \Sigma \mid q_i \xrightarrow{\mathtt{a}} q_j\}$ die Menge der Beschriftungen von direkten Übergängen von q_i zu q_j.

- Falls $i \neq j$, dann ist
$$\alpha^0[i,j] = \mathtt{a}_1 \mid \ldots \mid \mathtt{a}_m$$
- Falls $i = j$, dann ist
$$\alpha^0[i,j] = \mathtt{a}_1 \mid \ldots \mid \mathtt{a}_m \mid \epsilon$$

Die regulären Ausdrücke $\alpha^{k+1}[i,j]$

Zur Bestimmung von $\alpha^{k+1}[i,j]$ verwenden wir Ausdrücke $\alpha^{k}[i',j']$

- es gibt einen Lauf $q_i \xrightarrow{\mathtt{a}_1} p_1 \xrightarrow{\mathtt{a}_2} \ldots \xrightarrow{\mathtt{a}_{\ell-1}} p_{\ell-1} \xrightarrow{\mathtt{a}_\ell} q_j$, wobei
- für jedes p_z mit $z \in \{1,\ldots,\ell-1\}$ gilt $p_z \in \{q_1,\ldots,q_k\}$

$\leadsto$ zwei Möglichkeiten für Läufe bei $k+1$:

(1) kein p_i ist q_{k+1}, d.h. $p_i \in \{q_1,\ldots,q_k\}$

(2) einige p_i sind q_{k+1}; dann hat der Lauf die Form:

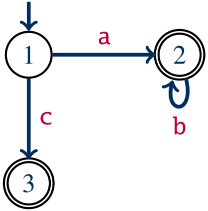
$$q_i\{q_1,\ldots,q_k\}^*q_{k+1}(\{q_1,\ldots,q_k\}^*q_{k+1})^*\{q_1,\ldots,q_k\}^*q_j$$

Teilläufe: $q_i \to q_{k+1}$ $\quad$ ($q_{k+1} \to q_{k+1}$)* $\quad$ $q_{k+1} \to q_j$

Daher gilt:

$$\alpha^{k+1}[i,j] = \underbrace{\alpha^{k}[i,j]}_{\text{Fall (1)}} \mid \underbrace{(\alpha^{k}[i,k+1](\alpha^{k}[k+1,k+1])^*\alpha^{k}[k+1,j])}_{\text{Fall (2)}}$$

Beispiel: Dynamische Programmierung (1)



Fall $k = 0$:

$\alpha^0[1,1] = \epsilon$ $\quad \alpha^0[1,2] = \mathtt{a}$ $\quad \alpha^0[1,3] = \mathtt{c}$

$\alpha^0[2,1] = \emptyset$ $\quad \alpha^0[2,2] = \mathtt{b} \mid \epsilon$ $\quad \alpha^0[2,3] = \emptyset$

$\alpha^0[3,1] = \emptyset$ $\quad \alpha^0[3,2] = \emptyset$ $\quad \alpha^0[3,3] = \epsilon$

Fall $k = 1$:

$$\alpha^1[1,1] = \underbrace{\alpha^0[1,1]}_{\epsilon} \mid \underbrace{(\alpha^0[1,1](\alpha^0[1,1])^*\alpha^0[1,1])}_{\epsilon\epsilon^*\epsilon} \equiv \epsilon$$

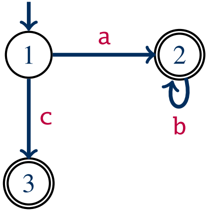
$$\alpha^1[1,2] = \underbrace{\alpha^0[1,2]}_{\mathtt{a}} \mid \underbrace{(\alpha^0[1,1](\alpha^0[1,1])^*\alpha^0[1,2])}_{\epsilon\epsilon^*\mathtt{a}} \equiv \mathtt{a}$$

$\ldots$ syntaktische, aber keine semantischen Änderungen

$$\alpha^1[i,j] \equiv \alpha^0[i,j]$$

(Grund: es gibt keine Pfade zu 1 oder von 1 zu 1)

Beispiel: Dynamische Programmierung (2)



Fall $k = 1$:

$\alpha^1[1,1] \equiv \epsilon$ $\quad \alpha^1[1,2] \equiv \mathtt{a}$ $\quad \alpha^1[1,3] \equiv \mathtt{c}$

$\alpha^1[2,1] \equiv \emptyset$ $\quad \alpha^1[2,2] \equiv \mathtt{b} \mid \epsilon$ $\quad \alpha^1[2,3] \equiv \emptyset$

$\alpha^1[3,1] \equiv \emptyset$ $\quad \alpha^1[3,2] \equiv \emptyset$ $\quad \alpha^1[3,3] \equiv \epsilon$

Fall $k = 2$:

$$\alpha^2[1,1] = \alpha^1[1,1] \mid (\alpha^1[1,2](\alpha^1[2,2])^*\alpha^1[2,1]) = \epsilon \mid (\mathtt{a}(\mathtt{b} \mid \epsilon)^*\emptyset) \equiv \epsilon$$

$$\alpha^2[1,2] = \alpha^1[1,2] \mid (\alpha^1[1,2](\alpha^1[2,2])^*\alpha^1[2,2]) = \mathtt{a} \mid (\mathtt{a}(\mathtt{b} \mid \epsilon)^*(\mathtt{b} \mid \epsilon)) \equiv \mathtt{ab}^*$$

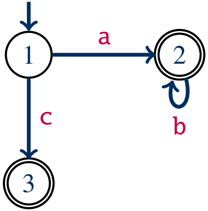
$$\alpha^2[1,3] = \alpha^1[1,3] \mid (\alpha^1[1,2](\alpha^1[2,2])^*\alpha^1[2,3]) = \mathtt{c} \mid (\mathtt{a}(\mathtt{b} \mid \epsilon)^*\emptyset) \equiv \mathtt{c}$$

$\ldots$

$\alpha^2[2,1] \equiv \emptyset$ $\quad \alpha^2[2,2] \equiv \mathtt{b}^*$ $\quad \alpha^2[2,3] \equiv \emptyset$

$\alpha^2[3,1] \equiv \emptyset$ $\quad \alpha^2[3,2] \equiv \emptyset$ $\quad \alpha^2[3,3] \equiv \epsilon$

Beispiel: Dynamische Programmierung (3)



Fall $k = 2$:

$\alpha^2[1,1] \equiv \epsilon$ $\quad \alpha^2[1,2] \equiv \mathtt{ab}^*$ $\quad \alpha^2[1,3] \equiv \mathtt{c}$

$\alpha^2[2,1] \equiv \emptyset$ $\quad \alpha^2[2,2] \equiv \mathtt{b}^*$ $\quad \alpha^2[2,3] \equiv \emptyset$

$\alpha^2[3,1] \equiv \emptyset$ $\quad \alpha^2[3,2] \equiv \emptyset$ $\quad \alpha^2[3,3] \equiv \epsilon$

Fall $k = 3$:

syntaktische, aber keine semantischen Änderungen:

$$\alpha^3[i,j] \equiv \alpha^2[i,j]$$

(Grund: es gibt keine Pfade von 3 zu 3)

Damit sind alle $\alpha^3[i,j] = \alpha^n[i,j]$ bestimmt und wir erhalten den folgenden regulären Ausdruck für den Automaten:

$$\alpha^3[1,2] \mid \alpha^3[1,3] \equiv \mathtt{ab}^* \mid \mathtt{c}$$

Zusammenfassung und Ausblick

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

Die Ersetzungsmethode definiert und löst ein Gleichungssystem, um aus einem NFA einen regulären Ausdruck zu erzeugen

Die Methode der dynamischen Programmierung berechnet reguläre Ausdrücke für Wörter „zwischen" Zustandspaaren, wobei immer größere Teilmengen von Zwischenzuständen verwendet werden dürfen

Offene Fragen:

- Wie aufwändig sind diese Umformungen im schlimmsten Fall?
- Welche Sprachen sind nicht regulär?
- Wie kann man Automaten systematisch vereinfachen?