

FORMALE SYSTEME

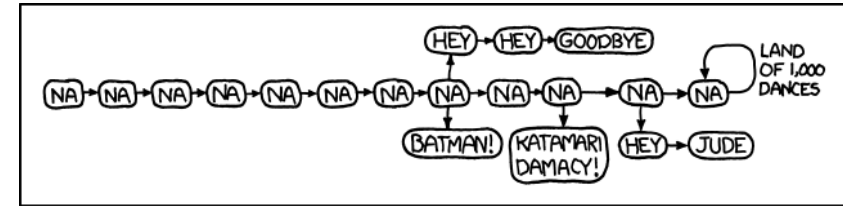
7. Vorlesung: Reguläre Ausdrücke

Markus Krötzsch
Daniel Borchmann

TU Dresden, 3. November 2016

Wiederholung: Reguläre Ausdrücke

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



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

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 \leadsto endlicher Automat“

- kompositionelle Methode
- explizite Methode

Heute: „endlicher Automat \leadsto regulärer Ausdruck“

- Ersetzungsmethode
- Dynamische Programmierung

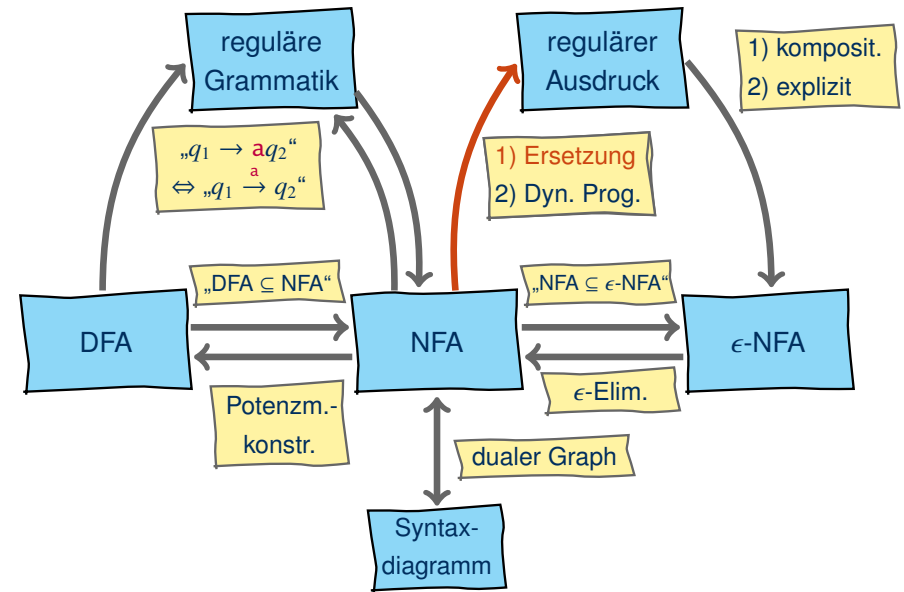


Stephen Cole Kleene 1978 *

*) Konrad Jacobs, Erlangen, © Mathematisches Forschungsinstitut Oberwolfach, CC-BY-SA de 2.0

Die Ersetzungsmethode

Darstellungen von Typ-3-Sprachen



NFA \rightsquigarrow regulärer Ausdruck: Vorbereitung

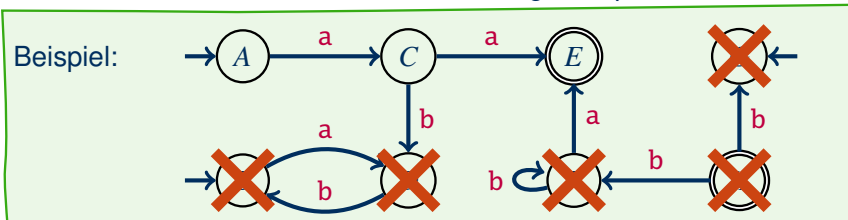
Wir vereinfachen den NFA zunächst wie folgt:

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

- Entferne alle Zustände, die von keinem Zustand in Q_0 erreichbar sind
- Entferne alle Zustände, die 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



Die Ersetzungsmethode

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

Gesucht: regulärer Ausdruck α mit $L(\alpha) = L(M)$

Ansatz:

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

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

Dann gilt:

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

Notation

Wir verwenden \sum als verallgemeinerte Alternative:

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

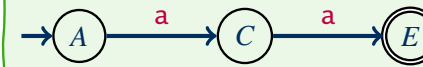
Wir wenden diese Notation auch in anderen ähnlichen Fällen an, zum Beispiel für den vorigen Ausdruck:

$$\sum_{q \in Q_0} \alpha_q = \alpha_{q_1} | \alpha_{q_2} | \dots | \alpha_{q_n}$$

Ersetzungsmethode – Schwierigkeit

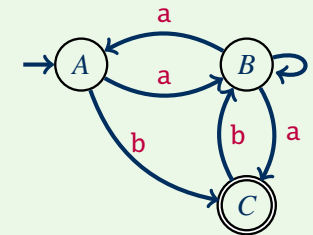
Wie kann man die regulären Ausdrücke α_q finden?

Beispiel: ohne Rekursion ist es einfach ...



$$\begin{aligned} \alpha_A &= \mathbf{aa}, \\ \alpha_C &= \mathbf{a}, \\ \alpha_E &= \epsilon \end{aligned}$$

Beispiel: mit Rekursion ist es weniger klar ...

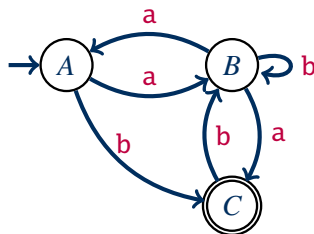


$$\alpha_A = ?$$

Ersetzungsmethode – Rekursion

Idee: rekursiver Automat \rightsquigarrow rekursive Definition von α_q

Beispiel:



$$\begin{aligned} \alpha_A &\equiv \mathbf{a}\alpha_B | \mathbf{b}\alpha_C \\ \alpha_B &\equiv \mathbf{a}\alpha_A | \mathbf{a}\alpha_C | \mathbf{b}\alpha_B \\ \alpha_C &\equiv \mathbf{b}\alpha_B | \epsilon \end{aligned}$$

\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 $q \in Q \setminus F$:

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

- Für jeden Zustand $q \in F$:

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

Jetzt müssen wir diese Gleichungen lediglich lösen ...

Gleichungssysteme Lösen

$$\alpha_A \equiv a\alpha_B \mid b\alpha_C \quad \alpha_B \equiv a\alpha_A \mid a\alpha_C \mid b\alpha_B \quad \alpha_C \equiv b\alpha_B \mid \epsilon$$

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

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

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 $\epsilon \notin L(\beta)$ folgt $\alpha \equiv \beta^*\gamma$.

Korrektheit der Ersetzungsregel (1)

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

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

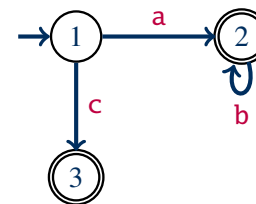
Wir zeigen: dies gilt nicht nur für $L(\alpha)$, $L(\beta)$ und $L(\gamma)$, sondern für beliebige Sprachen L , K und H :

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

Wir zeigen die beiden Richtungen der geforderten Gleichheit einzeln.

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

Beispiel: Gleichungssysteme Lösen



- (1) $\alpha_1 \equiv a\alpha_2 \mid c\alpha_3$
- (2) $\alpha_2 \equiv b\alpha_2 \mid \epsilon$
- (3) $\alpha_3 \equiv \epsilon$

- (4) $\alpha_2 \equiv b^*\epsilon \equiv b^*$ Arden (2)
- (5) $\alpha_1 \equiv ab^* \mid c\alpha_3$ (4) in (1)
- (6) $\alpha_1 \equiv ab^* \mid c$ (3) in (5)

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

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

Korrektheit der Ersetzungsregel (2)

Annahme: $L = KL \cup H$ und $\epsilon \notin K$

Teilbehauptung 1: $K^*H \subseteq L$

- Sei $w \in K^*H$ beliebig
- Dann hat w die Form $u_1 \cdots u_n v$ mit $n \geq 0$, $u_1, \dots, u_n \in K$ und $v \in H$
- Wegen $L = KL \cup H$ gilt $KL \subseteq L$ und $H \subseteq L$
- Wegen $v \in H$ und $H \subseteq L$ gilt $v \in L$
- Wegen $v \in L$, $u_n \in K$ und $KL \subseteq L$ gilt $u_n v \in L$
- Wegen $u_n v \in L$, $u_{n-1} \in K$ und $KL \subseteq L$ gilt $u_{n-1} u_n v \in L$
- ...
- Wegen $u_2 \cdots u_n v \in L$, $u_1 \in K$ und $KL \subseteq L$ gilt $\underbrace{u_1 \cdots u_n v}_w \in L$

Korrektheit der Ersetzungsregel (3)

Annahme: $L = KL \cup H$ und $\epsilon \notin K$

Teilbehauptung 2: $L \subseteq K^*H$

Sei $w \in L$ beliebig. Wir beweisen $w \in K^*H$ durch Induktion über $|w|$.

Induktionsanfang: sei $n = 0$

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

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)

Korrektheit der Ersetzungsregel (4)

Annahme: $L = KL \cup H$ und $\epsilon \notin K$

Teilbehauptung 2: $L \subseteq K^*H$

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

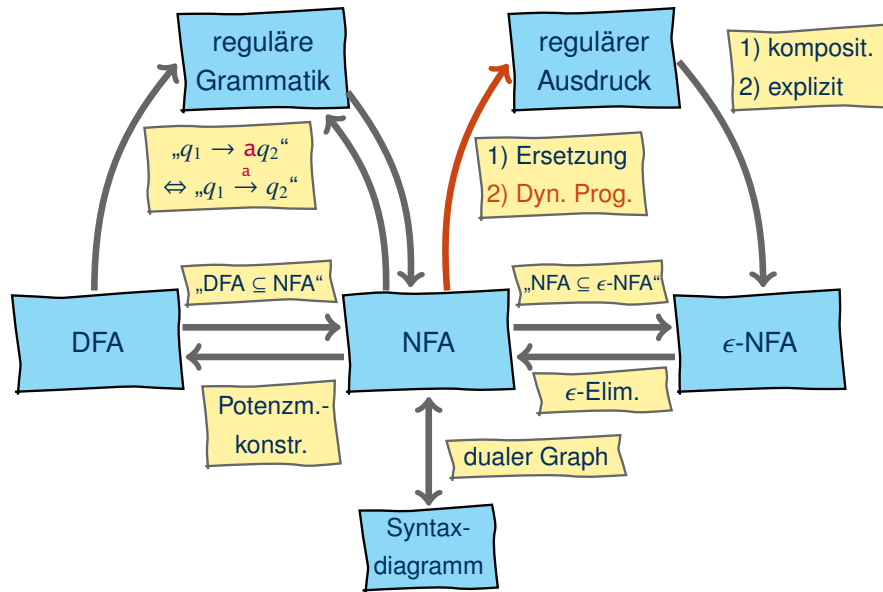
Induktionsschritt: sei $|w| = n$

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

Damit ist der Beweis abgeschlossen. □

Ermittlung regulärer Ausdrücke durch dynamische Programmierung

Darstellungen von Typ-3-Sprachen



Idee

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

Gesucht: regulärer Ausdruck α mit $L(\alpha) = L(M)$

Ansatz:

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

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

Dann gilt:

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

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

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

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

Gegeben M , Zahlen $i, j \in \{1, \dots, n\}$ und eine Zahl $k \in \{0, 1, \dots, n\}$ definieren wir die Sprache $L^k[i, j]$ als die Menge aller Wörter $w = a_1 \dots a_\ell$ für die gilt:

- es gibt einen Lauf $q_i \xrightarrow{a_1} p_1 \xrightarrow{a_2} \dots \xrightarrow{a_{\ell-1}} p_{\ell-1} \xrightarrow{a_\ell} q_j$, wobei
- für jeden Zwischenzustand p_i mit $i \in \{1, \dots, \ell - 1\}$ gilt $p_i \in \{q_1, \dots, q_k\}$

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

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

Der Fall $k = n$

Gegeben M , Zahlen $i, j \in \{1, \dots, n\}$ und eine Zahl $k \in \{0, 1, \dots, n\}$ definieren wir die Sprache $L^k[i, j]$ als die Menge aller Wörter $w = a_1 \dots a_\ell$ für die gilt:

- es gibt einen Lauf $q_i \xrightarrow{a_1} p_1 \xrightarrow{a_2} \dots \xrightarrow{a_{\ell-1}} p_{\ell-1} \xrightarrow{a_\ell} q_j$, wobei
- für jeden Zwischenzustand p_i mit $i \in \{1, \dots, \ell - 1\}$ gilt $p_i \in \{q_1, \dots, q_k\}$

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

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

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

Der Fall $k = 0$

Gegeben \mathcal{M} , Zahlen $i, j \in \{1, \dots, n\}$ und eine Zahl $k \in \{0, 1, \dots, n\}$ definieren wir die Sprache $\mathbf{L}^k[i, j]$ als die Menge aller Wörter $w = a_1 \cdots a_\ell$ für die gilt:

- es gibt einen Lauf $q_i \xrightarrow{a_1} p_1 \xrightarrow{a_2} \dots \xrightarrow{a_{\ell-1}} p_{\ell-1} \xrightarrow{a_\ell} q_j$, wobei
- für jeden Zwischenzustand p_i mit $i \in \{1, \dots, \ell - 1\}$ gilt $p_i \in \{q_1, \dots, q_k\}$

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

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

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

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

- es gibt einen Lauf $q_i \xrightarrow{a_1} p_1 \xrightarrow{a_2} \dots \xrightarrow{a_{\ell-1}} p_{\ell-1} \xrightarrow{a_\ell} q_j$, wobei
- für jedes p_i mit $i \in \{1, \dots, \ell - 1\}$ gilt $p_i \in \{q_1, \dots, q_k\}$

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

- (1) kein p_i ist q_{k+1} , d.h. $p_i \in \{q_1, \dots, q_k\}$
- (2) einige p_i sind q_{k+1} ; dann hat der Lauf die Form:

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

Teilläufe: $q_i \rightarrow q_{k+1}$ $(q_{k+1} \rightarrow q_{k+1})^*$ $q_{k+1} \rightarrow q_j$

Daher gilt:

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

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

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

Sei $\{a_1, \dots, a_m\} = \{a \in \Sigma \mid q_i \xrightarrow{a} q_j\}$ die Menge der Beschriftungen von direkten Übergängen von q_i zu q_j .

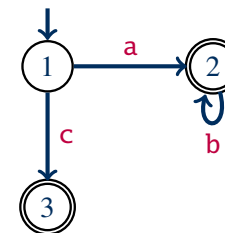
- Falls $i \neq j$, dann ist

$$\alpha^0[i, j] = a_1 \mid \dots \mid a_m$$

- Falls $i = j$, dann ist

$$\alpha^0[i, j] = a_1 \mid \dots \mid a_m \mid \epsilon$$

Beispiel: Dynamische Programmierung (1)



Fall $k = 0$:

$$\begin{array}{lll} \alpha^0[1, 1] = \epsilon & \alpha^0[1, 2] = a & \alpha^0[1, 3] = c \\ \alpha^0[2, 1] = \emptyset & \alpha^0[2, 2] = b \mid \epsilon & \alpha^0[2, 3] = \emptyset \\ \alpha^0[3, 1] = \emptyset & \alpha^0[3, 2] = \emptyset & \alpha^0[3, 3] = \epsilon \end{array}$$

Fall $k = 1$:

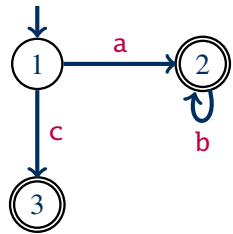
$$\begin{aligned} \alpha^1[1, 1] &= \underbrace{\alpha^0[1, 1]}_{\epsilon} \mid \underbrace{(\alpha^0[1, 1](\alpha^0[1, 1])^* \alpha^0[1, 1])}_{\epsilon \epsilon^* \epsilon} \equiv \epsilon \\ \alpha^1[1, 2] &= \underbrace{\alpha^0[1, 2]}_a \mid \underbrace{(\alpha^0[1, 1](\alpha^0[1, 1])^* \alpha^0[1, 2])}_{\epsilon \epsilon^* a} \equiv a \\ &\dots \end{aligned}$$

... syntaktische, aber keine semantische 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 $k = 1$:

$$\begin{aligned} \alpha^1[1, 1] &\equiv \epsilon & \alpha^1[1, 2] &\equiv a & \alpha^1[1, 3] &\equiv c \\ \alpha^1[2, 1] &\equiv \emptyset & \alpha^1[2, 2] &\equiv b \mid \epsilon & \alpha^1[2, 3] &\equiv \emptyset \\ \alpha^1[3, 1] &\equiv \emptyset & \alpha^1[3, 2] &\equiv \emptyset & \alpha^1[3, 3] &\equiv \epsilon \end{aligned}$$

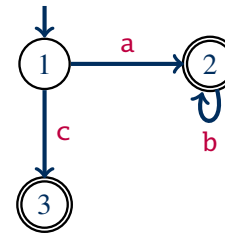
Fall $k = 2$:

$$\begin{aligned} \alpha^2[1, 1] &= \alpha^1[1, 1] \mid (\alpha^1[1, 2](\alpha^1[2, 2])^*\alpha^1[2, 1]) = \epsilon \mid (ab^*\emptyset) \equiv \epsilon \\ \alpha^2[1, 2] &= \alpha^1[1, 2] \mid (\alpha^1[1, 2](\alpha^1[2, 2])^*\alpha^1[2, 2]) = a \mid (a(b \mid \epsilon)^*(b \mid \epsilon)) \equiv ab^* \\ \alpha^2[1, 3] &= \alpha^1[1, 3] \mid (\alpha^1[1, 2](\alpha^1[2, 2])^*\alpha^1[2, 3]) = c \mid (ab^*\emptyset) \equiv c \end{aligned}$$

...

$$\begin{aligned} \alpha^2[2, 1] &\equiv \emptyset & \alpha^2[2, 2] &\equiv b^* & \alpha^2[2, 3] &\equiv \emptyset \\ \alpha^2[3, 1] &\equiv \emptyset & \alpha^2[3, 2] &\equiv \emptyset & \alpha^2[3, 3] &\equiv \epsilon \end{aligned}$$

Beispiel: Dynamische Programmierung (3)



Fall $k = 2$:

$$\begin{aligned} \alpha^2[1, 1] &\equiv \epsilon & \alpha^2[1, 2] &\equiv ab^* & \alpha^2[1, 3] &\equiv c \\ \alpha^2[2, 1] &\equiv \emptyset & \alpha^2[2, 2] &\equiv b^* & \alpha^2[2, 3] &\equiv \emptyset \\ \alpha^2[3, 1] &\equiv \emptyset & \alpha^2[3, 2] &\equiv \emptyset & \alpha^2[3, 3] &\equiv \epsilon \end{aligned}$$

Fall $k = 3$:

syntaktische, aber keine semantische 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:

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

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?