Automaten und Formale Sprachen

SoSe 2007 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

Automaten und Formale Sprachen — Gesamtübersicht

- Organisatorisches
- Einführung
- **Endliche Automaten und reguläre Sprachen**
- Kontextfreie Grammatiken und kontrextfreie Sprachen
- Chomsky-Hierarchie

Endliche Automaten und reguläre Sprachen

1. Deterministische endliche Automaten
2. Nichtdeterministische endliche Automaten
3. **Reguläre Ausdrücke**
4. Nichtreguläre Sprachen
5. Algorithmen mit / für endliche Automaten

Ein NEA-Generator-Kochrezept für Übungsaufgaben ?!

1. Man nehme: ein Alphabet Σ, ein paar Sprachen $L_0, \ldots, L_{k-1} \subseteq \Sigma^*$, zwei Menge $Q_0 \subseteq \mathbb{Z}_k$, $F \subseteq \mathbb{Z}_k$.

2. Nimm als Zustandsmenge $Q = \mathbb{Z}_k$.

3. Definiere als Übergangsrelation δ: $(q, a, r) \in \delta$ gdw. $\exists w \in L_q : wa \in L_r$.

REG $\subseteq$ DEA

Beweis: Betrachte $L \subseteq \Sigma^*$ definiert durch Monoid $(M, \circ, e)$, Menge $F \subseteq M$ und Morphismus $h : \Sigma^* \to M$.

Betrachte zu $m \in M$ das Urbild $h^{-1}(m) = \{w \in \Sigma^* \mid h(w) = m\} =: L_m$.

Das Kochrezept liefert für die L_m:

$(L_m, a, L_n) \in \delta$ gdw. $wa \in L_n$ für ein $w \in L_m$ gdw. $h(wa) = h(w)h(a) = mh(a) = n$.

Also ist der erwartete NEA sogar deterministisch.

Mit $q_0 = L_e$ und Endzustandsmenge $h^{-1}(F)$ gewinnen wir DEA A mit $L(A) = L$.

Beispiel: Mit $M = \mathbb{Z}_k$ erhalten wir gerade die beim Kochrezept diskutierten Automaten A_k.

DEA $\subseteq$ REG

Betrachte DEA $A = (Q, \Sigma, \delta, q_0, F)$.
Zu jedem festen $w \in \Sigma^*$ induziert A (eigentlich schon δ) eine Abbildung

$$\alpha_w : Q \to Q, p \mapsto q \text{ mit } (p, w) \vdash_A^* (q, \lambda).$$

Aus der ersten Vorlesung wissen wir: $M_A = \{\alpha_w \mid w \in \Sigma^*\}$ bilden zusammen mit der Komposition ein Monoid mit neutralem Element α_λ;
genauer legen wir hier fest: $(\alpha_w \circ' \alpha_u)(p) = \alpha_u(\alpha_w(p))$.
Damit wird $w \mapsto \alpha_w$ zu einem Monoidmorphismus h.
Lege ferner fest: $F' = \{\alpha_w \mid w \in L(A)\}$.
Das *Transformationsmonoid* $(M_A, \circ', \alpha_\lambda)$ ist ein Teilmonoid von $(Q^Q, \circ', id_Q)$.
Damit spezifieren $(M_A, \circ', \alpha_\lambda)$ zusammen mit h und F' die Sprache $L(A)$.

Satz: **NEA** $= \lambda -$ **NEA** $=$ **DEA** $=$ **REG**

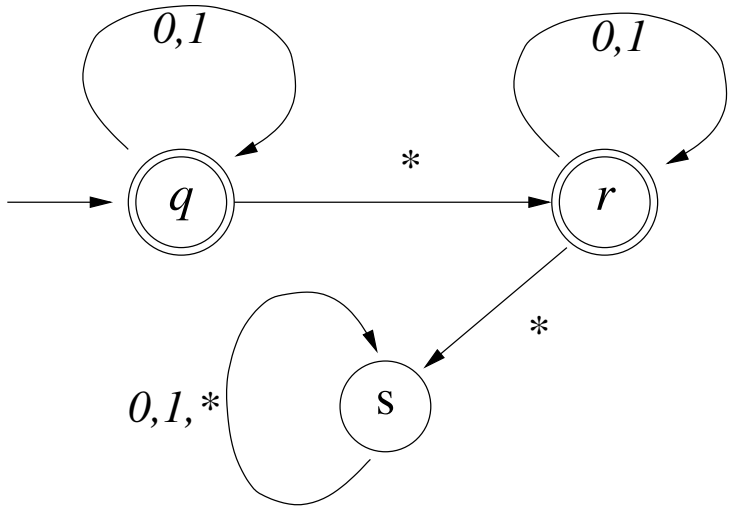
Ein großes Transformationsmonoid

Betrachte $\Sigma = \{a, b, c\}$, das Zustandsalphabet $\mathbb{Z}_n$ und die folgenden Abbildungen:

$$\begin{aligned}
\alpha_a &: \ 0 \mapsto 1, 1 \mapsto 0, k \mapsto k \text{ für } k \neq 0, 1 \\
\alpha_b &: \ 0 \mapsto 1, 1 \mapsto 2, \ldots, (n-2) \mapsto (n-1), (n-1) \mapsto 0 \\
\alpha_c &: \ (n-1) \mapsto 0, k \mapsto k \text{ für } k \neq n-1
\end{aligned}$$

Dies definiert zum einen die Überführungsfunktion eines DEA mit n Zuständen und Eingabealphabet Σ, zum anderen hat das zugehörige Transformationsmonoid n^n Elemente.

Ein Beispiel (lazy evaluation!)



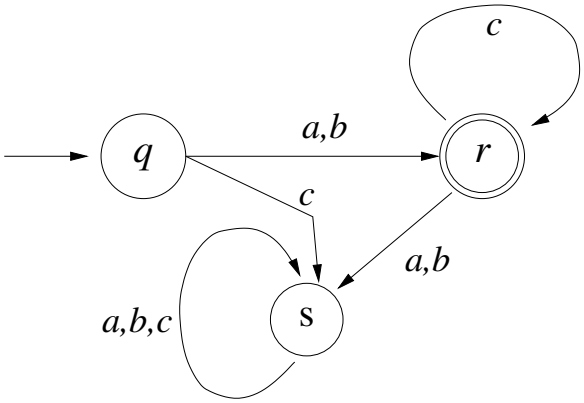
α_λ	q	r	s
$= \alpha_0$			
$= \alpha_1$			
	q	r	s

α_*	q	r	s
	r	s	s

α_{**}	q	r	s
	s	s	s

$(M_A, \circ')$	α_λ	α_*	α_{**}
α_λ	α_λ	α_*	α_{**}
α_*	α_*	α_{**}	α_{**}
α_{**}	α_{**}	α_{**}	α_{**}

Noch ein Beispiel (nicht-kommutativ!)



α_λ	q	r	s
	q	r	s

$\alpha_a = \alpha_b$	q	r	s
	r	s	s

$\alpha_c = \alpha_{cc}$	q	r	s
	s	r	s

$\alpha_x, x \in \{a,b\}^2$	q	r	s
	s	s	s

$\alpha_{ac} = \alpha_{bc} = \alpha_a \qquad \alpha_{ca} = \alpha_{cb} = \alpha_{aa}$

$(M_A, \circ')$	α_λ	α_a	α_c	α_{aa}
α_λ	α_λ	α_a	α_c	α_{aa}
α_a	α_a	α_{aa}	α_a	α_{aa}
α_c	α_c	α_{aa}	α_c	α_{aa}
α_{aa}	α_{aa}	α_{aa}	α_{aa}	α_{aa}

Eine algebraischere Betrachtungsweise von Sprachoperationen

Erinnerung: Eine Sprache $L \subseteq \Sigma^*$ gehört zu **REG** gdw. es ein endliches Monoid $(M, \circ, e)$, einen Monoidmorphismus $h : (\Sigma^*, \cdot, \lambda) \to (M, \circ, e)$ sowie eine endliche Menge $F \subseteq M$ gibt mit

$$L = \{w \in \Sigma^* \mid h(w) \in F\}.$$

Satz: **REG** ist gegen Komplementbildung abgeschlossen.

Beweis: Sei $L \subseteq \Sigma^*$ durch ein endliches Monoid $(M, \circ, e)$, einen Monoidmorphismus $h : (\Sigma^*, \cdot, \lambda) \to (M, \circ, e)$ sowie eine endliche Menge $F \subseteq M$ spezifiziert. Dann spezifizieren $(M, \circ, e)$, h und $M \setminus F$ gemeinsam $\Sigma^* \setminus L$.

Monoide aus Monoiden I

Es seien $(M, \circ, e)$ und $(N, \square, 1)$ Monoide.
Dann kann man die Menge $M \times N$ zu einem Monoid machen durch *komponentenweises Anwenden* der Operationen; definiere daher:

$$(m, n)\,[\circ, \square](m', n') := (m \circ m', n \square n').$$

Satz: $(M \times N, [\circ, \square], (e, 1))$ ist ein Monoid, das *Produktmonoid*.

Satz: Sind $h_M : (X, \Delta, I) \to (M, \circ, e)$ und $h_N : (X, \Delta, I) \to (N, \square, 1)$ Monoidmorphismen, so auch der *Produktmorphismus*
$h_M \times h_N : X \to M \times N, x \mapsto (h_M(x), h_N(x))$.

Beide Sätze lassen sich auf Produkte endlich vieler Monoide bzw. Morphismen verallgemeinern.

Mengenoperationen als Sprachoperationen allgemein

Satz: Ist f eine k-stellige Mengenoperation, so ist **REG** gegen f abgeschlossen.

Beweis: Betrachte k reguläre Sprachen L_i, spezifiziert durch endliche Monoide $(M_i, \circ_i, e_i)$, Morphismen h_i und endliche Mengen $F_i \subseteq M_i$. Dann spezifizieren das Produktmonoid $M_1 \times \cdots \times M_k$, der Morphismus $h_1 \times \cdots \times h_k$ und die endliche Menge $f(F'_1, \ldots, F'_k)$ die Sprache $f(L_1, \ldots, L_k)$; hierbei sei $F'_i = \{(x_1, \ldots, x_k) \mid \forall 1 \leq j \leq k : x_j \in M_j \wedge x_i \in F_i\}$. Mithin ist $f(L_1, \ldots, L_k)$ regulär.

Ist speziell f der Durchschnitt, so gilt: $F'_1 \cap F'_2 = F_1 \times F_2$.

Ein Beispiel

Betrachte als endliche Monoide $\mathcal{M}_1 = (\mathbb{Z}_2, +, 0)$ und $\mathcal{M}_2 = (\mathbb{Z}_3, +, 0)$. Dann übersetzt der Morphismus ϕ aus $\mathcal{M}_1 \times \mathcal{M}_2$ in das Monoid $\mathcal{M}_3 = (\mathbb{Z}_6, +, 0)$ gemäß: $(0,0) \mapsto 0$, $(0,1) \mapsto 4$, $(0,2) \mapsto 2$, $(1,0) \mapsto 3$, $(1,1) \mapsto 1$, $(1,2) \mapsto 5$.

$\mathcal{M}_1$, ℓ_2 und $\{0\}$ beschreiben die Wörter L_1 gerader Länge.
$\mathcal{M}_2$, ℓ_3 und $\{1,2\}$ beschreiben die Wörter L_2, deren Länge beim Teilen durch drei nicht den Rest 0 lässt.
$\mathcal{M}_1 \times \mathcal{M}_2$, $\ell_2 \times \ell_3$ und

$$F = \{(0,0),(0,1),(0,2)\} \cap \{(0,1),(1,1),(0,2),(1,2)\} = \{(0,1),(0,2)\} = \{0\} \times \{1,2\}$$

beschreiben die Wörter, deren Länge gerade ist und beim Teilen durch drei den Rest 1 oder 2 lässt.
Gleichwertig lässt sich $L_1 \cap L_2$ durch das Monoid $\mathcal{M}_3$, ℓ_6 sowie $\phi(F) = \{2,4\}$ darstellen.

Monoide aus Monoiden II

Ist $(M, \circ, e)$ ein Monoid, so kann der Menge 2^M durch das *Komplexprodukt* zu einem Monoid gemacht werden. Dazu definieren wir:

$$A \circ B := \{a \circ b \mid a \in A \wedge b \in B\}$$

Das zugehörige neutrale Element ist $\{e\}$.

Beispiel: $(\Sigma^*, \cdot, \lambda)$ ist ein Monoid, und so kann man auch $\cdot$ als Sprachoperation auffassen.

Satz: **REG** ist gegen Konkatenation abgeschlossen.

Beweis: Es seien $L_1, L_2 \in$ **REG**.
Wir können davon ausgehen, dass ein λ-NEA

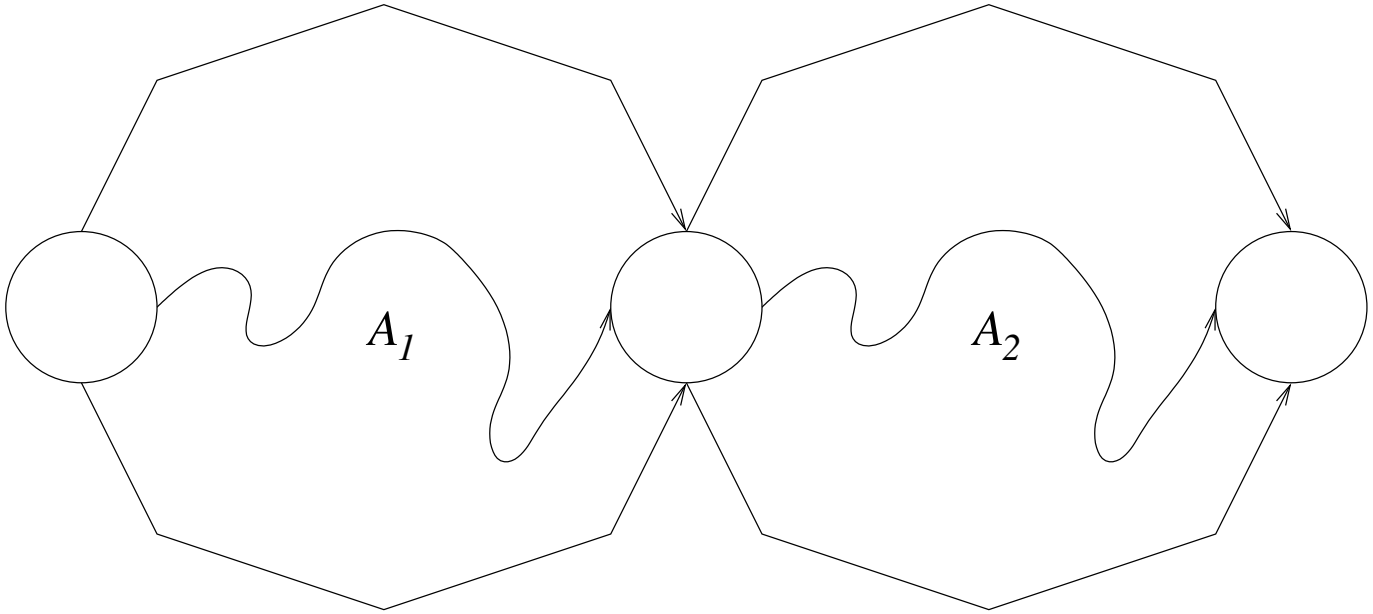
$$A_i = (Q_i, \Sigma, \delta_i, Q_{0,i}, F_i)$$

L_i akzeptiert, der nur einen Anfangs- und einen Endzustand besitzt; der Anfangszustand hat nur ausgehende Kanten und der Endzustand nur eingehende.

Wir gehen ferner davon aus, dass $Q_1 \cap Q_2 = F_1 = Q_{0,2}$ gilt.

Setze $Q = Q_1 \cup Q_2$ und $\delta = \delta_1 \cup \delta_2$. $A = (Q, \Sigma, \delta, Q_{0,1}, F_2)$ akzeptiert $L_1 \cdot L_2$.

REG ist gegen Konkatenation abgeschlossen: Skizze



Potenzen in Monoiden: Der Weg zum Kleene-Stern.

Ist $(M, \circ, e)$ ein Monoid, so können wir rekursiv die n-te *Potenz* eines Elementes $x \in M$ rekursiv festlegen durch:
$x^0 = e$
$x^{n+1} = x^n \circ x$ für $n \in \mathbb{N}$.

Wie wir gesehen haben, bildet auch $(2^M, \circ, \{e\})$ ein Monoid.
Somit ist auch A^n für $A \subseteq M$ und $n \in \mathbb{N}$ definiert.
(Leider kollidiert diese Schreibweise mit dem von dem kartesischen Mengenprodukt induzierten Potenz, aber nicht arg...)

Dann kann man $A^+ = \bigcup_{n \geq 1} A^n$ definieren und $A^* = \bigcup_{n \geq 0} A^n$.

Somit ist auch L^+ und L^* (*Kleene-Stern*) für $L \subseteq \Sigma^*$ festgelegt.

Satz: **REG** ist gegen Kleene-Stern abgeschlossen.

Beweis: Sei $L \in$ **REG** akzeptiert durch einen λ-NEA A, der nur einen Anfangs- und einen Endzustand q_0 und q_f besitzt; der Anfangszustand hat nur ausgehende Kanten und der Endzustand nur eingehende.
Durch Verschmelzen von q_0 und q_f als neuer Anfangs- und Endzustand erhalten wir einen NEA A' mit $L(A') = L^*$.

Reguläre Ausdrücke (ähnlich `grep`)

Definition durch *strukturelle Induktion*:

- $\emptyset$ und $\mathtt{a}$ sind RA für jedes $a \in \Sigma$.
- Ist R ein RA, so auch $(R)*$.
- Sind R_1 und R_2 RAs, so auch R_1R_2 und $(R_1 \cup R_2)$.

Beispiel: $((\mathtt{b} \cup \mathtt{a}))*\mathtt{aaa}(\mathtt{bb})*$ ist ein RA.

Klammern können weggelassen werden: $*$ bindet stärker als Konkatenation, und jenes wieder stärker als Vereinigung.

Die **durch einen RA beschriebene Sprache** ist ebenfalls rekursiv gegeben:

- $L(\emptyset) = \emptyset$; $L(a) = \{a\}$.
- Ist R aus RE, setze $L((R)*) = (L(R))^*$.
- Sind R_1 und R_2 RA, setze
  $L(R_1R_2) = L(R_1)L(R_2)$ und $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$.

Beispiel: $L((b \cup a)*) = \{a, b\}^*$

Beispiele

(1) Beschreibe die Sprache zum Ausdruck $(\mathtt{ab}*)\mathtt{a}$ in Mengennotation.

$$\begin{aligned} L((\mathtt{ab}*)\mathtt{a}) &= L((\mathtt{ab}*)) \cdot L(\mathtt{a}) \\ &= L(\mathtt{a}) \cdot L(\mathtt{b}*) \cdot \{\mathtt{a}\} \\ &= \{\mathtt{a}\} \cdot (L(\mathtt{b}))^* \cdot \{\mathtt{a}\} \\ &= \{\mathtt{a}\} \cdot \{\mathtt{b}\}^* \cdot \{\mathtt{a}\} \\ &= \{\mathtt{a}\mathtt{b}^n\mathtt{a} \mid n \in \mathbb{N}\} \end{aligned}$$

(2) Beschreibe die Sprache zum Ausdruck $(\mathtt{a} * \mathtt{b})*$ in Worten.

Die Menge aller Wörter über $\{\mathtt{a}, \mathtt{b}\}$, die nicht mit $\mathtt{a}$ enden.

Satz: Jede RA-Sprache ist regulär.

Beweis: (durch strukturelle Induktion)

- Endliche Sprachen sind regulär.
- Reguläre Sprachen sind gegen Kleene-Stern abgeschlossen.
- Reguläre Sprachen sind gegen Vereinigung und Konkatenation abgeschlossen.

Beispiel: $(\mathtt{a} \cup \mathtt{ab})*$ (siehe Tafel)

Satz: Jede reguläre Sprache ist durch einen RA beschreibbar.

Beweis: Betrachte DEA $A = (Q, \Sigma, \delta, q_0, F)$ mit $Q = \{1, \ldots, n\}$ und $q_0 = 1$.

$R[i, j, k]$ RA für die Sprache, die von A akzeptiert wird, indem (1) A in Zustand i anfängt, (2) in Zustand j aufhört, und (3) zwischendurch nur Zustände aus $\{1, \ldots, k\}$ erreicht.

Hinweis: Warshall/Floyd

Offenbar gilt: $L(A) = \bigcup_{j \in F} L(R[1, j, n]) = L(\bigcup_{j \in F} R[1, j, n])$.

$R[i, j, 0] = x_1 \cup \cdots \cup x_\ell$, wobei x_l alle Beschriftungen von Kanten zwischen i und j auflisten (zusätzlich $\emptyset*$ falls $i = j$)

Für $k > 0$ setze rekursiv $R[i, j, k] = R[i, j, k-1] \cup R[i, k, k-1] R[k, k, k-1]* R[k, j, k-1]$.

Daraus ergibt sich ein Algorithmus durch **dynamisches Programmieren**.

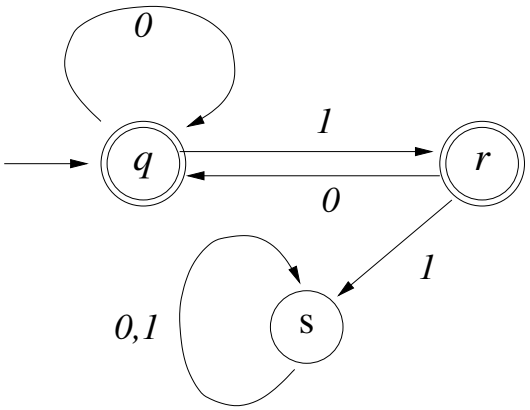
$R[1..n, 1..n, 0..n]$ ist 3-dimensionales Array, dessen Einträge reguläre Ausdrücke sind.

Für $i := 1$ bis n tue:
    Für $j := 1$ bis n tue:
        $R[i, j, 0] := \bigcup_{a \in \Sigma} a$
        Falls $i = j$, so setze $R[i, j, 0] := R[i, j, 0] \cup \emptyset*$.

Für $k := 1$ bis n tue:
    Für $i := 1$ bis n tue:
        Für $j := 1$ bis n tue:
            $R[i, j, k] := R[i, j, k-1] \cup R[i, k, k-1] R[k, k, k-1]* R[k, j, k-1]$.

Damit klar: kubische Komplexität, i.Z.: $O(n^3)$.

Ein Beispiel



$R[i,j,0]$	q	r	s
q	$0 \cup \emptyset*$	1	$\emptyset$
r	0	$\emptyset*$	1
s	$\emptyset$	$\emptyset$	$0 \cup 1 \cup \emptyset*$

Ein Beispiel (Forts.)

$R[i,j,0]$	q	r	s
q	$0 \cup \emptyset*$	1	$\emptyset$
r	0	$\emptyset*$	1
s	$\emptyset$	$\emptyset$	$0 \cup 1 \cup \emptyset*$

$R[i,j,\{q\}]$	q	r
q	$0 \cup \emptyset * \cup((0 \cup \emptyset*)(0 \cup \emptyset*) * (0 \cup \emptyset*)) = 0*$	$1 \cup ((0 \cup \emptyset*)(0 \cup \emptyset*) * 1) = 0 * 1$
r	$0 \cup (0(0 \cup \emptyset*) * (0 \cup \emptyset*)) = 00*$	$\emptyset * \cup(0(0 \cup \emptyset*) * 1) = 00 * 1$

$R[i,j,\{q,r\}]$	q	r
q	$0 * \cup(0 * 1(00 * 1) * 00*)$	$0 * 1 \cup (0 * 1(00 * 1) * 00 * 1) = 0 * 1(00 * 1)*$
r	$\ldots$	$\ldots$

$$L(A) = L(0 * \cup(0 * 1(00 * 1) * 0*)).$$

Hinweis: Explosion EA / RA in beiden Richtungen !

Ein alternatives Verfahren (siehe Kinber/Smith)

arbeitet direkt auf dem evtl. nichtdeterministischen Automatengraphen.
Wichtige Konventionen:
Zustandsmenge $Q = \{1, \ldots, n\}$ mit
1: Anfangszustand und
n (einziger) Endzustand.

Hilfsroutinen:

- `mergearcs`(i, j): Sind $\ell^1_{i,j}$, $\ldots \ell^m_{i,j}$ die Beschriftungen sämtlicher Kanten von i nach j im Automatengraphen, so ersetze diese m Kanten durch eine mit $(\ell_{i,j} \cup \ldots \cup \ell^m_{i,j})$ beschriftete.

- `shortcut`$(i, j; k)$: Falls es nur genau eine Kante von i nach k und genau eine Kante von k nach j gibt, tue:

  1. Gibt es genau eine Kante von k nach k mit Beschriftung $\ell_{k,k}$, so tue:
     Ersetze einzige Kante von i nach k mit Beschriftung $\ell_{i,k}$ und einzige Kante von k nach j mit Beschriftung $\ell_{k,j}$ durch neue Kante von i nach j mit Beschriftung $\ell_{i,k}(\ell_{k,k}) * \ell_{k,j}$.
  2. Andernfalls: Ersetze einzige Kante von i nach k mit Beschriftung $\ell_{i,k}$ und einzige Kante von k nach j mit Beschriftung $\ell_{k,j}$ durch neue Kante von i nach j mit Beschriftung $\ell_{i,k}\ell_{k,j}$.

- `remove`(k): Lösche Knoten k und alle mit k inzidenten Kanten.

Der zweite Algorithmus zur Erzeugung äquivalenter RAs

Für $i := 1$ bis n tue:
    Für $j := 1$ bis n tue:
        $\texttt{mergearcs}(i, j)$
Für $k := 2$ bis $n - 1$ tue:
    Für $i := 1$ bis n tue:
        Für $j := 1$ bis n tue:
            $\texttt{shortcut}(i, j; k)$;
            $\texttt{mergearcs}(i, j)$;
            $\texttt{remove}(k)$.

Der gewünschte reguläre Ausdruck findet sich am Schluss als Kantenbeschriftung von der (einzigen) Kante von Knoten 1 nach Knoten n. Sollte keine solche Kante existieren, so ist die Sprache leer und kann durch $\emptyset$ beschrieben werden.

Vorheriges Beispiel an der Tafel !