Automaten und Formale Sprachen

SoSe 2017 in Trier

Henning Fernau
Universität Trier
fernau@uni-trier.de

4. Mai 2017

Automaten und Formale Sprachen Gesamtübersicht

- Organisatorisches
- Einführung
- **Endliche Automaten und reguläre Sprachen**
- Kontextfreie Grammatiken und kontextfreie 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

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 $\Sigma^* \setminus L$. (Beweiseinzelheiten zur Übung.) $\square$

Monoide aus Monoiden I

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

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

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

Satz: Sind $h_M : (X, \Delta, I) \to (M, \circ, e)$ und $h_N : (X, \Delta, I) \to (N, \Box, 1)$ 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.
In dieser Form werden wir sie im Folgenden benutzen.
Wenn wir von k-stelligen Mengenoperationen reden, so meinen wir solche, die durch Mengenausdrücke mit Vereinigung und Komplementbildung ausgedrückt werden können.

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. $\square$

Ist speziell f der Durchschnitt, so gilt: $F_1' \cap F_2' = F_1 \times M_2 \cap M_1 \times F_2 = F_1 \times F_2$.
Schauen wir uns für diesen Fall beweistechnische Einzelheiten an:

$L_i = \{w \in \Sigma^* \mid h_i(w) \in F_i\}$ für $i = 1, 2$ laut Def.
Für die konstruierte Sprache ist:
$L_\cap = \{w \in \Sigma^* \mid (h_1 \times h_2)(w) \in (F_1 \times M_2 \cap M_1 \times F_2)\}$.
Zu zeigen bleibt: $L_\cap = L_1 \cap L_2$.

$$\begin{aligned} w \in L_1 \cap L_2 &\iff w \in L_1 \wedge w \in L_2 \iff h_1(w) \in F_1 \wedge h_2(w) \in F_2 \\ &\iff (h_1(w), h_2(w)) \in F_1 \times F_2 \iff (h_1 \times h_2)(w) \in (F_1 \times M_2 \cap M_1 \times F_2). \end{aligned}$$

Alternative Sicht: Operationen auf Sprachen werden zu Operationen auf Zustandsmengen eines deterministischen endlichen Automaten.

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 (Wdh.)

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

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

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. (Warum gibt es diese Normalform?)

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 $\delta = \delta_1 \cup \delta_2$.

Beh.: $A = (Q, \Sigma, \delta, Q_{0,1}, F_2)$ akzeptiert $L_1 \cdot L_2$.

$L_1 \cdot L_2 \subseteq L(A)$ ist durch die Konstruktion einzusehen.

$L(A) \subseteq L_1 \cdot L_2$ ist die schwierigere Richtung.

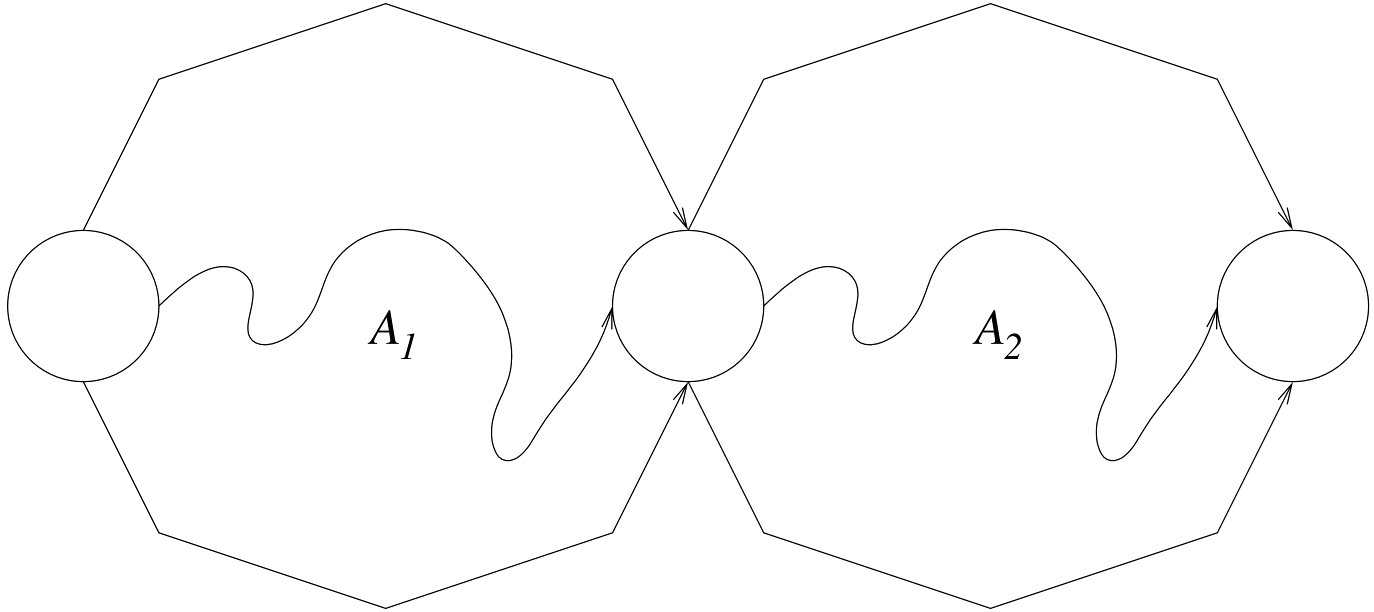
Wichtige Eigenschaften von A:

— Jeder Pfad von $Q_{0,1}$ nach F_2 führt durch $F_1 = Q_{0,2}$.

— Es gibt keinen Pfad von Q_2 nach $Q_1 \setminus F_1$.

(Einzelheiten zur Übung. Das nachfolgende Bild soll die Konstruktion erklären.) □

REG ist gegen Konkatenation abgeschlossen: Skizze



Potenzen in Monoiden: Der Weg zum Kleene-Stern.

Ist $(M, \circ, e)$ ein Monoid, so können wir induktiv die n-te *Potenz* eines Elementes $x \in M$ rekursiv festlegen durch:
$x^0 = e$ sowie $x^{n+1} = x^n \circ x$ 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: L^+ (L^*) ist die (das) durch L bezüglich der Konkatenation erzeugte Halbgruppe (Monoid).

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 habe nur ausgehende Kanten und der Endzustand nur eingehende.

Durch Verschmelzen von q_0 und q_f zu neuem Anfangs- und Endzustand q_{0f} erhalten wir einen NEA A' mit $L(A') = L^*$.

Betrachte $w \in L(A')$. Es gibt eine Folge von Zuständen $p_1, \ldots, p_n$ ($n \geq \ell(w)$ und $p_1 = p_n = q_{0f}$), sodass für geeignete Suffixe $w_1 = w, \ldots, w_n = \lambda$ von w gilt: $(p_i, w_i) \vdash_{A'} (p_{i+1}, w_{i+1})$ sowie $w_i = w_{i+1}$ oder $w_i = aw_{i+1}$ für ein Zeichen a, für $i = 1, \ldots, n-1$.

Definiere $J(w) = \{j \mid p_j = q_{0f}\}$. Wir zeigen die Beh. durch Induktion über $|J(w)| \geq 1$.

Für $|J(w)| = 1$ haben wir $w = \lambda$ vorzuliegen, also gilt sowieso $w \in L^*$.

Ist $j > 1$ der erste Index mit $p_j = q_{0f}$, so gilt für $w = u_j w_j$: $u_j \in L(A) = L$. Ferner gibt es für $w_j \in L(A')$ einen durch $p_j, \ldots, p_n$ beschriebenen Akzeptierungsweg mit $|J(w_j)| < |J(w)|$, sodass wir hier die IV anwenden können. Also folgt $w \in L \cdot L^* \subseteq L^*$.

Die Inklusion $L^* \subseteq L(A')$ sieht man leichter anhand der Konstruktion ein. □

Reguläre Ausdrücke (ähnlich `grep`) über festem aber bel. Alphabet Σ:

Definition durch *strukturelle Induktion*:

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

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

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

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

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

Ein RA über Σ beschreibt also eine Sprache über Σ.

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

Beispiele

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

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

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

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

Satz: Jede RA-Sprache ist regulär.

<u>Beweis:</u> (durch strukturelle Induktion)

- Endliche Sprachen sind regulär. (Dies liefert den Induktionsanfang.)
- Reguläre Sprachen sind gegen Kleene-Stern abgeschlossen.
- Reguläre Sprachen sind gegen Vereinigung und Konkatenation abgeschlossen.

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

Zusammenhang zwischen struktureller und vollständiger Induktion
ergeben sich zumeist durch Einführung geeigneter "Zählvariablen".
In unserem Fall sei zu RA R die Anzahl der Operationssymbole in R notiert als: $\#_{\mathrm{op}}(R)$.
Dies lässt sich auch wiederum strukturell induktiv definieren:

- $\#_{\mathrm{op}}(\emptyset) = 0$ und $\#_{\mathrm{op}}(a) = 0$ für jedes $a \in \Sigma$.

- Ist R ein RA, so gilt: $\#_{\mathrm{op}}((R)*) = \#_{\mathrm{op}}(R) + 1$.

- Sind R_1 und R_2 RAs, so gilt:
  $\#_{\mathrm{op}}(R_1R_2) = \#_{\mathrm{op}}((R_1 \cup R_2)) = \#_{\mathrm{op}}(R_1) + \#_{\mathrm{op}}(R_2) + 1$.

Zusammenhang zwischen struktureller und vollständiger Induktion

Die Beweisskizze lässt sich als Beweis durch vollständige Induktion nach $\#_{op}(R)$ begreifen.

IA: Für $\#_{op}(R) = 0$ gilt: $R = \emptyset$ oder $R = a$ für ein $a \in \Sigma$.
Die entsprechenden Sprachen $\emptyset$ bzw. $\{a\}$ sind regulär.

IV: Jeder RA mit höchstens n Operationssymbolen beschreibt eine reguläre Sprache.
Betrachte einen RA R mit $n + 1$ Operationssymbolen.
Hierfür sind drei Fälle möglich:
(a) $R = (R_1)*$; (b) $R = R_1R_2$; (c) $R = (R_1 \cup R_2)$.
In jedem Fall gilt: $\#_{op}(R_1) \leq n$ sowie $\#_{op}(R_2) \leq n$ (falls sinnvoll).
Also sind nach IV $L(R_1)$ und $L(R_2)$ regulär.
Da die regulären Sprachen gegen Kleene Stern, Konkatenation und Vereinigung abgeschlossen sind, ist (in jedem Fall) auch $L(R)$ regulär. $\square$

Satz: Jede reguläre Sprache ist durch einen RA beschreibbar.

<u>Beweis:</u> Betrachte DEA $A = (Q, \Sigma, \delta, q_0, F)$ mit $Q = \{1, \dots, n\}$ 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 $\{1, \dots, k\}$ erreicht.

<u>Hinweis:</u> Warshall/Floyd

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

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

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

Das liefert sofort einen rekursiven (schlechten) Algorithmus.

Alternativ: besserer Algorithmus durch **dynamisches Programmieren**.

$R[1..n, 1..n, 0..n]$ ist 3-dim. Array mit regulären Ausdrücken als Einträgen.

Für $i := 1$ bis n tue:

  Für $j := 1$ bis n tue:

    $R[i, j, 0] := \bigcup_{a \in \Sigma : \delta(i,a)=j} a$

    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:

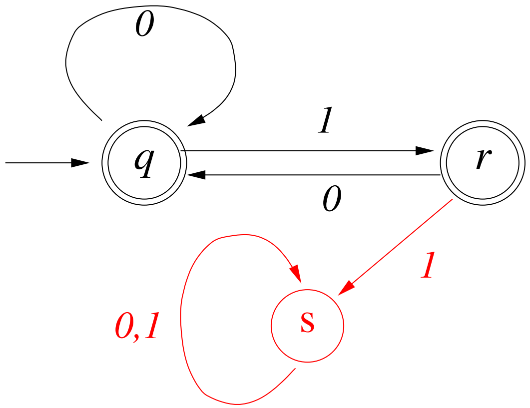
    Für $j := 1$ bis n tue:

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

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

Ein Beispiel

(roter Zustand kann weggelassen werden, da er nicht zur Sprache beiträgt)



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

Ein Beispiel (Forts.)

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

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

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

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 (ohne eingehende Kanten) und
n (einziger) Endzustand (ohne ausgehende Kanten).
Kantenbeschriftungen dürfen hierbei reguläre Ausdrücke sein.
(Tatsächlich kann man auch derartige Automaten betrachten.)

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 $(\ell^1_{i,j} \cup \ldots \cup \ell^m_{i,j})$ 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 !