Reguläre Ausdrücke

Dr. Eva Richter

November 13, 2007

Reguläre Ausdrücke

- in der Arithmetik werden $\times$ und $+$ zur Bildung von Termen verwendet, z.B.

$$(5 + 3) \times 4$$

- man verwendet reguläre Operationen zur Bildung von Termen, die Sprachen beschreiben: reguläre Ausdrücke

$$(0 + 1)0^*$$

- Wert von $(5 + 3) \times 4$ ist 32,
- der Wert eines reguären Ausdrucks ist eine Sprache,
- z.B. Wert von $(0 + 1)0^*$ ist die Menge der Wörter, die mit 0 oder 1 beginnen und mit beliebig vielen Nullen enden

Wert von $(0 + 1)0^*$

- der Ausdruck wird in seine einzelenen Bestandteile zerlegt
- die Symbole 0 und 1 repräsentieren $\{0\}$ und $\{1\}$,
- $(0 + 1)$ steht für $\{0\} \cup \{1\}$,
- 0^* steht für $\{0\}^*$, sein Wert ist die Sprache, die alle Zeichenketten enthält, die aus lauter Nullen bestehen
- wie $\times$ in der Algebra ist das Zeichen $\circ$ für Verkettung (Konkatenation) oft implizit in regulären Ausdrücken: $(0 + 1)0^*$ ist Abkürzung für $(0 + 1) \circ 0^*$
- Verkettung fügt die Teile eines Ausdruckes aneinander, um den Wert des Gesamtausdruckes zu erhalten

Anwendungen

- in verschiedenen Informatik-Anwendungen
- in Texten zur Suche nach Zeichenketten, die bestimmten Mustern genügen
- solche Muster werden mit regulären Ausdrücken beschrieben
- AWK und GREP in UNIX
- moderne Programmiersprachen wie PERL und Texteditoren erlauben die Beschreibung von Mustern unter Verwendung regulärer Ausdrücke

Formale Definition (Syntax)

Ein Term R ist ein regulärer Ausdruck genau dann, wenn

1. $R = a$ für ein Zeichen a aus dem Alphabet Σ,
2. $R = \varepsilon$,
3. $R = \emptyset$,
4. $R = R_1 + R_2$, wobei R_1 und R_2 reguläre Ausdrücke sind, oder
5. $R = R_1 \circ R_2$, wobei R_1 und R_2 reguläre Ausdrücke sind, oder
6. $R = (R_1)^*$, wobei R_1 ein regulärer Ausdrucke ist.

- Die Definition ist nicht selbstreferentiell, da R_1 und R_2 stets kleiner als R sind.
- Eine Definition dieses Typs heißt induktive Definition.

"Präferenzregeln": * vor $\circ$ und $+$.

Formale Definition (Semantik)

Die Bedeutung bzw. der Wert der regulären Ausdrücke ist:

1. a steht für die einelementige Sprache $\{a\}$
2. ε steht für die einelementige Sprache $\{\varepsilon\}$
3. $\emptyset$ steht für die leere Sprache
4. $R_1 + R_2$ steht für die Vereinigung der Sprachen von R_1 und R_2,
5. $R_1 \circ R_2$, steht für die Verkettung der Sprachen von R_1 und R_2,
6. $(R_1)^*$ steht für die Sprache, deren Wörter Zeichenketten von Wörtern aus R_1 sind.

Beachte den Unterschied zwischen $\emptyset$ und ε!

Beispiele

Sei $\Sigma = \{a, b\}$ dann steht

1. a^*ba^* für $\{w \mid$ w enthält genau ein $b\}$
2. $\Sigma^*a\Sigma^*$ für $\{w \mid$ w enthält mindestens ein $a\}$
3. $(a + \varepsilon)b^*$ für $ab^* + b^*$
4. a^+ für aa^*
5. $a^*\emptyset$ für $\emptyset$
6. $\emptyset^* = \{\varepsilon\}$, die *-Operation fügt eine beliebige Zahl von Zeichen der Sprache aneinander. Wenn die Sprache leer ist, können 0 Zeichen aneinandergefügt werden– was nur das leere Zeichen ergibt.

Schreibweise: $L(R)$ bezeichnet die Sprache, die R repräsentiert.

Äquivalenz regulärer Ausdrücke

Zwei reguläre Ausdrücke R_1, R_2 heißen äquivalent (geschrieben $R_1 \cong R_2$), wenn sie dieselbe Sprache beschreiben. $R_1 \cong R_2$ genau dann, wenn $L(R_1) = L(R_2)$

$R + \emptyset$	$\cong$	R	Vereinigung mit der leeren Sprache führt bei keiner Sprache zu einer Erweiterung
$R \circ \varepsilon$	$\cong$	R	Anhängen des leeren Strings an Wörter ergibt keine neuen Wörter

Umformungsregeln

- reguläre Ausdrücke können äquivalent umgeformt werden
- $+$ ist kommutativ: $R_1 + R_2 \cong R_2 + R_1$,
  assoziativ: $(R_1 + R_2) + R_3 \cong R_1 + (R_2 + R_3)$ und
  idempotent: $R + R \cong R$
- $\circ$ ist assoziativ: $(R_1 \circ R_2) \circ R_3 \cong R_1 \circ (R_2 \circ R_3)$
- $\circ$ ist distributiv über $+$:
  $(R_1 + R_2) \circ R_3 \cong (R_1 \circ R_3) + (R_2 \circ R_3)$ und
  $R_1 \circ (R_2 + R_3) \cong (R_1 \circ R_2) + (R_1 \circ R_3)$

Verwendung regulärer Ausdrücke

- für das Entwerfen von Compilern für Programmiersprachen
- elementare Objekte (Token) wie z.B. Variablennamen oder Konstanten, Schlüsselwörter können mit regulären Ausdrücken beschrieben werden
- ist die Syntax des Tokens festgelegt, kann ein automatische System den Lexer (lexical analyzer) generieren, der die Token identifiziert.

Beispiel: numerische Konstante, die ein Vorzeichen und/oder ein Komma enthalten kann, gehört zur Sprache $L(R)$ mit

$$R = \{+, -, \varepsilon\}(DD^* + DD^*.D^* + D^*.DD^*)$$

mit $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

Äquivalenz zu endlichen Automaten

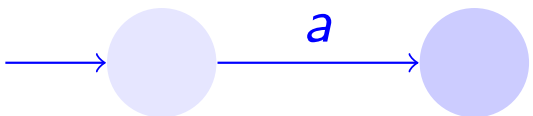
Theorem

Eine Sprache S ist genau dann regulär, wenn es einen regulären Ausdruck R gibt, sodaß $S = L(R)$.

1. *Wenn eine Sprache durch einen regulären Ausdruck beschrieben wird, dann ist sie regulär.*
2. *Zu jeder regulären Sprache gibt es einen regulären Ausdruck, der sie beschreibt.*

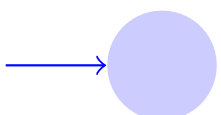
Konstruktion eines NEA zu R

1. $R = a$ für ein $a \in \Sigma$



   $N_1 = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$ mit $\delta(q_1, a) = q_2$, $\delta(r, b) = \emptyset$ für $r \neq q_1, b \neq a$

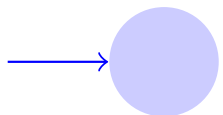
2. $R = \varepsilon$



   $N_2 = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$ mit $\delta(r, b) = \emptyset$ für jedes r und b

Konstruktion eines NEA zu R II

3. $R = \emptyset$ dann ist $L(R) = \emptyset$

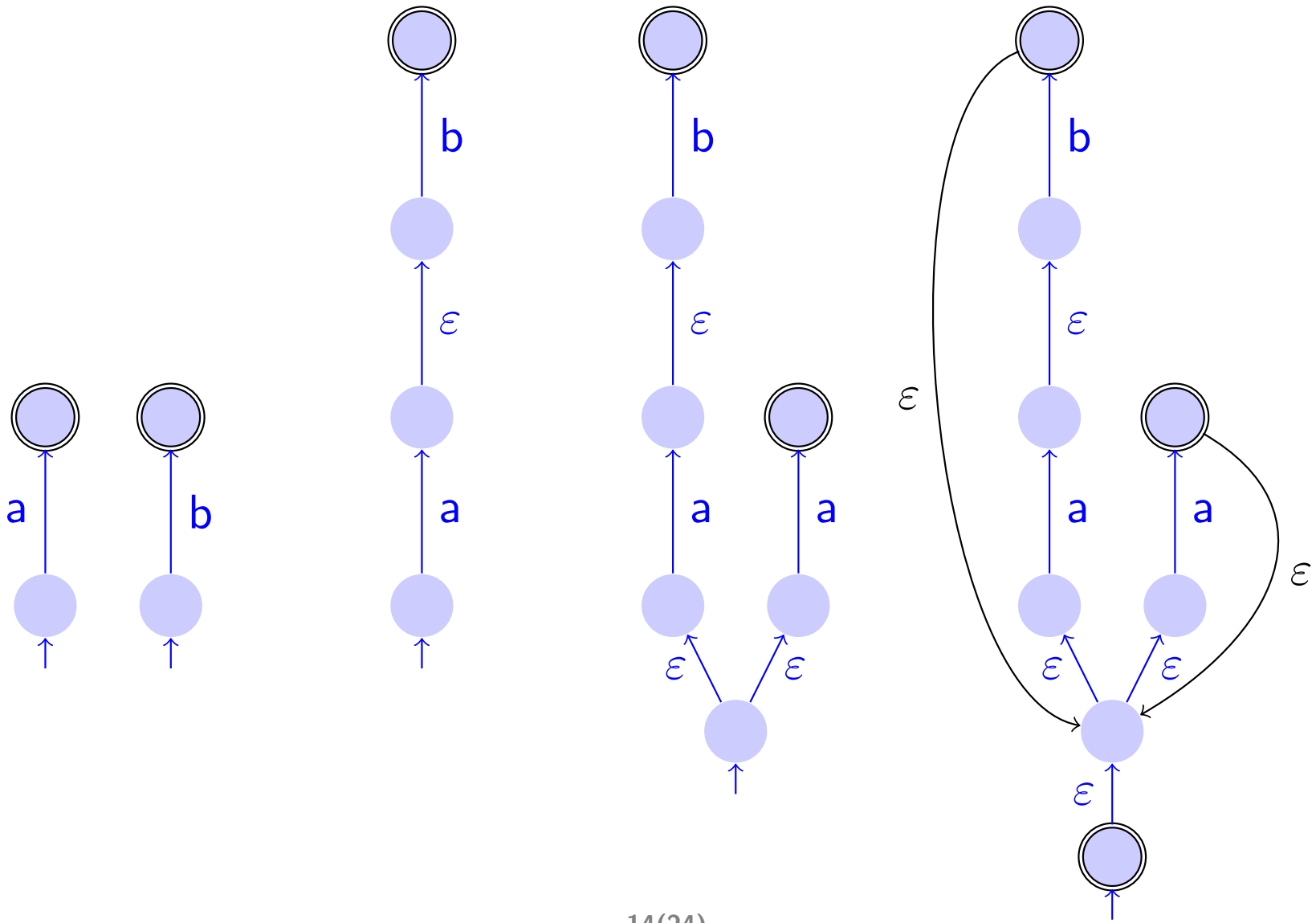


$N_1 = (\{q\}, \Sigma, \delta, q, \emptyset)$ mit $\delta(r, b) = \emptyset$ für jedes r und b

4. $R = R_1 + R_2$
5. $R = R_1 \circ R_2$
6. $R = R_1^*$

Für die Fälle 4-6 werden die NEA-s für R aus denen für R_1 und R_2 konstruiert (NEA-Sprachen sind abgeschlossen unter $\circ, \cup, ^*$). □

Beispiel: NEA zu $(ab + a)^*$



Äquivalenz zu endlichen Automaten II

Theorem

Eine Sprache S ist genau dann regulär, wenn es einen regulären Ausdruck R gibt, sodaß $S = L(R)$.

1. *Wenn eine Sprache durch einen regulären Ausdruck beschrieben wird, dann ist sie regulär.*
2. *Zu jeder regulären Sprache gibt es einen regulären Ausdruck, der sie beschreibt.*

Beweisidee

- regulären Sprachen werden von DEAs erkannt
- Prozedur zum Umwandeln von DEAs in äquivalente RAs
- Zwischenstufe: verallgemeinerte nichtdeterministische endliche Automaten (VNEA)
- Konversion von DEAs in VNEAs und von VNEAs in Reguläre Ausdrücke

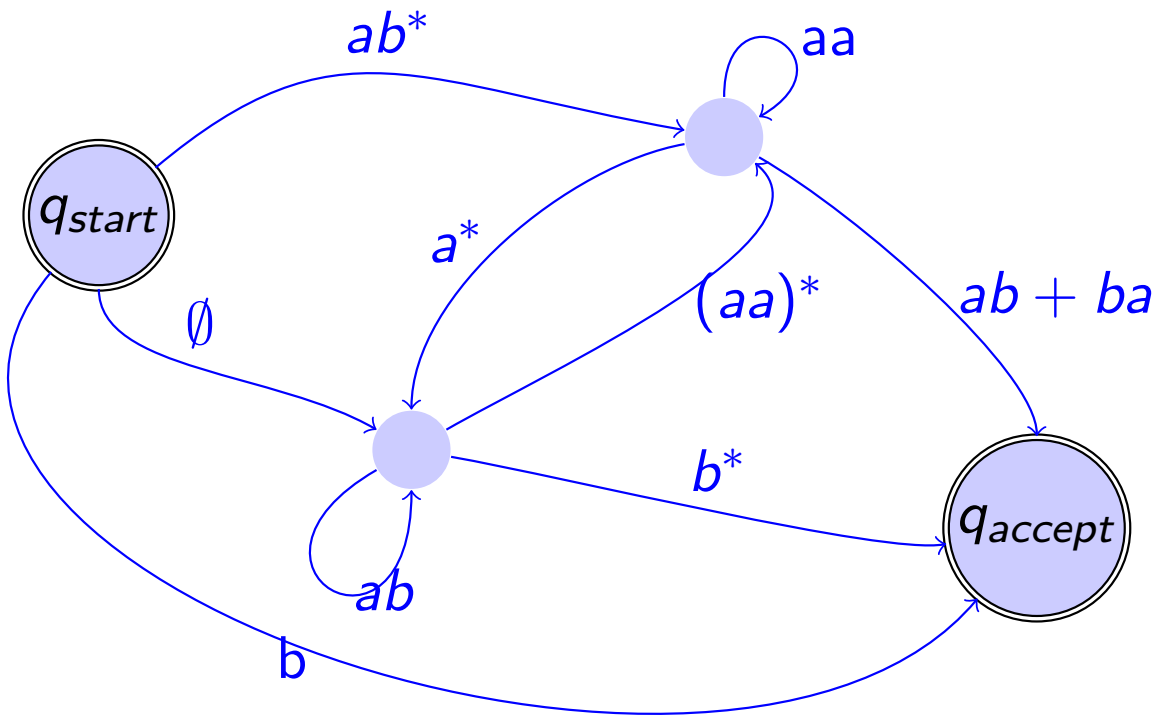
Verallgemeinerte nichtdeterministische endliche Automaten

- sind NEAs mit regulären Ausdrücken an den Zustandsübergängen
- VNEAs lesen Blöcke statt einzelner Zeichen auf einmal und bewegt sich entlang der "Pfeile"
- ein Block ist eine Zeichenkette, die durch den Regulären Ausdruck am Pfeil beschrieben wird
- wird auf einem der möglichen Wege ein Endzustand erreicht , wird das Wort akzeptiert

Spezielle Bedingungen für VNEAs

1. vom Start gehen Pfeile zu jedem Zustand, es gibt keinen Pfeil zum Start,
2. nur ein akzeptierender Zustand (verschieden vom Start) mit Pfeilen von allen Zuständen, es gibt keinen Pfeil, der herausführt
3. von jedem anderen Zustand geht ein Pfeil zu jedem anderem Zustand einschließlich zu sich selbst

Beispiel VNEA



Schritt 1: Konversion DEA in VNEA

1. Hinzufügen eines neuen Startzustandes mit ε-Übergang zum vorherigen Start,
2. Hinzufügen eines neuen akzeptierenden Zustandes mit ε-Übergängen von jedem vorherigen akzeptierenden Zustand,
3. gibt es mehrere Pfeile (oder meherer Label) von einem zu einem anderem Zustand ersetzen wir diese durch einen einzigen dessen Label die Vereinigung der vorherigen Labels ist,
4. wir ergänzen mit $\emptyset$-Pfeilen, wo vorher keine Pfeile waren (hat keinen Einfluß auf die Sprache)

Schritt 2: Konversion VNEA in Regulären Ausdruck

1. angenommen der VNEA hat n Zustände, wegen der Bedingungen gilt $n \geq 2$
2. für $n = 2$ ist das Label am einzigen Pfeil der gesuchte Ausdruck
3. für $n > 2$ konstruieren wir einen VNEA mit $(n-1)$ Zuständen und wiederholen die Prozedur bis wir bei 2 Zuständen ankommen

Konstruktion eines äquivalenten VNEA mit einem Zustand weniger

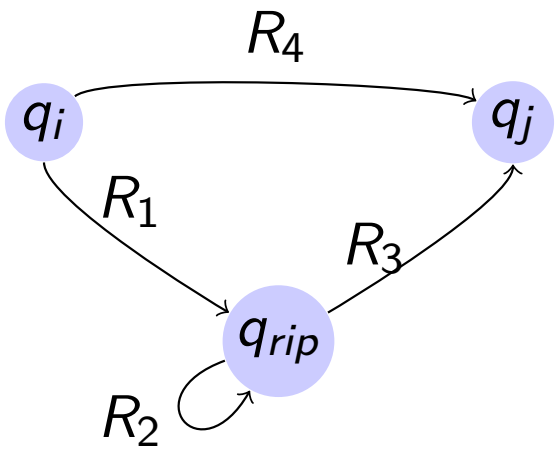
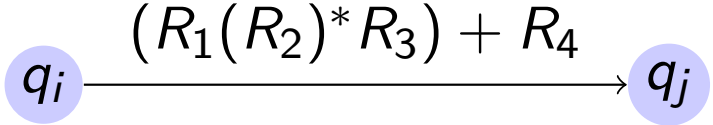


Figure: davor

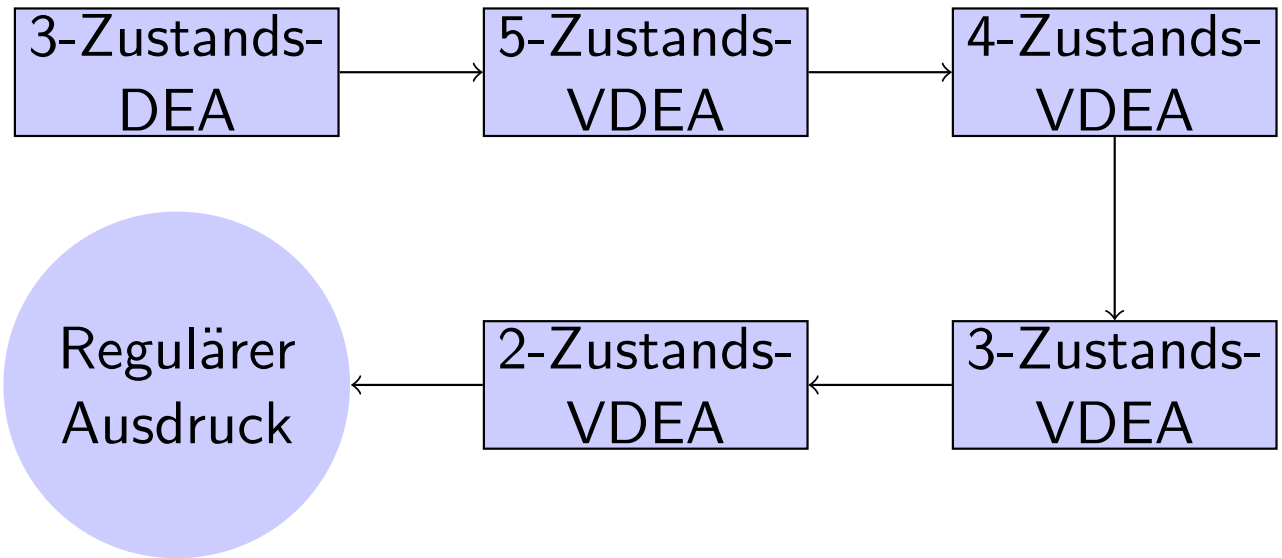


danach

Verwandlung von n-Zustands-VNEA in $(n-1)$-Zustands-VNEA

1. Auswählen eines (beliebigen) Zustandes q_{rip} verschieden von Start und Ende
2. Entfernen von q_{rip} und "Reparieren" des Automaten: die neuen Label müssen die durch das Entfernen von q_{rip} verlorenen Berechnungen ersetzen; das neue Label von q_i nach q_j ist der RA, der alle Strings beschreibt, die die vorherige Maschine braucht, um von q_i nach q_j zu kommen, direkt oder über q_{rip}

Beispiel Zustandsreduktion für VNEAs



1. Umwandeln DEA in VNEA mit n Zustände, wobei $n \geq 2$
2. für $n = 2$ ist das Label am einzigen Pfeil der gesuchte Ausdruck
3. für $n > 2$ konstruieren wir einen VNEA mit $(n-1)$ Zuständen und wiederholen die Prozedur bis wir bei 2 Zuständen ankommen