

Reguläre Sprachen

Definition

Die Menge $\mathcal{L}_{REG(I)}$ der **regulären Sprachen** ist rekursiv wie folgt definiert:

- $\emptyset, \{\lambda\}, \{x\} \in \mathcal{L}_{REG(I)}$ für $x \in I$,
- $L, L_1, L_2 \in \mathcal{L}_{REG(I)}$ impliziert $L_1 \cup L_2, L_1 L_2, L^* \in \mathcal{L}_{REG(I)}$.

Beispiel

$$(\{\lambda\} \cup (\{a\}\{b\}\{b\}))^* = (\{\lambda\} \cup \{abb\})^* = \{(abb)^n \mid n \in \mathbb{N}\}.$$

Endliche Automaten erkennen reguläre Sprachen

Satz

Jede von einem endlichen Automaten erkannte Sprache ist regulär.

Reguläre Ausdrücke

- ▶ Beschreibung regulärer Sprachen
- ▶ Anwendungsgebiete:
 - **Compilerbau**: Lexikalische Analyse, Eingabe für Scannergeneratoren
 - **Softwaretechnik**: Spezifikation zulässiger Eingaben für Dialogschnittstellen
 - **Suchbefehle**: Unix, Webbrowser
 - **Kontrollbedingungen, Ablaufspezifikationen**
 -

Definition

- I : endliches Alphabet mit $\lambda, \text{empty}, +, \circ, *, (,) \notin I$

Menge $REX(I)$ der *regulären Ausdrücke* über I

1. $\text{empty}, \lambda \in REX(I)$,
2. $I \subseteq REX(I)$,
3. für alle $r, r_1, r_2 \in REX(I)$ sind auch $(r_1 + r_2)$, $(r_1 \circ r_2)$ und (r^*) in $REX(I)$.

Sprache regulärer Ausdrücke

Jedem regulären Ausdruck r über I wird wie folgt eine Sprache $L(r)$ zugeordnet:

1. $L(empty) = \emptyset$, $L(lambda) = \{\lambda\}$ und $L(x) = \{x\}$ für alle $x \in I$,
2. für alle $r, r_1, r_2 \in REX(I)$
 - (a) $L((r_1 + r_2)) = L(r_1) \cup L(r_2)$,
 - (b) $L((r_1 \circ r_2)) = L(r_1)L(r_2)$,
 - (c) $L((r^*)) = L(r)^*$.

Klammerregeln/Vereinfachung

1. Der Operator $*$ bindet stärker als \circ ,
entsprechende Klammern dürfen entfallen
2. Der Operator \circ bindet stärker als $+$,
entsprechende Klammern dürfen entfallen
3. Der Operator \circ und $+$ sind assoziativ,
entsprechende Klammern dürfen entfallen
4. Der Operator \circ darf entfallen.

Beispiel (Vereinfachung)

$$\begin{aligned} (((a \circ (b^*)) + c)^*) &= ((a \circ (b^*)) + c)^* = ((a \circ b^*) + c)^* = \\ &= (a \circ b^* + c)^* = (ab^* + c)^* \end{aligned}$$

Beispiel (Regulärer Ausdruck und seine Sprache)

$(ab^* + c)^*$ ist ein regulärer Ausdruck über $\{a, b, c\}$ mit

$$\begin{aligned} L((ab^* + c)^*) &= L(ab^* + c)^* = (L(ab^*) \cup L(c))^* = \\ &= (L(ab^*) \cup \{c\})^* = ((L(a)L(b^*)) \cup \{c\})^* = \\ &= ((\{a\}L(b)^*) \cup \{c\})^* = ((\{a\}\{b\}^* \cup \{c\})^* = \\ &= (\{ab^n \mid n \in \mathbb{N}\} \cup \{c\})^* = \\ &= \{w \in \{a, b, c\}^* \mid w \text{ fängt nicht mit } b \text{ an und } cb \text{ ist kein} \\ &\text{Teilwort von } w\} \end{aligned}$$

Rechenregeln

1. $empty \circ r = empty = r \circ empty$
2. $empty^* = lambda = lambda^*$
3. **Kommutativgesetz:** $r + t = t + r$
4. **Neutralität:** $r + empty = r = lambda \circ r = r \circ lambda$
5. **Distributivgesetze:** $r(s + t) = rs + rt$, $(s + t)r = sr + st$
6. **Idempotenz:** $(r^*)^* = r^*$, $r + r = r$
7. **Assoziativgesetze:** $(r + s) + t = r + (s + t) = r + s + t$,
 $(r \circ s) \circ t = r \circ (s \circ t) = r \circ s \circ t = rst$

Satz

1. Jeder reguläre Ausdruck r definiert eine reguläre Sprache, d.h. $L(r) \in \mathcal{L}_{REG(I)}$.
2. Jede reguläre Sprache $L \in \mathcal{L}_{REG(I)}$ lässt sich durch einen regulären Ausdruck beschreiben, d.h. es existiert ein regulärer Ausdruck $r \in \mathcal{L}_{REG(I)}$ mit $L(r) = L$.

Fazit

- ▶ Endliche Automaten erkennen genau die Klasse der regulären Sprachen.
- ▶ Reguläre Ausdrücke beschreiben genau die Klasse der regulären Sprachen.

$$\mathcal{L}_{REG(I)} = \mathcal{L}_{REX(I)} = \mathcal{L}_{AUT(I)}$$

mit $\mathcal{L}_{REX(I)} = \{L(r) \mid r \in REX(I)\}$.

Pumping-Lemma für reguläre Sprachen

Endliche Automaten

- Modellierungskonzept mit vielen brauchbaren Eigenschaften
- schnelle Spracherkennung
- graphisch-visuelle Beschreibung
- automatische Korrektheitsbeweise
- gute Kompositionalitätseigenschaften

Aber:

- Was können endliche Automaten nicht?

Erstes Pumping-Lemma

Sei L eine von einem endlichen Automaten erkannte Sprache.

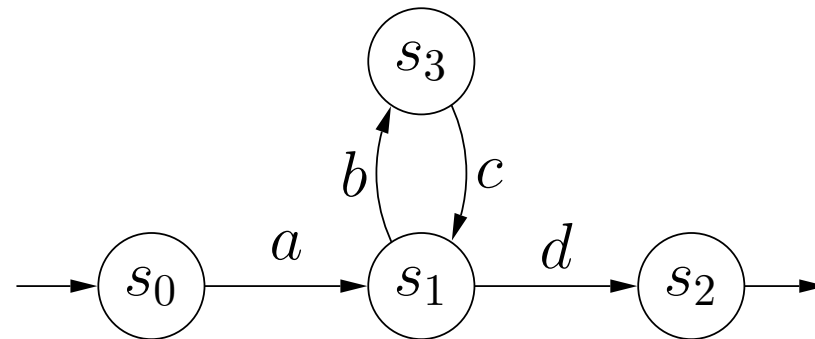
Dann existiert eine natürliche Zahl $p \in \mathbb{N}$ derart, dass jedes Wort $w \in L$ mit $\text{length}(w) \geq p$ zerlegt werden kann in drei Teilwörter $w = xyz$ mit

1. $\text{length}(xy) \leq p$
2. $y \neq \lambda$
3. $xy^iz \in L$ für alle $i \geq 0$.

Beispiel

$$L = \{a(bc)^n d \mid n \in \mathbb{N}\}$$

EA für L :



Für $p = 4$ hat jedes $w \in L$ mit $\text{length}(w) \geq 4$ die Form $a(bc)^n d$ mit $n \geq 1$. Eine **mögliche Zerlegung** von w in xyz ist $x = a$, $y = bc$, $z = (bc)^{n-1}d$, denn $\text{length}(xy) = 3 \leq 4$, $y \neq \lambda$ und $xy^i z = a(bc)^i (bc)^{n-1} d \in L \quad \forall i \in \mathbb{N}$.

Anwendung des Pumping-Lemmas

Das Pumping-Lemma kann benutzt werden, um zu zeigen, dass eine Sprache von keinem endlichen Automaten erkannt wird.