Reguläre Ausdrücke

Satz von Kleene-Myhill

$$E(Reg_\Sigma) = \mathcal{R}eg_\Sigma$$

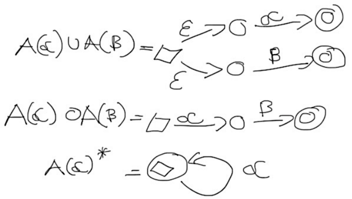
Dieser Satz bedeutet u.a., dass jede reguläre Sprache aus den endlichen Sprachen durch die Operationen *Vereinigung*, *Produkt* und *Stern* erzeugt werden kann.

Die Abbildung $\mathcal{A}$ ist wieder induktiv definiert:

- $\mathcal{A}(\emptyset) = \Diamond$,
- $\mathcal{A}(\epsilon) = \rightarrow \circledcirc$,
- $\forall a \in \Sigma : \mathcal{A}(a) = \Diamond \xrightarrow{a} \circledcirc$,

und für reguläre Ausdrücke α und β soll gelten:

- $\mathcal{A}([\alpha \cup \beta]) := \mathcal{A}(\alpha) \cup \mathcal{A}(\beta)$,
- $\mathcal{A}([\alpha\beta]) := \mathcal{A}(\alpha) \circ \mathcal{A}(\beta)$,
- $\mathcal{A}(\alpha^*) := \mathcal{A}(\alpha)^*$.



Pumping-Lemma Kontextfrei

Wichtig: $L_1 \cup L_2$ & $L_1 \circ L_2$ zweier kontextfreien Sprachen sind nicht zwingend kontextfrei!

Wir nehmen an, dass die Sprache L kontextfrei ist: $xs^iyt^iz \in L$

1. $|st| \geq 1$,
2. $|syt| \leq n$,
3. $xs^iyt^iz \in L$ *für alle* $i \in \mathbb{N}$.

$\forall n \in \mathbb{N}^* : \exists \omega \; L \quad \exists x, s, y, t, z \in \Sigma^*$

$\omega = a \ldots ab \ldots bc \ldots c \Rightarrow |w_n| = 3n$

$x \,|\, \underbrace{s \,|\, y \,|\, t}_{\leq n} \,|\, z$

Chromsky Normalform

1. Schritt: Elimination von Regeln der Form $A \longrightarrow \epsilon$, falls $A \neq S_0$.

Die Regel $A \longrightarrow \epsilon$ wird gestrichen und für jedes Vorkommen von A auf der rechten Seite einer Regel wird A durch ϵ ersetzt.

2. Schritt: Elimination von Regeln der Form $A \longrightarrow B$.

Regeln vom Typ $A \longrightarrow B$ werden Kettenregeln und Variablen, die in Kettenregeln vorkommen, Kettenvariable genannt. Ihre Elimination ist der komplizierteste Schritt. Wir zerlegen ihn in zwei Teilschritte.

Schritt 2.1: Zuerst prüfen wir, ob es Kettenregeln gibt, die einen Zykel bilden, z.B.

$$A \longrightarrow B \; , \; B \longrightarrow C \; , \; C \longrightarrow A \; .$$

In diesem Fall wählen wir eine Variable, z.B. B, aus und ersetzen alle anderen Variablen, die im Zykel vorkommen, in allen Regeln durch B. Wenn eine der Variablen S ist, müssen Sie die Variable S auswählen, sonst fehlt Ihnen die Startvariable.

Schritt 2.2:

1. Bestimme für alle $A \in \mathcal{N}$ die Menge $[A]^+$.
2. Lösche alle verbleibenden Kettenregeln.
3. Führe den folgenden Prozess durch:

   for $A \in \mathcal{N}$:

       for $B \in [A]^+$:

           if $(B \longrightarrow \omega) \in R$: put the rule $(A \longrightarrow \omega)$ into R:

3.Schritt: Elimination nutzloser Variablen.

Definition 3.12 *Sei* $A \in \mathcal{N}$. *Wir sagen, dass* A *terminiert* $:\Longleftrightarrow$

Entweder es gibt eine Regel $A \longrightarrow \omega$ *mit* $\omega \in \mathcal{T}^+$

oder es gibt eine Regel $A \longrightarrow \omega$ *mit* $\omega \in ((\mathcal{N} - \{A\}) \cup \mathcal{T})^+$ *wobei alle in* ω *vorkommenden Variablen terminierend sind.*

Definition 3.13 *Sei* $A \in \mathcal{N}$. *Die Variable* A *heisst nutzlos* $:\Longleftrightarrow$

Entweder terminiert A *nicht*

oder A *ist von* S *aus nicht erreichbar.*

Es können alle Regeln gestrichen werden, die auf der linken oder rechten Seite eine nutzlose Variable enthalten.

4. Schritt: Abändern von Regeln der Form $A \longrightarrow \omega$ mit $|\omega| \geq 2$.

Zuerst führen wir für jedes Terminalsymbol eine neue Variable und eine neue Regel ein. Ist z.B. $a \in \mathcal{T}$ führen wir die Variable Y_a und die Regel $Y_a \longrightarrow a$ ein. (Das tun wir natürlich nur für Terminalsymbole, die wirklich auf den rechten Seiten von entsprechenden Regeln vorkommen.) Dann ersetzen wir die in ω vorkommenden Terminalsymbole durch ihre Variablen, z.B. wird

$$A \longrightarrow aBbCCa \quad \text{durch} \quad A \longrightarrow Y_aBY_bCCY_a$$

ersetzt. Nun werden die Regeln der Form $A \longrightarrow BBAC$ durch eine Folge von Regeln, die die gewünschte Form haben, ersetzt:

$$A \longrightarrow BBAC \quad \Longrightarrow \quad A \longrightarrow BZ_1 \; , \; Z_1 \longrightarrow BZ_2 \; , \; Z_2 \longrightarrow AC \; ,$$

wobei Z_1, Z_2 neue Variablen sind.

Beispiel

Eti 2

Bsp 1) $G = (N, T, R, S)$

$N = \{S, A, B\}$, $T = \{*, +, (,), a\}$

$R = \{ S \to S+A \,|\, \cancel{A} \,|\, A*B \,|\, (S) \,|\, a$

$A \to A*B \,|\, \cancel{B} \,|\, (S) \,|\, a$

$B \to (S) \,|\, a \}$

CNF

$\Rightarrow N = NN$

$N = T$

Elimination der Kettenregel

$S \to A \to B$

$[S]^+ = \{A, B\}$

$[A]^+ = \{B\}$

$[B]^+ = \emptyset$

S terminiert

A "

B "

Schritt 4+5

$Y_*, Y_+, Y_(, Y_)$

$S \to SY_+A \,|\, AY_*B \,|\, Y_(SY_) \,|\, a$

$A \to AY_*B \,|\, Y_(SY_) \,|\, a$

$B \to Y_(SY_) \,|\, a$

$\tilde{G}$ in CNF

$\tilde{N} = \{S, A, B, C$ $\quad T = \{*, +, (,), a\}$

Regeln:

Kontext freie Regeln

$S \to SC$ $\quad B \to Y_(E$

$S \to AD$

$S \to Y_(E$

$A \to AD$

$A \to Y_(E$

$C \to Y_+A$

$D \to Y_*B$

$E \to SY_)$

Reguläre Regeln

$S \to a$

$A \to a$

$B \to a$

$Y_* \to *$

$Y_+ \to +$

$Y_(\to ($

$Y_) \to)$

Cocke-Younger-Kasami

Ist Wort in kontextfreier Grammatik?
CYK löst das das Problem in polynomieller Zeit.
2110

$G = (\{S, A, B, C\}, \{a, b\}, P, S)$

$S \rightarrow AB \mid BC$
$A \rightarrow BA \mid a$
$B \rightarrow CC \mid b$
$C \rightarrow AB \mid a$

$w = bbabaa$

A,S	/	/	/	/	/
B	A,S	/	/	/	/
S,C	B	S,A	/	/	/
A	S,C	B	Ø	/	/
Ø	S,A	S,C	A,S	B	/
B	B	A,C	B	A,C	A,C
b	b	a	b	a	a

Kellerautomaten

Q, *einer endlichen Menge von* **Zuständen**,

Σ, *einem endlichen Eingabealphabet*,

Γ, *einem endlichen Keller- oder Stackalphabet*,

$\delta : Q \times \Sigma_\epsilon \times \Gamma \longrightarrow P(Q \times \Gamma^*)$, *der* **Übergangsfunktion**,

q_0, *dem* **Startzustand**,

$\perp \in \Gamma$, *dem Zeichen, das den Boden des Stacks markiert, und*

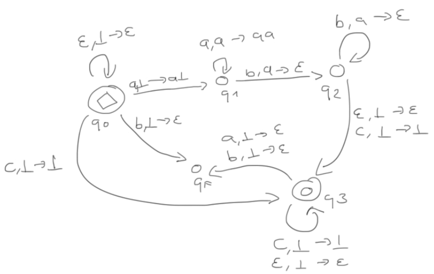
F, *der Menge der akzeptierenden Zustände.*

$$M = (Q,\ \Sigma,\ \Gamma,\ \delta,\ q_0,\ \perp,\ F)$$

q_0	ϵ	$\perp$	q_0	ϵ
q_0	0	$\perp$	q_0	$0\perp$
q_0	0	0	q_0	00
q_0	1	$\perp$	q_r	ϵ

Beispiel

$L = \{a^m b^m c^n \mid m, n \in N\}$
$M = \{Q, \Sigma, \Gamma, \delta, q_0, \perp, F\}$
$\Sigma = \{a. b. c\}$
$\Gamma = \{a, \perp\}$
$F = \{q_0, q_3\}$

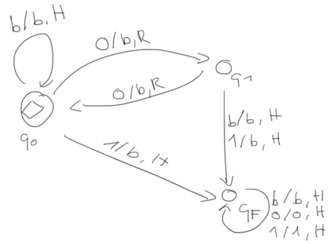


Turing-Maschine (TM)

Maschine kann von Band lesen und schreiben, Kopf kann fahren (Tabelle wie bei Kellerauto.)

- Zeitbedarf (Anzahl Schritte bis Stop)
- Platzbedarf (Anzahl Speicherstellen die der Kopf berechnet bis Stop)

$M = (Q, \Sigma, \Gamma, \delta, q_0, F)$
$Q = \{q_0, q_1, q_F\}$
$\Sigma = \{0, 1\}$
$\Gamma = \{0, 1, b\}$
$F = \{q_0\}$



Notation:

Input $w = 00$

$q_0 00 \vdash b q_1 0 \vdash b b q_0 b \Rightarrow 00 \in L(M)$

Berechenbarkeit

Prim-Rekursive Funktionen

$Z() = 0$
$S(x) = x + 1$
$P(x) = x - 1 \mid 0$
$U_j^n(a_1, a_2, a_3, \ldots, a_n) = a_j$

$f(x, y)$ 2-Stellig ← $g(x)$ 1-Stellig ← $h(x, y, z)$ 3-stellig

$f(x, 0) = g(x)$
$f(x, 1) = h(x, 0, f(x, 0))$
$f(x, 2) = h(x, 1, f(x, 1))$
$\boxed{f(x, y+1) = h(x, y, f(x, y))}$

Bsp: $P(y)$ 1-Stellig
$P(0) = Z \,//\, 0$
$P(y+1) = h(y, P(y)) = y$
$= U_1^2(y, P(y))$

Loop-Berechenbar?

Ist die folgende Funktion Loop-berechenbar?

$$f : \mathbb{N} \longrightarrow \mathbb{N}$$

mit

$$f(n) = \begin{cases} 0 & \text{, falls } n \text{ gerade,} \\ 1 & \text{, sonst.} \end{cases}$$

```
Input:  x1
Output: x1

x2:=0;
x3:=0;
x4:=S(x2);
loop x1 do
        x2:=x4;
        x4:=x3;
        x3:=x2
od;
x1:=x2
```

Prüfung

Es sei $L \subset \Sigma^*$ eine Sprache und $\bar{L}$ ihr Komplement.

Entscheiden Sie, ob die folgenden Aussagen richtig oder falsch sind.

a) Wenn L regulär ist, dann ist auch $\bar{L}$ regulär.

b) Wenn L und $\bar{L}$ semi-entscheidbar sind, dann ist L rekursiv.

c) Wenn L semi-entscheidbar ist und $\bar{L}$ rekursiv aufzählbar, dann ist L rekursiv.

d) Jede kontextsensitive Sprache ist rekursiv.

e) L kann rekursiv sein und $\bar{L}$ nicht.

f) Es gibt Sprachen, für die es keine erzeugende Grammatik gibt.

a) r
b) r
c) r
d) r
e) f
f) r