Karlsruher Institut für Technologie

# Tutorium 42, #13

Max Göckel– *uzkns@student.kit.edu*

Institut für Theoretische Informatik - Grundbegriffe der Informatik

# Sprachen

Es gibt Sprachen, die ein Automat entscheiden kann, z.B. $\{a^n | n \in \mathbb{N}\}$.

Aber es gibt auch solche, die kein Automat entscheiden kann, z.B. $\{a^n b^n | n \in \mathbb{N}\}$.
Für diese Sprache müsste der Automat die unendlich hohe Anzahl der a's zählen, was in endlich vielen Zuständen nicht machbar ist.

# Sprachen

Es gibt demnach verschiedene Arten (oder "Klassen") von Sprachen. Diese sind sortiert nach der sog. Chomsky-Hierarchie.

In GBI interessieren uns nur die Sprachen von Typ 2 und Typ 3 (kontextfrei und regulär). Dabei gilt jede Chomsky-Typ-3-Sprache ist auch eine Chomsky-Typ-2-Sprache, bzw. für jede reguläre Sprache existiert eine kontextfreie Grammatik.

# Reguläre Sprachen: Grundlagen

Reguläre Sprachen sind also die *eingeschränktesten* Sprachen von allen formalen Sprachen.

Für eine Sprache $L$ gilt:
L ist eine reguläre Sprache $\Leftrightarrow$
$L$ wird durch einen Automaten $A$ akzeptiert $\Leftrightarrow$
$\exists$ Regulärer Ausdruck $R$ der $L$ beschreibt $\Leftrightarrow$
$L$ wird von einer rechtsregulären Grammatik $G_1$ erzeugt $\vee$ $L$ wird von einer linksregulären Grammatik $G_2$ erzeugt

Wir interessieren uns erst nur für reguläre Ausdrücke.

# Reguläre Ausdrücke: Grundlagen

Ein regulärer Ausdruck besteht aus zwei Alphabeten: $A$ und $Z = \{$

- $|$ $\Leftrightarrow$ "oder", zB $a|b$
- $*$ $\Leftrightarrow$ "beliebig oft", zB $a*$
- $\emptyset$ $\Leftrightarrow$ "nichts", ähnlich zu $\epsilon$
- $(,)$ $\Leftrightarrow$ Klammern

$\}$

Atomare reguläre Ausdrücke sind dabei: $\emptyset$ und $a \in A$
Daraus lassen sich wieder Ausdrücke zusammensetzen:
$(R_1*), (R_1|R_2), (R_1R_2)$ falls $R_1$ und $R_2$ wieder reg. Ausdrücke sind

# Reguläre Ausdrücke: Aufgabe

Sei $A = \{a, b\}$

| Ausdruck | regulär (ja/nein) |
|---|---|
| $((a\|b)*)$ | |
| $((\emptyset\emptyset)\|a)$ | |
| $(\|aa)$ | |
| $(*bab)$ | |
| $(r_1\|a$ | |

# Reguläre Ausdrücke: Lösung

| Ausdruck | regulär (ja/nein) | Wieso nicht? |
|---|---|---|
| $((a\|b)*)$ | ja | - |
| $((\emptyset\emptyset)\|a)$ | ja | - |
| $(\|aa)$ | nein | vor $\|$ fehlt ein Ausdruck |
| $(*bab)$ | nein | vor $*$ fehlt ein Ausdruck |
| $(c\|a$ | nein | $r_1 \notin A$ und nicht richtig geklammert |

# Reguläre Ausdrücke: Klammereinsparung

Auch bei regulären Ausdrücken kann man die Klammern weglassen um den Ausdruck lesbarer zu machen.

Die Wichtigkeit der Operationen ist: $*$ vor Konkatenation vor $|$.

Damit entspricht $((a|(b(a*)))|(ba))$ dem Ausdruck $a|ba*|ba$.

# Reguläre Ausdrücke: Sprachen

Wir wissen: Ein regulärer Ausdruck beschreibt eine Sprache, die auch von einem Automaten akzeptiert wird.

Ergo können wir zu jedem Ausdruck $R$ auch die Sprache $< R >$ in bekannter Mengenschreibweise finden, die von R beschrieben wird. Ausserdem können wir dann einen Automaten erstellen, der genau $< R >$ akzeptiert.

# Reguläre Ausdrücke: Aufgabe

Erstellt zu den Ausdrücken die jeweiligen Sprachen in Mengenschreibweise.

- $R_1 = (a|b) * abba(a|b)*$
- $R_2 = b * * * * *$
- $R_3 = abb * |ab * a$

# Reguläre Ausdrücke: Lösung

Erstellt zu den Ausdrücken die jeweiligen Sprachen in Mengenschreibweise.

- $R_1 = (a|b) * abba(a|b)*$
  - $\{w_1 abbaw_2 | w_1, w_2 \in \{a, b\}^*\}$
- $R_2 = b * * * **$
  - $\{b\}^*$
- $R_3 = abb * |ab * a$
  - $\{ab^n | n \in \mathbb{N}_+\} \cup \{ab^m a | m \in \mathbb{N}_0\}$

# Reguläre Ausdrücke: Aufgabe

Gebt den Ausdruck für die Sprachen an.

- Alle Wörter $w \in \{a, b\}^*$ in denen *genau* zwei Mal ein $a$ vorkommt.
- $\{\}$
- $\{\epsilon\}^*$
- Alle $w \in \{a, b\}^*$ in denen aba nicht vorkommt

# Reguläre Ausdrücke: Lösung

Gebt den Ausdruck für die Sprachen an.

- Alle Wörter $w \in \{a, b\}^*$ in denen *genau* zwei Mal ein $a$ vorkommt.
  - $(b*)a(b*)a(b*)$
- $\{\}$
  - $\emptyset$
- $\{\epsilon\}$
  - $\emptyset*$
- Alle $w \in \{a, b\}^*$ in denen aba nicht vorkommt
  - $b * (a * bbb*) * a * (b|\emptyset*)$

# Reguläre Ausdrücke: Aufgabe

Gebt den Ausdruck zum endlichen Automat/Akzeptor an.

# Reguläre Ausdrücke: Aufgabe

Gebt den Ausdruck zum endlichen Automat/Akzeptor an.

$(ab*a)*(bb(b)\emptyset*)|\emptyset*)$

# Rechtsreguläre Grammatiken: Grundlagen

Rechtsreguläre Grammatiken sind besondere Kontextfreie Grammatiken $G = (N, T, S, P)$, bei denen alle $p_x \in P$ so aussehen:

- $X \to w, X \in N, w \in T^*$
- $X \to wY, X, Y \in N, w \in T^*$

Zu jeder Rechtsregülären Grammatik $G$ kann man einen Ausdruck $R$ finden.

# Rechtsreguläre Grammatiken: Aufgabe

Gebt eine Grammatik zum Ausdruck an und umgekehrt.

- $R_1 = a * b$
- $X \to aX|b|\epsilon$

# Rechtsreguläre Grammatiken: Aufgabe

Gebt mögliche Produktionen zum Ausdruck an und umgekehrt.

- $a * b$
  - $X \to aX|bY, Y \to \epsilon$
- $X \to aX|b|\epsilon$
  - $a * (b|\emptyset *)$

# Rechtsreguläre Grammatiken: Aufgabe

Gebt einen endlichen Automat zur Sprache $G$ an.

$G = \{\{X, Y, Z\}, \{a, b\}, X, P\}$
$P = \{$
$X \to aX|bY|\epsilon,$
$Y \to aX|bZ|\epsilon,$
$Z \to aZ|bZ$
$\}$

# Reguläre Ausdrücke: Kantrowitsch-Bäume

Reguläre Ausdrücke können, wie ein Ableitungsbaum, grafisch dargestellt werden.
Im Unterschied zu den Ableitungsbäumen stehen im Ausdruck alle Knoten.

Beispielausdruck: $R_1 = (a(b|c)*)a$.

# Turing-Maschinen

Eine Turingmaschine ist ein Modell zur Funktionsweise eines Computers. Der Mathematiker Alan Turing erfand sie 1936 um das Entscheidungsproblem zu lösen.
Eine Turingmaschine kann jeden Algorithmus simulieren und damit auch ganze Programme und Betriebssysteme. Das und ihre Formalisierung dieser Begriffe macht sie zu einem mathematischen Objekt und kann damit mit Funktionen und Methoden untersucht werden. (dazu mehr in TGI)

# Turing-Maschinen: Funktionsweise

Eine Turingmaschine besteht aus einem Lese-Schreibkopf und einem Array/Band mit einer Eingabe.
Pro Ausführungsschritt liest der Kopf ein Symbol vom Band, schreibt ein Symbol auf das Band und geht 1 Feld nach links, rechts, oder bleibt an der selben Stelle.

Das Band ist in beide Richtungen unbegrenzt lang und Felder ohne Symbol haben das *Blanksymbol* □

# Turing-Maschinen: Definition

Eine Turingmaschine ist ein 6-Tupel $T = (Z, z_0, X, f, g, m)$, bestehend aus:

- $Z$ als Menge der Zustände des Kopfes
- $z_0 \in Z$ als Anfangszustand
- Bandalphabet $X$
- partielle Zustandsübergangsfunktion $f : Z \times Z \to Z$
- p. Ausgabefunktion $g : Z \times X \to X$
- p. Bewegungsfunktion $m : Z \times X \to \{-1, 0, 1\}$
  - $-1 = L$ und $1 = R$

Gibt man ein nicht definiertes Tupel in $f, g, m$ ein, so hält die Turingmaschine an.

# Turing-Maschinen: Beispiel

Was tut die Turingmaschine?

# Turing-Maschinen: Beispiel

Was tut die Turingmaschine?

Sie vertauscht an jeder 2. Stelle *a* und *b*, hat die Eingabe ungerade Länge so wird das letzte Zeichen mit dem Blanksymbol überschrieben.

# Turing-Maschinen: Aufgabe

Gib jeden Band-Zustand der TM für *aabba* an.

(Am Anfang und Ende des Bandes nur so viele Blanksymbole wie nötig angeben)

# Turing-Maschinen: Aufgabe

Entwerfe t eine TM die Wörter $w \in \{a, b\}^*$ umdreht.
So soll aus dem Wort *aab* das Wort *baa* werden.

Ihr dürft das Bandalphabet beliebig um andere Hilfssymbole erweitern.