

# Grundbegriffe der Informatik

## Einheit 15: Reguläre Ausdrücke und rechtslineare Grammatiken

Prof. Dr. Tanja Schultz

Karlsruher Institut für Technologie, Fakultät für Informatik

Wintersemester 2011/2012

# Was kann man mit endlichen Akzeptoren?

- ▶ manche Sprachen kann man mit endlichen Akzeptoren erkennen
  - ▶ z. B.  $\{a\}^+ \{b\} \cup \{b\}^+ \{a\}$
- ▶ manche Sprachen *nicht*
  - ▶ z. B.  $\{a^k b^k \mid k \in \mathbb{N}_0\}$
- ▶ *Charakterisierung* der erkennbaren Sprachen?
  - ▶ i. e. Beschreibung ohne Benutzung endlicher Akzeptoren

## Reguläre Ausdrücke

- Definition

- Beschriebene formale Sprache

- Beispiel: Datums-/Zeitangaben in Emails

- Zusammenhang mit Automaten und Grammatiken

## Rechtslineare Grammatiken (Typ 3)

## Kantorowitsch-Bäume und strukturelle Induktion

## Reguläre Ausdrücke

- Definition

- Beschriebene formale Sprache

- Beispiel: Datums-/Zeitangaben in Emails

- Zusammenhang mit Automaten und Grammatiken

## Rechtslineare Grammatiken (Typ 3)

## Kantorowitsch-Bäume und strukturelle Induktion

## Reguläre Ausdrücke

### Definition

Beschriebene formale Sprache

Beispiel: Datums-/Zeitangaben in Emails

Zusammenhang mit Automaten und Grammatiken

## Rechtslineare Grammatiken (Typ 3)

## Kantorowitsch-Bäume und strukturelle Induktion

## regulärer Ausdruck

- ▶ Ursprung:  
Stephen Kleene: *Representation of Events in Nerve Nets and Finite Automata*,  
in: Shannon, McCarthy, Ashby (eds.): *Automata Studies*, 1956
- ▶ heute: verschiedene Bedeutungen
  - ▶ in dieser Vorlesung: die „klassische“ Definition
  - ▶ Verallgemeinerung: *regular expressions*
    - ▶ sehr nützlich (emacs, grep, sed, . . . , Java, Python, . . . )

## Definition regulärer Ausdrücke (1)

- ▶ sei  $A$  ein Alphabet, das kein Zeichen aus  $Z$  enthält
- ▶ sei  $Z$  das Alphabet  $Z = \{ |, (, ), *, \emptyset \}$  („Hilfssymbole“)
- ▶ *regulärer Ausdruck* über  $A$  ist eine Zeichenfolge über dem Alphabet  $A \cup Z$ , die gewissen Vorschriften genügt.
- ▶ Menge der regulären Ausdrücke ist wie folgt festgelegt:
  - ▶  $\emptyset$  ist ein regulärer Ausdruck.
  - ▶ Für jedes  $x \in A$  ist  $x$  ein regulärer Ausdruck.
  - ▶ Wenn  $R_1$  und  $R_2$  reguläre Ausdrücke sind, dann sind auch  $(R_1 | R_2)$  und  $(R_1 R_2)$  reguläre Ausdrücke.
  - ▶ Wenn  $R$  ein regulärer Ausdruck ist, dann auch  $(R^*)$ .
  - ▶ Nichts anderes sind reguläre Ausdrücke.

Es sei  $A = \{a, b\}$ :

- $\emptyset$
- $(ab)$
- $(\emptyset|b)$
- $(\emptyset^*)$
- $((a^*)^*)$
- $a$
- $((ab)a)$
- $(a|b)$
- $(a^*)$
- $(((((ab)b)^*)^*)^*|(\emptyset^*))$
- $b$
- $((ab)a)a)$
- $((a(a|b))|b)$
- $((ba)(b^*))$
- $((ab)(aa))$
- $(a|(b|(a|a)))$
- $((ba)b)^*$



- ▶ „Stern- vor Punktrechnung“
- ▶ „Punkt- vor Strichrechnung“
- ▶ Beispiel:
  - ▶  $R_1 | R_2 R_3 *$  Kurzform für
  - ▶  $(R_1 | (R_2 (R_3 *)))$
- ▶ Bei mehreren gleichen binären Operatoren gilt das als links geklammert
- ▶ Beispiel
  - ▶  $R_1 | R_2 | R_3$  Kurzform für
  - ▶  $((R_1 | R_2) | R_3)$

Es sei  $A = \{a, b\}$ :

- $\emptyset$
- $(ab)$
- $(\emptyset|b)$
- $(\emptyset^*)$
- $((a^*)^*)$
- $a$
- $((ab)a)$
- $(a|b)$
- $(a^*)$
- $(((((ab)b)^*)^*)|(\emptyset^*))$
- $b$
- $((((ab)a)a)$
- $((a(a|b))|b)$
- $((ba)(b^*))$
- $((ab)(aa))$
- $(a|(b|(a|a)))$
- $((ba)b)^*$

Mit Klammereinsparungsregeln:

- $ab$
- $\emptyset|b$
- $\emptyset^*$
- $a^{**}$
- $aba$
- $a|b$
- $a^*$
- $(abb)^{**}| \emptyset^*$
- $abaa$
- $a(a|b)|b$
- $bab^*$
- $ab(aa)$
- $(a|(b|(a|a)))$
- $(bab)^*$

*keine* regulären Ausdrücke über  $\{a, b\}$ :

- $(|b)$  vor  $|$  fehlt ein regulärer Ausdruck
- $|\emptyset|$  vor und hinter  $|$  fehlt je ein regulärer Ausdruck
- $()ab$  zwischen  $($  und  $)$  fehlt ein regulärer Ausdruck
- $((ab)$  Klammern müssen „gepaart“ auftreten
- $*(ab)$  vor  $*$  fehlt ein regulärer Ausdruck
- $c*$   $c$  ist nicht Zeichen des Alphabetes

## Definition regulärer Ausdrücke (2)

- ▶ alternative Definition mit Hilfe einer kontextfreien Grammatik
- ▶ reguläre Ausdrücke über Alphabet  $A$  sind die Wörter, die von der folgenden Grammatik erzeugt werden:

$$G = (\{R\}, \{ |, (, ), *, \emptyset \} \cup A, R, P)$$

$$\text{mit } P = \{R \rightarrow \emptyset\} \cup \{R \rightarrow x \mid x \in A\} \\ \cup \{R \rightarrow (R|R), R \rightarrow (RR), R \rightarrow (R*)\}$$

## Reguläre Ausdrücke

Definition

**Beschriebene formale Sprache**

Beispiel: Datums-/Zeitangaben in Emails

Zusammenhang mit Automaten und Grammatiken

## Rechtslineare Grammatiken (Typ 3)

## Kantorowitsch-Bäume und strukturelle Induktion

*von einem regulären Ausdruck  $R$  beschriebene formale Sprache  $\langle R \rangle$*

- ▶  $\langle \emptyset \rangle = \{ \}$  (d. h. die leere Menge).
- ▶ Für  $x \in A$  ist  $\langle x \rangle = \{ x \}$ .
- ▶ Sind  $R_1$  und  $R_2$  reguläre Ausdrücke, so ist  $\langle R_1 \mid R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$ .
- ▶ Sind  $R_1$  und  $R_2$  reguläre Ausdrücke, so ist  $\langle R_1 R_2 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$ .
- ▶ Ist  $R$  ein regulärer Ausdruck, so ist  $\langle R^* \rangle = \langle R \rangle^*$ .

Definition folgt der für reguläre Ausdrücke

einfache Beispiele:

- ▶  $R = a|b$ : Dann ist

$$\langle R \rangle = \langle a|b \rangle = \langle a \rangle \cup \langle b \rangle = \{a\} \cup \{b\} = \{a, b\}$$

- ▶  $R = (a|b)^*$ : Dann ist

$$\langle R \rangle = \langle (a|b)^* \rangle = \langle a|b \rangle^* = \{a, b\}^*$$

- ▶  $R = (a^*b^*)^*$ : Dann ist

$$\begin{aligned}\langle R \rangle &= \langle (a^*b^*)^* \rangle = \langle a^*b^* \rangle^* \\ &= (\langle a^* \rangle \langle b^* \rangle)^* = (\langle a \rangle^* \langle b \rangle^*)^* = (\{a\}^* \{b\}^*)^*\end{aligned}$$

- ▶ Nachdenken:  $(\{a\}^* \{b\}^*)^* = \{a, b\}^*$

einfache Beispiele:

- ▶  $R = a|b$ : Dann ist

$$\langle R \rangle = \langle a|b \rangle = \langle a \rangle \cup \langle b \rangle = \{a\} \cup \{b\} = \{a, b\}$$

- ▶  $R = (a|b)^*$ : Dann ist

$$\langle R \rangle = \langle (a|b)^* \rangle = \langle a|b \rangle^* = \{a, b\}^*$$

- ▶  $R = (a^*b^*)^*$ : Dann ist

$$\begin{aligned}\langle R \rangle &= \langle (a^*b^*)^* \rangle = \langle a^*b^* \rangle^* \\ &= (\langle a^* \rangle \langle b^* \rangle)^* = (\langle a \rangle^* \langle b \rangle^*)^* = (\{a\}^* \{b\}^*)^*\end{aligned}$$

- ▶ Nachdenken:  $(\{a\}^* \{b\}^*)^* = \{a, b\}^*$



einfache Beispiele:

- ▶  $R = a|b$ : Dann ist
$$\langle R \rangle = \langle a|b \rangle = \langle a \rangle \cup \langle b \rangle = \{a\} \cup \{b\} = \{a, b\}$$
- ▶  $R = (a|b)^*$ : Dann ist
$$\langle R \rangle = \langle (a|b)^* \rangle = \langle a|b \rangle^* = \{a, b\}^*$$
- ▶  $R = (a^*b^*)^*$ : Dann ist
$$\begin{aligned}\langle R \rangle &= \langle (a^*b^*)^* \rangle = \langle a^*b^* \rangle^* \\ &= (\langle a^* \rangle \langle b^* \rangle)^* = (\langle a \rangle^* \langle b \rangle^*)^* = (\{a\}^* \{b\}^*)^*\end{aligned}$$
- ▶ Nachdenken:  $(\{a\}^* \{b\}^*)^* = \{a, b\}^*$

einfache Beispiele:

- ▶  $R = a|b$ : Dann ist
$$\langle R \rangle = \langle a|b \rangle = \langle a \rangle \cup \langle b \rangle = \{a\} \cup \{b\} = \{a, b\}$$
- ▶  $R = (a|b)^*$ : Dann ist
$$\langle R \rangle = \langle (a|b)^* \rangle = \langle a|b \rangle^* = \{a, b\}^*$$
- ▶  $R = (a^*b^*)^*$ : Dann ist
$$\begin{aligned}\langle R \rangle &= \langle (a^*b^*)^* \rangle = \langle a^*b^* \rangle^* \\ &= (\langle a^* \rangle \langle b^* \rangle)^* = (\langle a \rangle^* \langle b \rangle^*)^* = (\{a\}^* \{b\}^*)^*\end{aligned}$$
- ▶ Nachdenken:  $(\{a\}^* \{b\}^*)^* = \{a, b\}^*$

# Wie ist das denn eigentlich?

- ▶ Kann man „allgemein“ von regulären Ausdrücken  $R_1, R_2$  feststellen, ob  $\langle R_1 \rangle = \langle R_2 \rangle$  ist?
- ▶ Geht das algorithmisch?
- ▶ Welche formalen Sprachen sind denn durch reguläre Ausdrücke beschreibbar?

- ▶ Es gibt Algorithmen, um für reguläre Ausdrücken  $R_1, R_2$  festzustellen, ob  $\langle R_1 \rangle = \langle R_2 \rangle$  ist
  - ▶ sogar konzeptionell ziemlich einfache
- ▶ **Aber:** Dieses Problem ist PSPACE-vollständig.
  - ▶ Definition: in einer anderen Vorlesung
  - ▶ alle bisher bekannten (!) Algorithmen sind sehr sehr sehr sehr langsam
- ▶ Man weiß nicht, ob es vielleicht doch Algorithmen mit polynomieller Laufzeit für das Problem gibt.

►  $ab(ab)^*$

►  $\langle ab(ab)^* \rangle = \langle ab \rangle \langle (ab)^* \rangle = \{ab\} \{ab\}^* = \{ab\}^+$

► allgemein:  $R(R)^*$

►  $\langle R \rangle \langle (R)^* \rangle = \langle R \rangle \langle R \rangle^* = \langle R \rangle^+$

► gelegentlich Abkürzung  $(R)^+$  bzw.  $R^+$

►  $abc|\emptyset^*$

►  $\langle abc|\emptyset^* \rangle = \dots = \langle abc \rangle \cup \langle \emptyset^* \rangle = \langle abc \rangle \cup \langle \emptyset \rangle^*$   
 $= \{abc\} \cup \{\}^* = \{abc, \varepsilon\}$

► allgemein:  $R|\emptyset^*$ :

►  $\langle R|\emptyset^* \rangle = \langle R \rangle \cup \langle \emptyset^* \rangle = \langle R \rangle \cup \{\varepsilon\}$

► m. a. W.: das Vorkommen eines Wortes aus  $\langle R \rangle$  ist „optional“

► auch dafür verschiedene Abkürzungen (je nach Anwendung)  
z. B.  $(R)?$  bzw.  $R?$  oder  $[R]$  oder ...

►  $ab(ab)^*$

►  $\langle ab(ab)^* \rangle = \langle ab \rangle \langle (ab)^* \rangle = \{ab\} \{ab\}^* = \{ab\}^+$

► allgemein:  $R(R)^*$

►  $\langle R \rangle \langle (R)^* \rangle = \langle R \rangle \langle R \rangle^* = \langle R \rangle^+$

► gelegentlich Abkürzung  $(R)^+$  bzw.  $R^+$

►  $abc|\emptyset^*$

►  $\langle abc|\emptyset^* \rangle = \dots = \langle abc \rangle \cup \langle \emptyset^* \rangle = \langle abc \rangle \cup \langle \emptyset \rangle^*$   
 $= \{abc\} \cup \{\}^* = \{abc, \varepsilon\}$

► allgemein:  $R|\emptyset^*$ :

►  $\langle R|\emptyset^* \rangle = \langle R \rangle \cup \langle \emptyset^* \rangle = \langle R \rangle \cup \{\varepsilon\}$

► m. a. W.: das Vorkommen eines Wortes aus  $\langle R \rangle$  ist „optional“

► auch dafür verschiedene Abkürzungen (je nach Anwendung)  
z. B.  $(R)?$  bzw.  $R?$  oder  $[R]$  oder ...

►  $ab(ab)^*$

►  $\langle ab(ab)^* \rangle = \langle ab \rangle \langle (ab)^* \rangle = \{ab\} \{ab\}^* = \{ab\}^+$

► allgemein:  $R(R)^*$

►  $\langle R \rangle \langle (R)^* \rangle = \langle R \rangle \langle R \rangle^* = \langle R \rangle^+$

► gelegentlich Abkürzung  $(R)^+$  bzw.  $R^+$

►  $abc|\emptyset^*$

►  $\langle abc|\emptyset^* \rangle = \dots = \langle abc \rangle \cup \langle \emptyset^* \rangle = \langle abc \rangle \cup \langle \emptyset \rangle^*$   
 $= \{abc\} \cup \{\}^* = \{abc, \varepsilon\}$

► allgemein:  $R|\emptyset^*$ :

►  $\langle R|\emptyset^* \rangle = \langle R \rangle \cup \langle \emptyset^* \rangle = \langle R \rangle \cup \{\varepsilon\}$

► m. a. W.: das Vorkommen eines Wortes aus  $\langle R \rangle$  ist „optional“

► auch dafür verschiedene Abkürzungen (je nach Anwendung)  
z. B.  $(R)?$  bzw.  $R?$  oder  $[R]$  oder ...

- ▶  $ab(ab)^*$ 
  - ▶  $\langle ab(ab)^* \rangle = \langle ab \rangle \langle (ab)^* \rangle = \{ab\} \{ab\}^* = \{ab\}^+$
- ▶ allgemein:  $R(R)^*$ 
  - ▶  $\langle R \rangle \langle (R)^* \rangle = \langle R \rangle \langle R \rangle^* = \langle R \rangle^+$
  - ▶ gelegentlich Abkürzung  $(R)^+$  bzw.  $R^+$
- ▶  $abc|\emptyset^*$ 
  - ▶  $\langle abc|\emptyset^* \rangle = \dots = \langle abc \rangle \cup \langle \emptyset^* \rangle = \langle abc \rangle \cup \langle \emptyset \rangle^*$   
 $= \{abc\} \cup \{\}^* = \{abc, \varepsilon\}$
- ▶ allgemein:  $R|\emptyset^*$ :
  - ▶  $\langle R|\emptyset^* \rangle = \langle R \rangle \cup \langle \emptyset^* \rangle = \langle R \rangle \cup \{\varepsilon\}$
  - ▶ m. a. W.: das Vorkommen eines Wortes aus  $\langle R \rangle$  ist „optional“
  - ▶ auch dafür verschiedene Abkürzungen (je nach Anwendung)  
z. B.  $(R)?$  bzw.  $R?$  oder  $[R]$  oder ...



## Reguläre Ausdrücke

Definition

Beschriebene formale Sprache

Beispiel: Datums-/Zeitangaben in Emails

Zusammenhang mit Automaten und Grammatiken

## Rechtslineare Grammatiken (Typ 3)

## Kantorowitsch-Bäume und strukturelle Induktion

- ▶ siehe z. B. <http://tools.ietf.org/html/rfc5322>
- ▶ legt fest, was wo in einer Email stehen muss, soll, darf oder auch nicht ...
- ▶ benutzt dazu etwas namens ABNF
  - ▶ „augmented Backus Naur form“
  - ▶ seinerseits spezifiziert in RFC 5234
  - ▶ für nachfolgendes Beispiel: im wesentlichen reguläre Ausdrücke
  - ▶ im allgemeinen deutlich mächtiger
- ▶ Beispiel: Datums- und Zeitangaben in Emails
  - ▶ Date: Wed, 12 Jan 2011 15:13:47 +0100

```

date-time      = [ day-of-week "," ] date time [CFWS]
day-of-week    = ([FWS] day-name)
day-name       = "Mon" / "Tue" / "Wed" / "Thu" /
                  "Fri" / "Sat" / "Sun"
date           = day month year
day            = ([FWS] 1*2DIGIT FWS)
month          = "Jan" / "Feb" / "Mar" / "Apr" /
                  "May" / "Jun" / "Jul" / "Aug" /
                  "Sep" / "Oct" / "Nov" / "Dec"
year           = (FWS 4*DIGIT FWS)
time           = time-of-day zone
time-of-day    = hour ":" minute [ ":" second ]
hour           = 2DIGIT
minute         = 2DIGIT
second         = 2DIGIT
zone           = (FWS ( "+" / "-" ) 4DIGIT)

```

- ▶ Beispiel

```
time-of-day = hour ":" minute [ ":" second ]  
hour         = 2DIGIT  
minute       = 2DIGIT  
second       = 2DIGIT
```

- ▶ in jeder Zeile

- ▶ links vom = ein Name für einen regulären Ausdruck
- ▶ rechts vom = etwas wie ein regulärer Ausdruck,  
der statt Teilausdrücken deren Namen benutzen kann

- ▶ an anderer Stelle definiert:

$\langle \text{DIGIT} \rangle = 0|1|2|3|4|5|6|7|8|9$ , also

$\langle \text{DIGIT} \rangle = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

- ▶ 2DIGIT ist Abkürzung für  $\langle \text{DIGIT} \rangle \langle \text{DIGIT} \rangle$ , also z. B.

$\langle \text{2DIGIT} \rangle = \{00, 01, 02, \dots, 99\}$

- ▶ also auch

$\langle \text{hour} \rangle = \langle \text{minute} \rangle = \langle \text{second} \rangle = \{00, 01, 02, \dots, 99\}$

- ▶ Beispiel

```
time-of-day = hour ":" minute [ ":" second ]  
hour        = 2DIGIT  
minute      = 2DIGIT  
second      = 2DIGIT
```

- ▶ Wörter in Anführungszeichen stehen für sich
- ▶ eckige Klammern bedeuten, dass ein Teil optional ist
- ▶ also

`time-of-day` = `hour` : `minute` | `hour` : `minute` : `second`

- ▶ Beispiele: 09:50 oder 09:50:17
- ▶ beachte:
  - ▶ 9:50 ist nicht syntaktisch korrekt
  - ▶ aber 39:71 ist syntaktisch korrekt

## Reguläre Ausdrücke

Definition

Beschriebene formale Sprache

Beispiel: Datums-/Zeitangaben in Emails

Zusammenhang mit Automaten und Grammatiken

## Rechtslineare Grammatiken (Typ 3)

## Kantorowitsch-Bäume und strukturelle Induktion

## Satz

Für jede formale Sprache  $L$  sind die folgenden drei Aussagen äquivalent:

1.  $L$  kann von einem endlichen Akzeptor erkannt werden.
  2.  $L$  kann durch einen regulären Ausdruck beschrieben werden.
  3.  $L$  kann von einer rechtslinearen Grammatik erzeugt werden.
- 
- ▶ rechtslineare Grammatiken kommen gleich
  - ▶ Eine formale Sprache, die die Eigenschaften des Satzes hat, heißt *reguläre Sprache*.
  - ▶ Jede rechtslineare Grammatik ist eine kontextfreie Grammatik, also ist jede reguläre Sprache eine kontextfreie Sprache,
  - ▶ *aber nicht umgekehrt*, wie man z. B. an  $\{a^k b^k \mid k \in \mathbb{N}_0\}$  sieht.

## Satz

Für jede formale Sprache  $L$  sind die folgenden drei Aussagen äquivalent:

1.  $L$  kann von einem endlichen Akzeptor erkannt werden.
  2.  $L$  kann durch einen regulären Ausdruck beschrieben werden.
  3.  $L$  kann von einer rechtslinearen Grammatik erzeugt werden.
- 
- ▶ rechtslineare Grammatiken kommen gleich
  - ▶ Eine formale Sprache, die die Eigenschaften des Satzes hat, heißt *reguläre Sprache*.
  - ▶ Jede rechtslineare Grammatik ist eine kontextfreie Grammatik, also ist jede reguläre Sprache eine kontextfreie Sprache,
  - ▶ *aber nicht umgekehrt*, wie man z. B. an  $\{a^k b^k \mid k \in \mathbb{N}_0\}$  sieht.



konstruktiv:

- ▶ von endlichem Akzeptor  $A$  zu regulärem Ausdruck  $R$  mit  $L(A) = \langle R \rangle$ 
  - ▶ „mittel schwer“, z. B. inspiriert vom Algorithmus von Warshall
- ▶ von regulärem Ausdruck  $R$  zu rechtslinearer Grammatik  $G$  mit  $\langle R \rangle = L(G)$ :
  - ▶ „relativ leicht“
- ▶ von rechtslinearer Grammatik  $G$  zu endlichem Akzeptor  $A$  mit  $L(G) = L(A)$ :
  - ▶ „am schwierigsten“
- ▶ **beachte:** für die umgekehrten Richtungen braucht man keine zusätzlichen expliziten Konstruktionen

konstruktiv:

- ▶ von endlichem Akzeptor  $A$  zu regulärem Ausdruck  $R$  mit  $L(A) = \langle R \rangle$ 
  - ▶ „mittel schwer“, z. B. inspiriert vom Algorithmus von Warshall
- ▶ von regulärem Ausdruck  $R$  zu rechtslinearer Grammatik  $G$  mit  $\langle R \rangle = L(G)$ :
  - ▶ „relativ leicht“
- ▶ von rechtslinearer Grammatik  $G$  zu endlichem Akzeptor  $A$  mit  $L(G) = L(A)$ :
  - ▶ „am schwierigsten“
- ▶ **beachte:** für die umgekehrten Richtungen braucht man keine zusätzlichen expliziten Konstruktionen

konstruktiv:

- ▶ von endlichem Akzeptor  $A$  zu regulärem Ausdruck  $R$  mit  $L(A) = \langle R \rangle$ 
  - ▶ „mittel schwer“, z. B. inspiriert vom Algorithmus von Warshall
- ▶ von regulärem Ausdruck  $R$  zu rechtslinearer Grammatik  $G$  mit  $\langle R \rangle = L(G)$ :
  - ▶ „relativ leicht“
- ▶ von rechtslinearer Grammatik  $G$  zu endlichem Akzeptor  $A$  mit  $L(G) = L(A)$ :
  - ▶ „am schwierigsten“
- ▶ **beachte:** für die umgekehrten Richtungen braucht man keine zusätzlichen expliziten Konstruktionen

konstruktiv:

- ▶ von endlichem Akzeptor  $A$  zu regulärem Ausdruck  $R$  mit  $L(A) = \langle R \rangle$ 
  - ▶ „mittel schwer“, z. B. inspiriert vom Algorithmus von Warshall
- ▶ von regulärem Ausdruck  $R$  zu rechtslinearer Grammatik  $G$  mit  $\langle R \rangle = L(G)$ :
  - ▶ „relativ leicht“
- ▶ von rechtslinearer Grammatik  $G$  zu endlichem Akzeptor  $A$  mit  $L(G) = L(A)$ :
  - ▶ „am schwierigsten“
- ▶ **beachte:** für die umgekehrten Richtungen braucht man keine zusätzlichen expliziten Konstruktionen

## Das sollten Sie mitnehmen:

- ▶ Definition „klassischer“ regulärer Ausdrücke
  - ▶ atomare:
    - ▶  $\emptyset$
    - ▶  $a \in A$
  - ▶ zusammengesetzte:
    - ▶  $(R_1 | R_2)$
    - ▶  $(R_1 R_2)$
    - ▶  $(R)^*$
- ▶ wissen: reguläre Ausdrücke und die Verallgemeinerung **Regular Expressions** sind z. B. bei Textverarbeitungsaufgaben manchmal nützlich

## Das sollten Sie üben:

- ▶ zu  $L$  ein  $R$  mit  $\langle R \rangle = L$  konstruieren
- ▶ zu  $R$  das  $\langle R \rangle$  bestimmen

## Reguläre Ausdrücke

- Definition

- Beschriebene formale Sprache

- Beispiel: Datums-/Zeitangaben in Emails

- Zusammenhang mit Automaten und Grammatiken

## Rechtslineare Grammatiken (Typ 3)

Kantorowitsch-Bäume und strukturelle Induktion

- ▶ Mit kontextfreien Grammatiken kann man jedenfalls zum Teil andere formale Sprachen erzeugen, als man mit endlichen Akzeptoren erkennen kann.
- ▶ Beispiel:
  - ▶  $G = (\{X\}, \{a, b\}, X, \{X \rightarrow aXb \mid \varepsilon\})$  erzeugt  $\{a^k b^k \mid k \in \mathbb{N}_0\}$
  - ▶ und diese Sprache ist nicht regulär.
- ▶ Kann man kontextfreie Grammatiken so einschränken, dass sie zu endlichen Akzeptoren passen?

- ▶ Eine *rechtslineare Grammatik* ist eine kontextfreie Grammatik  $G = (N, T, S, P)$ , die der folgenden Einschränkung genügt:  
Jede Produktion ist
  - ▶ entweder von der Form  $X \rightarrow w$  mit  $w \in T^*$
  - ▶ oder von der Form  $X \rightarrow wY$  mit  $w \in T^*$  und  $X, Y \in N$ .
- ▶ also auf jeder rechten Seite
  - ▶ höchstens ein Nichtterminalsymbol
  - ▶ und wenn dann nur als letztes Symbol



- ▶  $G = (\{X\}, \{a, b\}, X, \{X \rightarrow abX \mid bbaX \mid \varepsilon\})$

$$L(G) = \langle (ab \mid bba)^* \rangle$$

- ▶  $G = (\{X, Y\}, \{a, b\}, X, \{X \rightarrow aX \mid bX \mid ababbY, Y \rightarrow aY \mid bY \mid \varepsilon\})$

$$L(G) = \langle (a \mid b)^* ababb (a \mid b)^* \rangle$$

- ▶  $G = (\{X\}, \{a, b\}, X, \{X \rightarrow abX \mid bbaX \mid \varepsilon\})$

$$L(G) = \langle (ab|bba)^* \rangle$$

- ▶  $G = (\{X, Y\}, \{a, b\}, X, \{X \rightarrow aX \mid bX \mid ababbY, Y \rightarrow aY \mid bY \mid \varepsilon\})$

$$L(G) = \langle (a|b)^* ababb (a|b)^* \rangle$$

- ▶  $G = (\{X\}, \{a, b\}, X, \{X \rightarrow abX \mid bbaX \mid \varepsilon\})$

$$L(G) = \langle (ab|bba)^* \rangle$$

- ▶  $G = (\{X, Y\}, \{a, b\}, X, \{X \rightarrow aX \mid bX \mid ababbY, Y \rightarrow aY \mid bY \mid \varepsilon\})$

$$L(G) = \langle (a|b)^* ababb (a|b)^* \rangle$$

- ▶  $G = (\{X\}, \{a, b\}, X, \{X \rightarrow abX \mid bbaX \mid \varepsilon\})$

$$L(G) = \langle (ab|bba)^* \rangle$$

- ▶  $G = (\{X, Y\}, \{a, b\}, X, \{X \rightarrow aX \mid bX \mid ababbY, Y \rightarrow aY \mid bY \mid \varepsilon\})$

$$L(G) = \langle (a|b)^* ababb (a|b)^* \rangle$$

- ▶  $G = (\{X\}, \{a, b\}, X, \{X \rightarrow aXb \mid \varepsilon\})$  ist **nicht** rechtslinear,
  - ▶ denn in  $X \rightarrow aXb$  steht das Nichtterminalsymbol  $X$  nicht am rechten Ende.
- ▶ Da die erzeugte formale Sprache  $\{a^k b^k \mid k \in \mathbb{N}_0\}$  von keinem endlichen Akzeptor erkannt wird, kann es auch gar keine rechtslineare Grammatik geben.

- ▶ Rechtslineare Grammatiken heißen auch *Typ-3-Grammatiken*.
- ▶ Kontextfreien Grammatiken heißen auch *Typ-2-Grammatiken*.
- ▶ Man ahnt schon: Es gibt auch noch
  - ▶ *Typ-1-Grammatiken* und
  - ▶ *Typ-0-Grammatiken*,die wir hier nicht weiter betrachten werden.
- ▶ Wenn für ein  $i \in \{0, 1, 2, 3\}$  eine formale Sprache  $L$  von einer Typ- $i$ -Grammatik erzeugt wird, dann sagt man auch,  $L$  sei eine *Typ- $i$ -Sprache* oder kurz *vom Typ  $i$* .

- ▶ Rechtslineare Grammatiken heißen auch *Typ-3-Grammatiken*.
- ▶ Kontextfreien Grammatiken heißen auch *Typ-2-Grammatiken*.
- ▶ Man ahnt schon: Es gibt auch noch
  - ▶ *Typ-1-Grammatiken* und
  - ▶ *Typ-0-Grammatiken*,die wir hier nicht weiter betrachten werden.
- ▶ Wenn für ein  $i \in \{0, 1, 2, 3\}$  eine formale Sprache  $L$  von einer Typ- $i$ -Grammatik erzeugt wird, dann sagt man auch,  $L$  sei eine *Typ- $i$ -Sprache* oder kurz *vom Typ  $i$* .

- ▶ Rechtslineare Grammatiken heißen auch *Typ-3-Grammatiken*.
- ▶ Kontextfreien Grammatiken heißen auch *Typ-2-Grammatiken*.
- ▶ Man ahnt schon: Es gibt auch noch
  - ▶ *Typ-1-Grammatiken* und
  - ▶ *Typ-0-Grammatiken*,die wir hier nicht weiter betrachten werden.
- ▶ Wenn für ein  $i \in \{0, 1, 2, 3\}$  eine formale Sprache  $L$  von einer Typ- $i$ -Grammatik erzeugt wird, dann sagt man auch,  $L$  sei eine *Typ- $i$ -Sprache* oder kurz *vom Typ  $i$* .



- ▶ Rechtslineare Grammatiken heißen auch *Typ-3-Grammatiken*.
- ▶ Kontextfreien Grammatiken heißen auch *Typ-2-Grammatiken*.
- ▶ Man ahnt schon: Es gibt auch noch
  - ▶ *Typ-1-Grammatiken* und
  - ▶ *Typ-0-Grammatiken*,die wir hier nicht weiter betrachten werden.
- ▶ Wenn für ein  $i \in \{0, 1, 2, 3\}$  eine formale Sprache  $L$  von einer Typ- $i$ -Grammatik erzeugt wird, dann sagt man auch,  $L$  sei eine *Typ- $i$ -Sprache* oder kurz *vom Typ  $i$* .

- ▶ Wozu rechtslineare Grammatiken?
- ▶ gegenüber deterministischen endlichen Akzeptoren:  
manchmal deutlich kürzer und übersichtlicher hinschreiben
- ▶ genaueres Verständnis dafür: im 3. Semester bei nichtdeterministischen endlichen Akzeptoren

- ▶ Wozu rechtslineare Grammatiken?
- ▶ gegenüber deterministischen endlichen Akzeptoren:  
manchmal deutlich kürzer und übersichtlicher hinschreiben
- ▶ genaueres Verständnis dafür: im 3. Semester bei nichtdeterministischen endlichen Akzeptoren

## Reguläre Ausdrücke

- Definition

- Beschriebene formale Sprache

- Beispiel: Datums-/Zeitangaben in Emails

- Zusammenhang mit Automaten und Grammatiken

## Rechtslineare Grammatiken (Typ 3)

## Kantorowitsch-Bäume und strukturelle Induktion

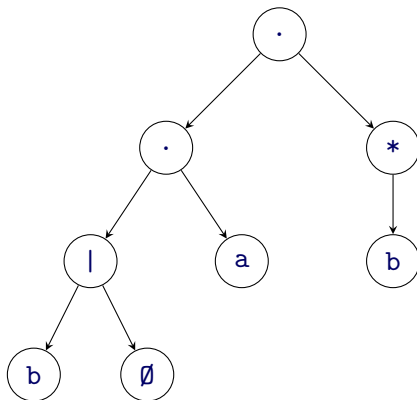
Beweisskizze für das folgende

**Lemma.**

Zu jedem regulären Ausdruck  $R$  gibt es eine rechtslineare Grammatik  $G$  mit  $L(G) = \langle R \rangle$ .

- ▶ Wie beweist man, dass eine Aussage für alle regulären Ausdrücke gilt?
- ▶ eine Möglichkeit: *strukturelle Induktion*
  - ▶ Variante/Verallgemeinerung vollständiger Induktion,
  - ▶ ohne explizit über natürliche Zahlen zu sprechen
- ▶ darauf arbeiten wir jetzt in mehreren Schritten hin:
  - ▶ Darstellung regulärer Ausdrücke mit Bäumen
  - ▶ eine Variante „normaler vollständiger Induktion“
  - ▶ strukturelle Induktion

- ▶ regulärer Ausdruck:  $((b|\emptyset)a)(b^*)$
- ▶ Darstellung als sogenannter Kantorowitsch-Baum



- ▶ **Beachte:**
  - ▶ das ist *nicht* der Ableitungsbaum gemäß einer Grammatik
  - ▶ aber „genauso gut“ und kompakter

Es sei  $A$  irgendein Alphabet.

Ein Baum ist ein Regex-Baum, wenn gilt:

- ▶ Entweder ist es Baum dessen Wurzel zugleich Blatt ist und das ist mit einem  $x \in A$  oder  $\emptyset$  beschriftet,
- ▶ oder es ist ein Baum, dessen Wurzel mit  $*$  beschriftet ist und die genau einen Nachfolgeknoten hat, der Wurzel eines Regex-Baumes ist
- ▶ oder es ist ein Baum, dessen Wurzel mit  $\cdot$  oder mit  $|$  beschriftet ist und die genau zwei Nachfolgeknoten hat, die Wurzeln zweier Regex-Bäume sind.

Beachte:

- ▶ Linker und rechter Unter-Regex-Baum können unterschiedliche Höhe haben.

- ▶ Größere Bäume sind „aus kleineren zusammengesetzt“, und zwar auf eindeutige Weise.
- ▶ Bijektion Regex-Bäume  $\leftrightarrow$  reguläre Ausdrücke
- ▶ Die **Höhe**  $h(T)$  eines Baumes ist definiert als

$$h(T) = \begin{cases} 0 & \text{falls die Wurzel Blatt ist} \\ 1 + \max_i h(U_i) & \text{falls die } U_i \text{ alle Unterbäume von } T \text{ sind} \end{cases}$$

- ▶ Beweis einer Aussage für alle regulären Ausdrücke:  
durch Beweis der Aussage für alle Regex-Bäume
- ▶ Beweis einer Aussage für alle Regex-Bäume:  
durch vollständige Induktion über die Höhe der Regex-Bäume



- ▶ Größere Bäume sind „aus kleineren zusammengesetzt“, und zwar auf eindeutige Weise.
- ▶ Bijektion Regex-Bäume  $\leftrightarrow$  reguläre Ausdrücke
- ▶ Die **Höhe**  $h(T)$  eines Baumes ist definiert als

$$h(T) = \begin{cases} 0 & \text{falls die Wurzel Blatt ist} \\ 1 + \max_i h(U_i) & \text{falls die } U_i \text{ alle Unterbäume von } T \text{ sind} \end{cases}$$

- ▶ Beweis einer Aussage für alle regulären Ausdrücke:  
durch Beweis der Aussage für alle Regex-Bäume
- ▶ Beweis einer Aussage für alle Regex-Bäume:  
durch vollständige Induktion über die Höhe der Regex-Bäume

- ▶ Größere Bäume sind „aus kleineren zusammengesetzt“, und zwar auf eindeutige Weise.
- ▶ Bijektion Regex-Bäume  $\leftrightarrow$  reguläre Ausdrücke
- ▶ Die **Höhe**  $h(T)$  eines Baumes ist definiert als

$$h(T) = \begin{cases} 0 & \text{falls die Wurzel Blatt ist} \\ 1 + \max_i h(U_i) & \text{falls die } U_i \text{ alle Unterbäume von } T \text{ sind} \end{cases}$$

- ▶ Beweis einer Aussage für alle regulären Ausdrücke:  
durch Beweis der Aussage für alle Regex-Bäume
- ▶ Beweis einer Aussage für alle Regex-Bäume:  
durch vollständige Induktion über die Höhe der Regex-Bäume

- ▶ Größere Bäume sind „aus kleineren zusammengesetzt“, und zwar auf eindeutige Weise.
- ▶ Bijektion Regex-Bäume  $\leftrightarrow$  reguläre Ausdrücke
- ▶ Die **Höhe**  $h(T)$  eines Baumes ist definiert als

$$h(T) = \begin{cases} 0 & \text{falls die Wurzel Blatt ist} \\ 1 + \max_i h(U_i) & \text{falls die } U_i \text{ alle Unterbäume von } T \text{ sind} \end{cases}$$

- ▶ Beweis einer Aussage für alle regulären Ausdrücke:  
durch Beweis der Aussage für alle Regex-Bäume
- ▶ Beweis einer Aussage für alle Regex-Bäume:  
durch vollständige Induktion über die Höhe der Regex-Bäume

- ▶ Größere Bäume sind „aus kleineren zusammengesetzt“, und zwar auf eindeutige Weise.
- ▶ Bijektion Regex-Bäume  $\leftrightarrow$  reguläre Ausdrücke
- ▶ Die **Höhe**  $h(T)$  eines Baumes ist definiert als

$$h(T) = \begin{cases} 0 & \text{falls die Wurzel Blatt ist} \\ 1 + \max_i h(U_i) & \text{falls die } U_i \text{ alle Unterbäume von } T \text{ sind} \end{cases}$$

- ▶ Beweis einer Aussage für alle regulären Ausdrücke:  
durch Beweis der Aussage für alle Regex-Bäume
- ▶ Beweis einer Aussage für alle Regex-Bäume:  
durch vollständige Induktion über die Höhe der Regex-Bäume

- ▶ **naive Vorgehensweise:**  
beim Schritt zu Bäumen der Höhe  $n + 1$   
Induktionsvoraussetzung nur für Bäume der Höhe  $n$
- ▶ **aber:**  
die Unterbäume eines Baumes der Höhe  $n + 1$  können beliebige Höhen  $i \leq n$  haben.
- ▶ anschaulich: man darf auch für die kleineren Unterbäume so etwas wie die Induktionsvoraussetzung benutzen.
- ▶ präzise?

- ▶ Es sei  $\mathcal{B}(n)$  eine Aussage, die von einer Zahl  $n \in \mathbb{N}_0$  abhängt.
- ▶ wollen beweisen:  $\forall n \in \mathbb{N}_0 : \mathcal{B}(n)$
- ▶ definiere Aussage  $\mathcal{A}(n)$  als  $\forall i \leq n : \mathcal{B}(i)$ .
- ▶ beweise  $\forall n \in \mathbb{N}_0 : \mathcal{A}(n)$ :  
das reicht, denn aus  $\mathcal{A}(n)$  folgt  $\mathcal{B}(n)$
- ▶ Induktionsbeweis für  $\forall n \in \mathbb{N}_0 : \mathcal{A}(n)$ :
  - Induktionsanfang  $n = 0$ : Man muss zeigen:  
 $\mathcal{A}(0)$ , also  $\forall i \leq 0 : \mathcal{B}(i)$ , also  $\mathcal{B}(0)$ .
  - Induktionsvoraussetzung: es gilt  $\mathcal{A}(n)$ , also  $\forall i \leq n : \mathcal{B}(i)$ ,
  - Induktionsschluss: zu zeigen: es gilt  $\mathcal{A}(n+1)$  also  
 $\forall i \leq n+1 : \mathcal{B}(i)$ ,  
das ist aber nichts anderes als:  $(\forall i \leq n : \mathcal{B}(i)) \wedge \mathcal{B}(n+1)$ 
    - $\forall i \leq n : \mathcal{B}(i)$ : gilt nach Induktionsvoraussetzung
    - $\mathcal{B}(n+1)$ : hier muss man was tun,  
*aber man kann  $\forall i \leq n : \mathcal{B}(i)$  benutzen*

- ▶ Es sei  $\mathcal{B}(n)$  eine Aussage, die von einer Zahl  $n \in \mathbb{N}_0$  abhängt.
- ▶ wollen beweisen:  $\forall n \in \mathbb{N}_0 : \mathcal{B}(n)$
- ▶ definiere Aussage  $\mathcal{A}(n)$  als  $\forall i \leq n : \mathcal{B}(i)$ .
- ▶ beweise  $\forall n \in \mathbb{N}_0 : \mathcal{A}(n)$ :  
das reicht, denn aus  $\mathcal{A}(n)$  folgt  $\mathcal{B}(n)$
- ▶ Induktionsbeweis für  $\forall n \in \mathbb{N}_0 : \mathcal{A}(n)$ :
  - Induktionsanfang**  $n = 0$ : Man muss zeigen:  
 $\mathcal{A}(0)$ , also  $\forall i \leq 0 : \mathcal{B}(i)$ , also  $\mathcal{B}(0)$ .
  - Induktionsvoraussetzung**: es gilt  $\mathcal{A}(n)$ , also  $\forall i \leq n : \mathcal{B}(i)$ ,
  - Induktionsschluss**: zu zeigen: es gilt  $\mathcal{A}(n+1)$  also  
 $\forall i \leq n+1 : \mathcal{B}(i)$ ,  
das ist aber nichts anderes als:  $(\forall i \leq n : \mathcal{B}(i)) \wedge \mathcal{B}(n+1)$ 
    - $\forall i \leq n : \mathcal{B}(i)$ : gilt nach Induktionsvoraussetzung
    - $\mathcal{B}(n+1)$ : hier muss man was tun,  
*aber man kann  $\forall i \leq n : \mathcal{B}(i)$  benutzen*

Aussage  $\mathcal{B}(n)$ :

*Für jeden Regex-Baum  $R$  der Höhe  $n$  gibt es eine rechtslineare Grammatik  $G$  mit  $\langle R \rangle = L(G)$ .*

- ▶ **Induktionsanfang:** zeige  $\mathcal{B}(0)$ :  
finde T3G, die  $\{x\} = \langle x \rangle$  für  $x \in A$  und  $\{\} = \langle \emptyset \rangle$  erzeugen.
- ▶ **Induktionsvoraussetzung:** f.e.b.a.f.  $n \in \mathbb{N}_0$  gelte  $\forall i \leq n : \mathcal{B}(i)$ ,  
d. h. für jeden Regex-Baum  $R'$  mit einer Höhe  $i \leq n$  gibt es eine T3G  $G$  mit  $\langle R' \rangle = L(G)$ .
- ▶ **Induktionsschluss:** zeige:  $\mathcal{B}(n+1)$  gilt  
d. h. für jeden Regex-Baum  $R$  der Höhe  $n+1$  existiert T3G  $G$  mit  $\langle R \rangle = L(G)$ .



Aussage  $\mathcal{B}(n)$ :

*Für jeden Regex-Baum  $R$  der Höhe  $n$  gibt es eine rechtslineare Grammatik  $G$  mit  $\langle R \rangle = L(G)$ .*

- ▶ **Induktionsanfang:** zeige  $\mathcal{B}(0)$ :  
finde T3G, die  $\{x\} = \langle x \rangle$  für  $x \in A$  und  $\{\} = \langle \emptyset \rangle$  erzeugen.
- ▶ **Induktionsvoraussetzung:** f.e.b.a.f.  $n \in \mathbb{N}_0$  gelte  $\forall i \leq n : \mathcal{B}(i)$ ,  
d. h. für jeden Regex-Baum  $R'$  mit einer Höhe  $i \leq n$  gibt es eine T3G  $G$  mit  $\langle R' \rangle = L(G)$ .
- ▶ **Induktionsschluss:** zeige:  $\mathcal{B}(n+1)$  gilt  
d. h. für jeden Regex-Baum  $R$  der Höhe  $n+1$  existiert T3G  $G$  mit  $\langle R \rangle = L(G)$ .

Aussage  $\mathcal{B}(n)$ :

*Für jeden Regex-Baum  $R$  der Höhe  $n$  gibt es eine rechtslineare Grammatik  $G$  mit  $\langle R \rangle = L(G)$ .*

- ▶ **Induktionsanfang:** zeige  $\mathcal{B}(0)$ :  
finde T3G, die  $\{x\} = \langle x \rangle$  für  $x \in A$  und  $\{\} = \langle \emptyset \rangle$  erzeugen.
- ▶ **Induktionsvoraussetzung:** f.e.b.a.f.  $n \in \mathbb{N}_0$  gelte  $\forall i \leq n : \mathcal{B}(i)$ ,  
d. h. für jeden Regex-Baum  $R'$  mit einer Höhe  $i \leq n$  gibt es eine T3G  $G$  mit  $\langle R' \rangle = L(G)$ .
- ▶ **Induktionsschluss:** zeige:  $\mathcal{B}(n+1)$  gilt  
d. h. für jeden Regex-Baum  $R$  der Höhe  $n+1$  existiert T3G  $G$  mit  $\langle R \rangle = L(G)$ .

Aussage  $\mathcal{B}(n)$ :

*Für jeden Regex-Baum  $R$  der Höhe  $n$  gibt es eine rechtslineare Grammatik  $G$  mit  $\langle R \rangle = L(G)$ .*

- ▶ **Induktionsanfang:** zeige  $\mathcal{B}(0)$ :  
finde T3G, die  $\{x\} = \langle x \rangle$  für  $x \in A$  und  $\{\} = \langle \emptyset \rangle$  erzeugen.
- ▶ **Induktionsvoraussetzung:** f.e.b.a.f.  $n \in \mathbb{N}_0$  gelte  $\forall i \leq n : \mathcal{B}(i)$ ,  
d. h. für jeden Regex-Baum  $R'$  mit einer Höhe  $i \leq n$  gibt es eine T3G  $G$  mit  $\langle R' \rangle = L(G)$ .
- ▶ **Induktionsschluss:** zeige:  $\mathcal{B}(n+1)$  gilt  
d. h. für jeden Regex-Baum  $R$  der Höhe  $n+1$  existiert T3G  $G$  mit  $\langle R \rangle = L(G)$ .

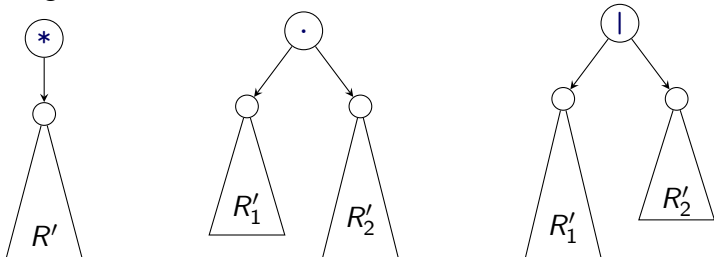
Aussage  $\mathcal{B}(n)$ :

*Für jeden Regex-Baum  $R$  der Höhe  $n$  gibt es eine rechtslineare Grammatik  $G$  mit  $\langle R \rangle = L(G)$ .*

- ▶ **Induktionsanfang:** zeige  $\mathcal{B}(0)$ :  
finde T3G, die  $\{x\} = \langle x \rangle$  für  $x \in A$  und  $\{\} = \langle \emptyset \rangle$  erzeugen.
- ▶ **Induktionsvoraussetzung:** f.e.b.a.f.  $n \in \mathbb{N}_0$  gelte  $\forall i \leq n : \mathcal{B}(i)$ ,  
d. h. für jeden Regex-Baum  $R'$  mit einer Höhe  $i \leq n$  gibt es eine T3G  $G$  mit  $\langle R' \rangle = L(G)$ .
- ▶ **Induktionsschluss:** zeige:  $\mathcal{B}(n+1)$  gilt  
d. h. für jeden Regex-Baum  $R$  der Höhe  $n+1$  existiert T3G  $G$  mit  $\langle R \rangle = L(G)$ .

# Skizze des Induktionsschritts (1)

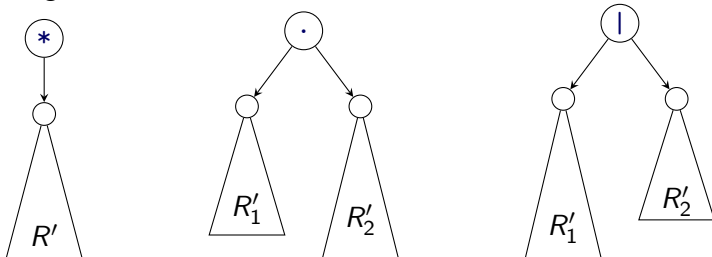
- ▶ sei  $R$  beliebiger Regex-Baum der Höhe  $n + 1$   
mögliche Fälle:



- ▶ Induktionsvoraussetzung: für alle Unterbäume gibt es T3G
- ▶ Aus T3G für Unterbäume kann man T3G für ganzen Regex-Baum konstruieren.

# Skizze des Induktionsschritts (1)

- ▶ sei  $R$  beliebiger Regex-Baum der Höhe  $n + 1$   
mögliche Fälle:



- ▶ Induktionsvoraussetzung: für alle Unterbäume gibt es T3G
- ▶ Aus T3G für Unterbäume kann man T3G für ganzen Regex-Baum konstruieren.

## Skizze des Induktionsschritts (2)

- ▶ hier nicht alle Details
- ▶ Beispiel :  $R$  ist  $R_1 \mid R_2$
- ▶ nach Induktionsvoraussetzung gibt es T3G  
 $G_1 = (N_1, A, S_1, P_1)$  und  $G_2 = (N_2, A, S_2, P_2)$   
mit  $L(G_1) = \langle R_1 \rangle$  bzw.  $L(G_2) = \langle R_2 \rangle$
- ▶ ohne Beschränkung der Allgemeinheit  $N_1 \cap N_2 = \emptyset$
- ▶ wähle „neues“ Nichtterminalsymbol  $S \notin N_1 \cup N_2$
- ▶ Behauptung:

$$G = (\{S\} \cup N_1 \cup N_2, A, S, \{S \rightarrow S_1 \mid S_2\} \cup P_1 \cup P_2)$$

ist T3G mit  $L(G) = \langle R_1 \mid R_2 \rangle$

- ▶  $G$  ist rechtslinear: leicht
- ▶  $L(G) = L(G_1) \cup L(G_2)$ : macht Arbeit, aber nicht hier

## Skizze des Induktionsschritts (2)

- ▶ hier nicht alle Details
- ▶ Beispiel :  $R$  ist  $R_1 \mid R_2$
- ▶ nach Induktionsvoraussetzung gibt es T3G  
 $G_1 = (N_1, A, S_1, P_1)$  und  $G_2 = (N_2, A, S_2, P_2)$   
mit  $L(G_1) = \langle R_1 \rangle$  bzw.  $L(G_2) = \langle R_2 \rangle$
- ▶ ohne Beschränkung der Allgemeinheit  $N_1 \cap N_2 = \emptyset$
- ▶ wähle „neues“ Nichtterminalsymbol  $S \notin N_1 \cup N_2$
- ▶ Behauptung:

$$G = (\{S\} \cup N_1 \cup N_2, A, S, \{S \rightarrow S_1 \mid S_2\} \cup P_1 \cup P_2)$$

ist T3G mit  $L(G) = \langle R_1 \mid R_2 \rangle$

- ▶  $G$  ist rechtslinear: leicht
- ▶  $L(G) = L(G_1) \cup L(G_2)$ : macht Arbeit, aber nicht hier



## Skizze des Induktionsschritts (2)

- ▶ hier nicht alle Details
- ▶ Beispiel :  $R$  ist  $R_1 \mid R_2$
- ▶ nach Induktionsvoraussetzung gibt es T3G  
 $G_1 = (N_1, A, S_1, P_1)$  und  $G_2 = (N_2, A, S_2, P_2)$   
mit  $L(G_1) = \langle R_1 \rangle$  bzw.  $L(G_2) = \langle R_2 \rangle$
- ▶ ohne Beschränkung der Allgemeinheit  $N_1 \cap N_2 = \emptyset$
- ▶ wähle „neues“ Nichtterminalsymbol  $S \notin N_1 \cup N_2$
- ▶ Behauptung:

$$G = (\{S\} \cup N_1 \cup N_2, A, S, \{S \rightarrow S_1 \mid S_2\} \cup P_1 \cup P_2)$$

ist T3G mit  $L(G) = \langle R_1 \mid R_2 \rangle$

- ▶  $G$  ist rechtslinear: leicht
- ▶  $L(G) = L(G_1) \cup L(G_2)$ : macht Arbeit, aber nicht hier

## Skizze des Induktionsschritts (2)

- ▶ hier nicht alle Details
- ▶ Beispiel :  $R$  ist  $R_1 \mid R_2$
- ▶ nach Induktionsvoraussetzung gibt es T3G  
 $G_1 = (N_1, A, S_1, P_1)$  und  $G_2 = (N_2, A, S_2, P_2)$   
mit  $L(G_1) = \langle R_1 \rangle$  bzw.  $L(G_2) = \langle R_2 \rangle$
- ▶ ohne Beschränkung der Allgemeinheit  $N_1 \cap N_2 = \emptyset$
- ▶ wähle „neues“ Nichtterminalsymbol  $S \notin N_1 \cup N_2$
- ▶ Behauptung:

$$G = (\{S\} \cup N_1 \cup N_2, A, S, \{S \rightarrow S_1 \mid S_2\} \cup P_1 \cup P_2)$$

ist T3G mit  $L(G) = \langle R_1 \mid R_2 \rangle$

- ▶  $G$  ist rechtslinear: leicht
- ▶  $L(G) = L(G_1) \cup L(G_2)$ : macht Arbeit, aber nicht hier

## 1. generelle Situation:

irgendwelche „Gebilde“ (eben reguläre Ausdrücke). Es gibt

- ▶ kleinste „atomare“ oder „elementare“ Gebilde (eben reguläre Ausdrücke  $x$  für  $x \in A$  und  $\emptyset$ ) und
- ▶ eine oder mehrere Konstruktionsvorschriften, nach denen man aus kleineren Gebilden größere bauen kann (eben  $*$ ,  $|$ ,  $\cdot$ ).

## 2. Aufgabe:

zeige, dass alle Gebilde eine gewisse Eigenschaft haben.

**Strukturelle Induktion:**

- ▶ **Induktionsanfang:** zeige, dass *alle* „atomaren“ Gebilde die Eigenschaft haben
- ▶ **Induktionsschritt:** zeige, wie bei einem „großen“ Gebilde die Eigenschaft daraus folgt, dass schon alle Untergebilde die Eigenschaft haben, egal welche Konstruktionsvorschrift benutzt wurde.

## 1. generelle Situation:

irgendwelche „Gebilde“ (eben reguläre Ausdrücke). Es gibt

- ▶ kleinste „atomare“ oder „elementare“ Gebilde (eben reguläre Ausdrücke  $x$  für  $x \in A$  und  $\emptyset$ ) und
- ▶ eine oder mehrere Konstruktionsvorschriften, nach denen man aus kleineren Gebilden größere bauen kann (eben  $*$ ,  $|$ ,  $\cdot$ ).

## 2. Aufgabe:

zeige, dass alle Gebilde eine gewisse Eigenschaft haben.

**Strukturelle Induktion:**

- ▶ **Induktionsanfang:** zeige, dass *alle* „atomaren“ Gebilde die Eigenschaft haben
- ▶ **Induktionsschritt:** zeige, wie bei einem „großen“ Gebilde die Eigenschaft daraus folgt, dass schon alle Untergebilde die Eigenschaft haben, egal welche Konstruktionsvorschrift benutzt wurde.

## 1. generelle Situation:

irgendwelche „Gebilde“ (eben reguläre Ausdrücke). Es gibt

- ▶ kleinste „atomare“ oder „elementare“ Gebilde (eben reguläre Ausdrücke  $x$  für  $x \in A$  und  $\emptyset$ ) und
- ▶ eine oder mehrere Konstruktionsvorschriften, nach denen man aus kleineren Gebilden größere bauen kann (eben  $*$ ,  $|$ ,  $\cdot$ ).

## 2. Aufgabe:

zeige, dass alle Gebilde eine gewisse Eigenschaft haben.

**Strukturelle Induktion:**

- ▶ **Induktionsanfang:** zeige, dass *alle* „atomaren“ Gebilde die Eigenschaft haben
- ▶ **Induktionsschritt:** zeige, wie bei einem „großen“ Gebilde die Eigenschaft daraus folgt, dass schon alle Untergebilde die Eigenschaft haben, egal welche Konstruktionsvorschrift benutzt wurde.

Das sollten Sie mitnehmen:

- ▶ Definition **rechtslineare Grammatiken**

Das sollten Sie üben:

- ▶ rechtslineare Grammatiken konstruieren  
(zu gegebenem Akzeptor, regulären Ausdruck, formaler Sprache)
- ▶ **strukturelle Induktion**

- ▶ reguläre Ausdrücke
  - ▶ werden von diversen „Unix-Tools“ genutzt
  - ▶ in manchen Programmiersprachen zur Textverarbeitung praktisch
- ▶ rechtslineare Grammatiken
- ▶ strukturelle Induktion