Grundbegriffe der Informatik

Kapitel 19: Reguläre Ausdrücke und rechtslineare Grammatiken

Thomas Worsch

KIT, Institut für Theoretische Informatik

Wintersemester 2015/2016

Was können endliche Akzeptoren?

manche Sprachen mit EA erkennbar

- z. B. $\{\mathtt{a}\}^+\{\mathtt{b}\} \cup \{\mathtt{b}\}^+\{\mathtt{a}\}$

manche Sprachen *nicht* mit EA erkennbar

- z. B. $\{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\}$

Charakterisierung der erkennbaren Sprachen?

- i. e. Beschreibung ohne Benutzung endlicher Akzeptoren?

Überblick

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

Wo sind wir?

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

Wo sind wir?

Der Begriff regulärer Ausdruck hat heute verschiedene Bedeutungen

Beschreibung formaler Sprachen

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)

Alphabet $Z = \{|, (,), *, \emptyset\}$ von «Hilfssymbolen»

Alphabet A enthalte kein Zeichen aus Z

regulärer Ausdruck über A ist eine Zeichenfolge über dem Alphabet $A \cup Z$, die gewissen Vorschriften genügt.

Menge der regulären Ausdrücke (RA) ist wie folgt festgelegt:

- $\emptyset$ ist RA
- für jedes $x \in A$ ist x RA
- wenn R_1 und R_2 RA sind, dann auch $(R_1|R_2)$ und (R_1R_2)
- wenn R ein RA ist, dann auch $(R*)$
- Nichts anderes sind reguläre Ausdrücke.

Beispiele

sei $A = \{\texttt{a}, \texttt{b}\}$

Beispiele regulärer Ausdrücke:

`∅`	`a`	`b`
`(ab)`	`(∅b)`	`(a∅)`
`((ab)a)`	`(((ab)a)a)`	`((ab)(aa))`
`(∅|b)`	`(a|b)`	`(a|(ba))`
`((a(a|b))|b)`	`(a|(b|(a|a)))`	
`(∅*)`	`(a*)`	`((a*)*)`
`((ba)(b*))`	`(((ba)b)*)`	
`(((((ab)b)*)*)|(∅*))`		

Klammereinsparungsregeln

„Stern- vor Punktrechnung“

„Punkt- vor Strichrechnung“

Beispiel:

- $R_1 \mid R_2 R_3 *$ Kurzform für
- $(R_1 \mid (R_2 (R_3 *)))$

Bei mehreren gleichen binären Operatoren ohne Klammern gilt das als links geklammert

Beispiel

- $R_1 \mid R_2 \mid R_3$ Kurzform für
- $((R_1 \mid R_2) \mid R_3)$

Beispiele für Klammereinsparungsregeln

`abaa` statt `(((ab)a)a)`

`ab(aa)` statt `((ab)(aa))`

`a(a|b)|b` statt `((a(a|b))|b)`

`(abb)**|Ø*` statt `(((((ab)b)*)*)|(Ø*))`

Nichtbeispiele

keine regulären Ausdrücke über {a, b}:

`(|b)`	vor `|` fehlt ein regulärer Ausdruck
`|∅|`	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 der Syntax regulärer Ausdrücke (2)

alternative Definition mit Hilfe einer kontextfreien Grammatik

reguläre Ausdrücke:

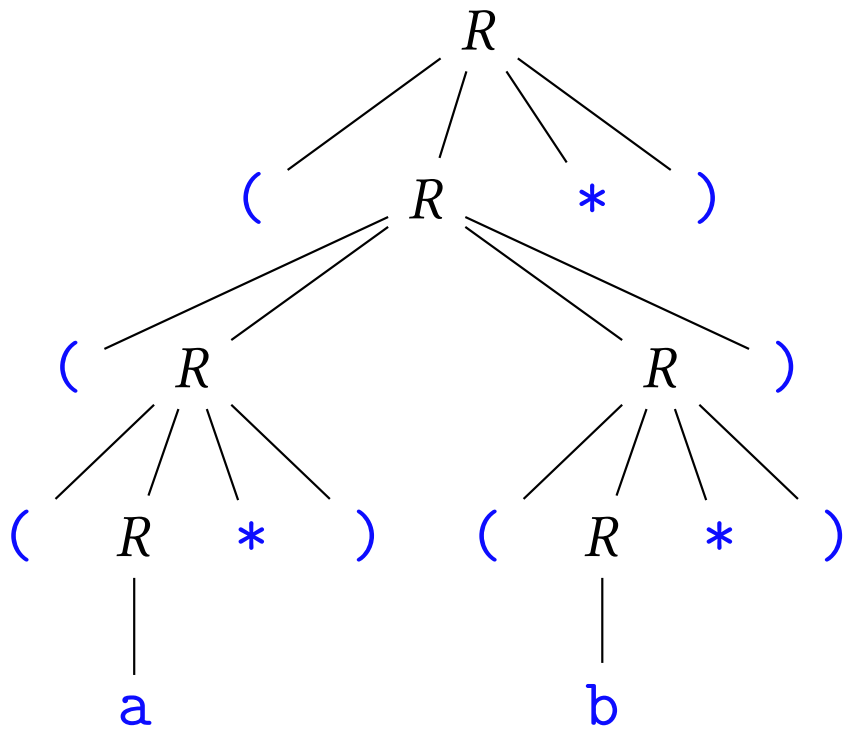
die von der folgenden Grammatik erzeugten Wörter

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

mit
$$\begin{aligned} P = {} & \{R \to \emptyset\} \\ & \cup \{R \to x \mid x \in A\} \\ & \cup \{R \to (R|R),\ \ R \to (RR)\} \\ & \cup \{R \to (R*)\} \end{aligned}$$

vergleiche wohlgeformte Klammerausdrücke (Kap. 11)

Ableitungsbaum eines regulären Ausdrucks



(((a*)(b*))*)

oder kürzer

(a*b*)*

Wo sind wir?

Durch R beschriebene formale Sprache $\langle R \rangle$

- $\langle \emptyset \rangle = \{\}$
- $\langle x \rangle = \{x\}$ für jedes $x \in A$
- $\langle R_1 \mid R_2 \rangle = \langle R_1 \rangle \cup \langle R_2 \rangle$
- $\langle R_1 R_2 \rangle = \langle R_1 \rangle \cdot \langle R_2 \rangle$
- $\langle R* \rangle = \langle R \rangle^*$

Definition folgt der für reguläre Ausdrücke

Beispiele für $\langle R \rangle$

- $R = \mathtt{a} \mid \mathtt{b}$: Dann ist
  $\langle R \rangle = \langle \mathtt{a} \mid \mathtt{b} \rangle = \langle \mathtt{a} \rangle \cup \langle \mathtt{b} \rangle = \{\mathtt{a}\} \cup \{\mathtt{b}\} = \{\mathtt{a}, \mathtt{b}\}$

Beispiele für $\langle R \rangle$

- $R = \texttt{a|b}$: Dann ist
  $\langle R \rangle = \langle \texttt{a|b} \rangle = \langle \texttt{a} \rangle \cup \langle \texttt{b} \rangle = \{\texttt{a}\} \cup \{\texttt{b}\} = \{\texttt{a}, \texttt{b}\}$

- $R = \texttt{(a|b)*}$: Dann ist
  $\langle R \rangle = \langle \texttt{(a|b)*} \rangle = \langle \texttt{a|b} \rangle^* = \{\texttt{a}, \texttt{b}\}^*$

Beispiele für $\langle R \rangle$

- $R = \texttt{a|b}$: Dann ist
  $\langle R \rangle = \langle \texttt{a|b} \rangle = \langle \texttt{a} \rangle \cup \langle \texttt{b} \rangle = \{\texttt{a}\} \cup \{\texttt{b}\} = \{\texttt{a}, \texttt{b}\}$

- $R = \texttt{(a|b)*}$: Dann ist
  $\langle R \rangle = \langle \texttt{(a|b)*} \rangle = \langle \texttt{a|b} \rangle^* = \{\texttt{a}, \texttt{b}\}^*$

- $R = \texttt{(a*b*)*}$: Dann ist
  $\langle R \rangle = \langle \texttt{(a*b*)*} \rangle = \langle \texttt{a*b*} \rangle^*$
  $= (\langle \texttt{a*} \rangle \langle \texttt{b*} \rangle)^* = (\langle \texttt{a} \rangle^* \langle \texttt{b} \rangle^*)^* = (\{\texttt{a}\}^* \{\texttt{b}\}^*)^*$

Beispiele für $\langle R \rangle$

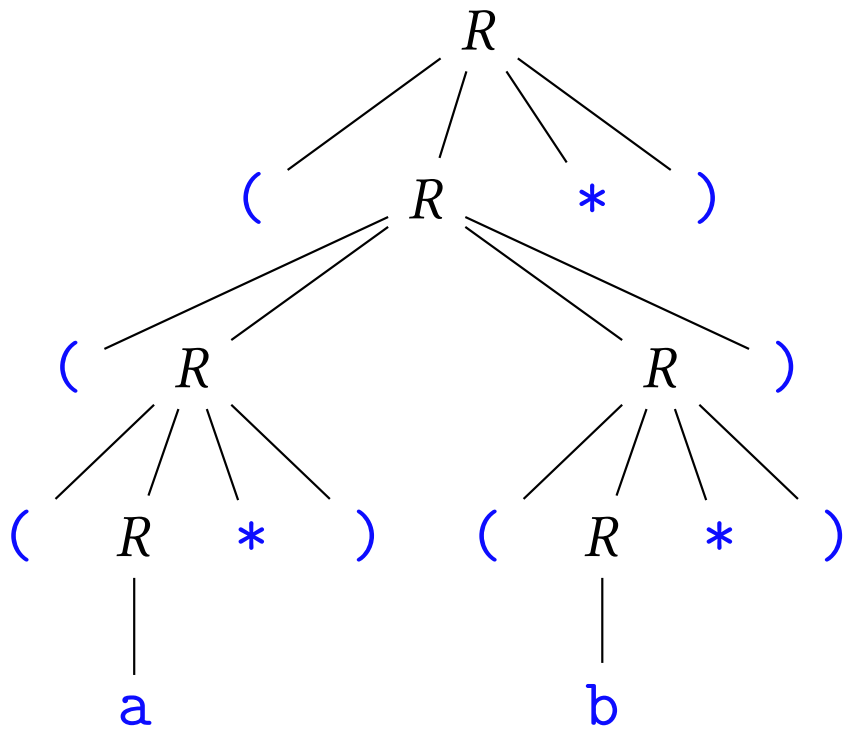
- $R = \texttt{a|b}$: Dann ist
  $\langle R \rangle = \langle \texttt{a|b} \rangle = \langle \texttt{a} \rangle \cup \langle \texttt{b} \rangle = \{\texttt{a}\} \cup \{\texttt{b}\} = \{\texttt{a}, \texttt{b}\}$

- $R = \texttt{(a|b)*}$: Dann ist
  $\langle R \rangle = \langle \texttt{(a|b)*} \rangle = \langle \texttt{a|b} \rangle^* = \{\texttt{a}, \texttt{b}\}^*$

- $R = \texttt{(a*b*)*}$: Dann ist
  $\langle R \rangle = \langle \texttt{(a*b*)*} \rangle = \langle \texttt{a*b*} \rangle^*$
  $= (\langle \texttt{a*} \rangle \langle \texttt{b*} \rangle)^* = (\langle \texttt{a} \rangle^* \langle \texttt{b} \rangle^*)^* = (\{\texttt{a}\}^* \{\texttt{b}\}^*)^*$

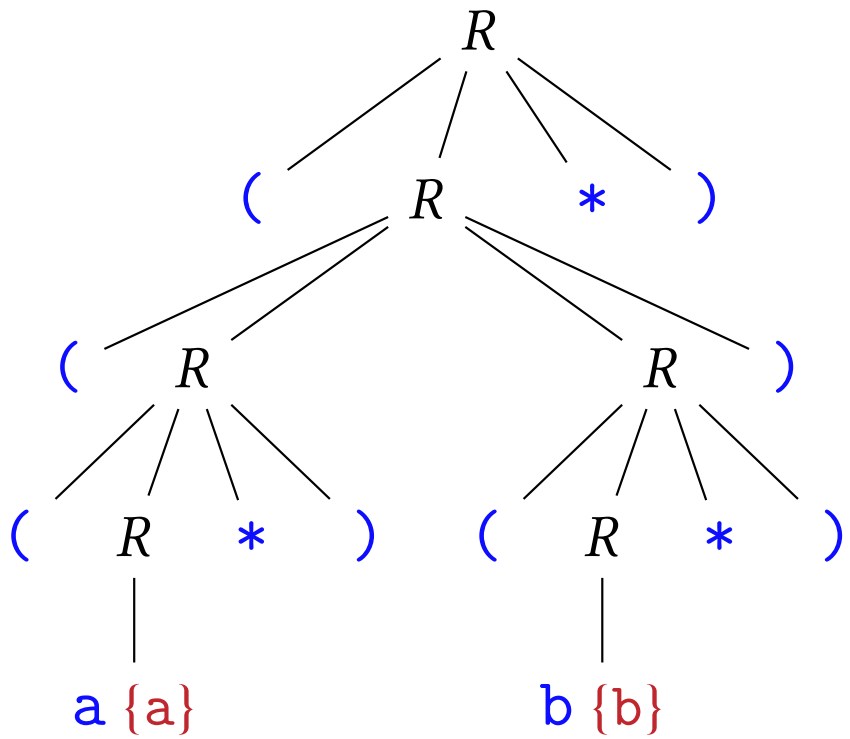
- Nachdenken: $(\{\texttt{a}\}^* \{\texttt{b}\}^*)^* = \{\texttt{a}, \texttt{b}\}^*$

Bestimmung von $\langle R \rangle$ entlang des Ableitungsbaums von R



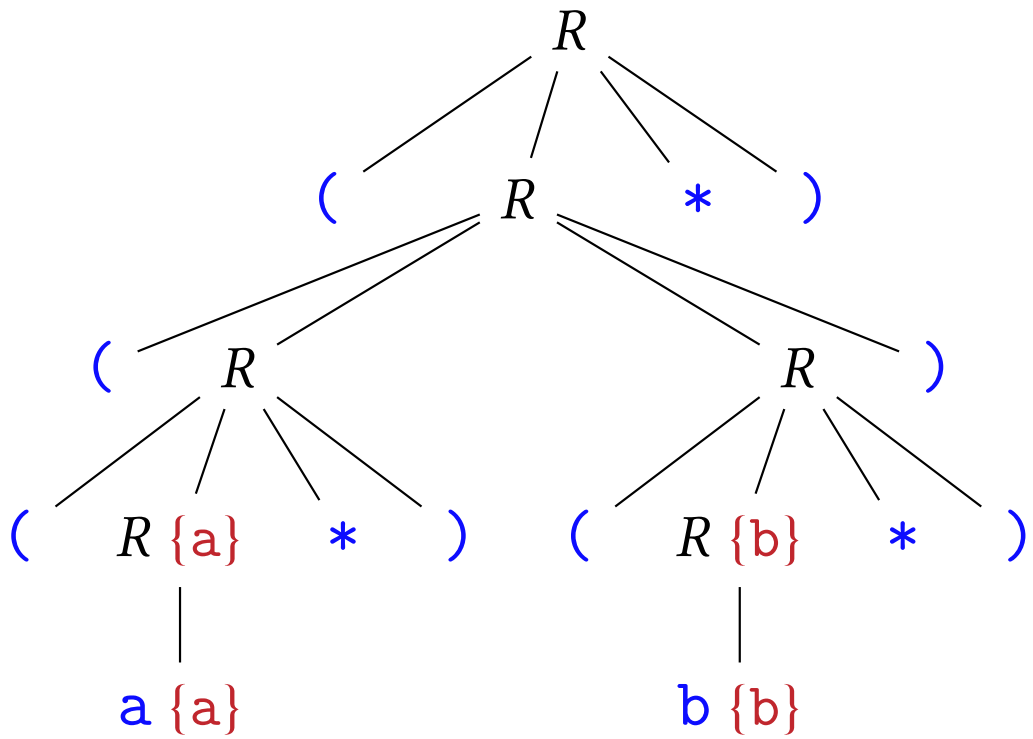
analog zur Auswertung
arithmetischer Ausdrücke

Bestimmung von $\langle R \rangle$ entlang des Ableitungsbaums von R



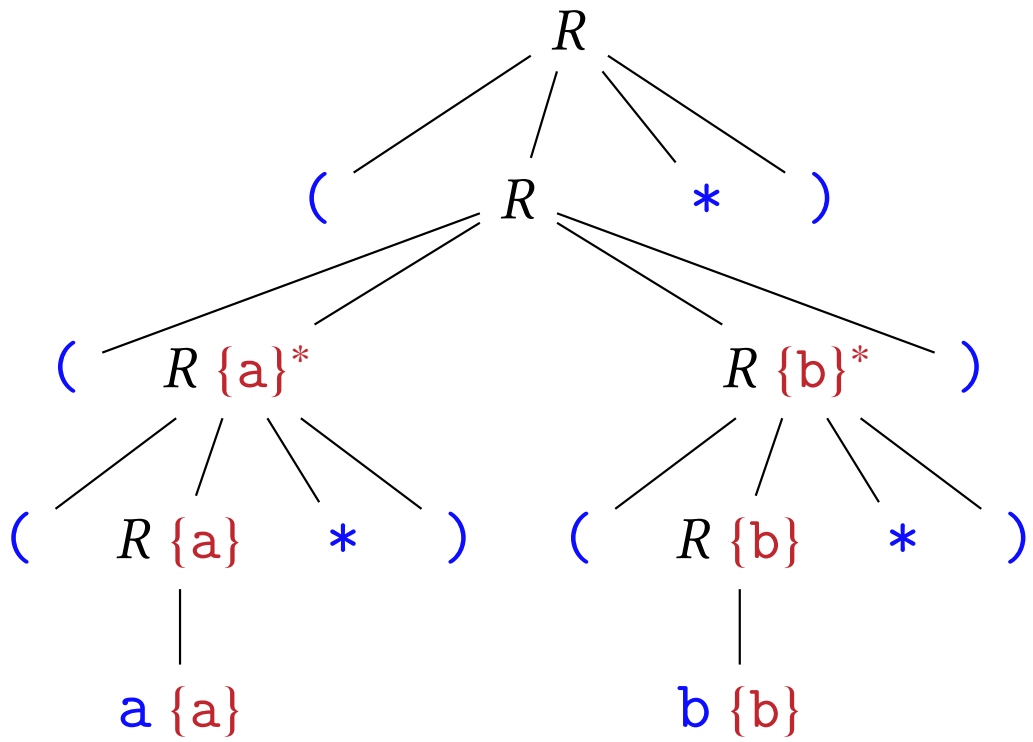
analog zur Auswertung
arithmetischer Ausdrücke

Bestimmung von $\langle R \rangle$ entlang des Ableitungsbaums von R



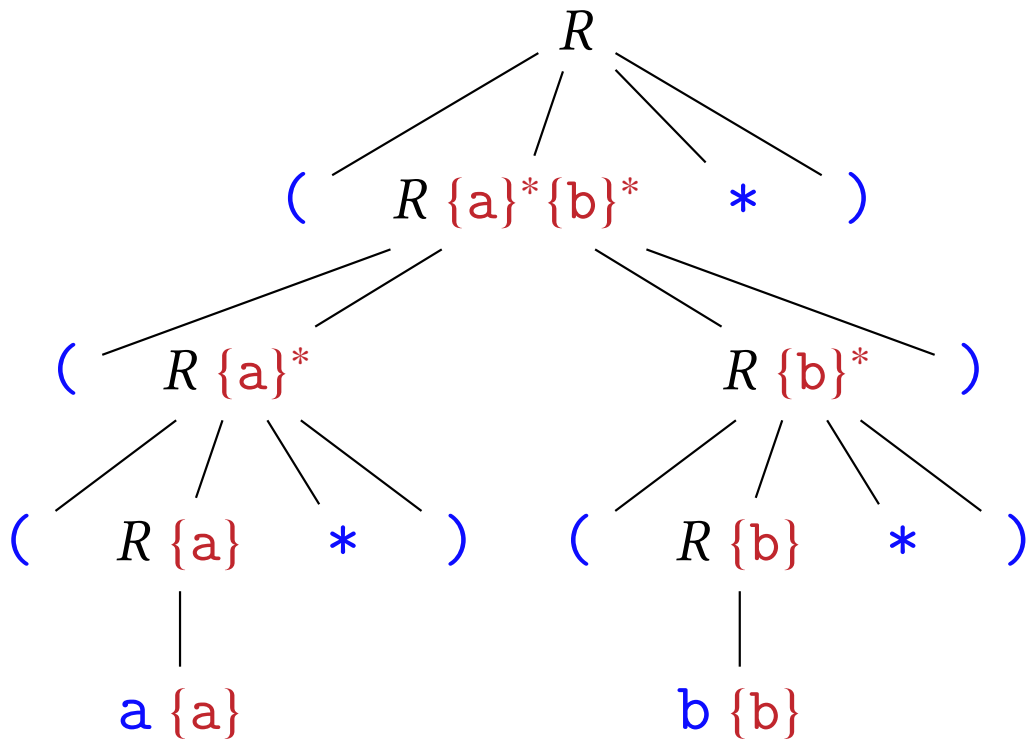
analog zur Auswertung
arithmetischer Ausdrücke

Bestimmung von $\langle R \rangle$ entlang des Ableitungsbaums von R



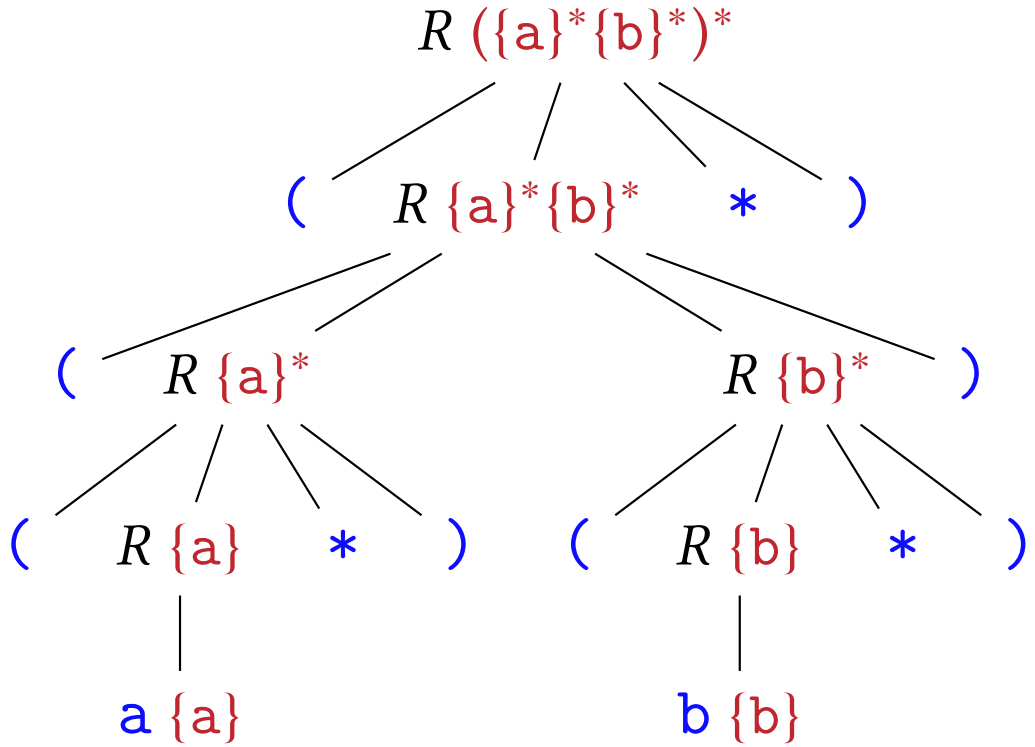
analog zur Auswertung
arithmetischer Ausdrücke

Bestimmung von $\langle R \rangle$ entlang des Ableitungsbaums von R



analog zur Auswertung arithmetischer Ausdrücke

Bestimmung von $\langle R \rangle$ entlang des Ableitungsbaums von R



analog zur Auswertung
arithmetischer Ausdrücke

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?

Äquivalenz regulärer Ausdrücke

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.

Weitere Beispiele für $\langle R \rangle$

`ab(ab)*`

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

Weitere Beispiele für $\langle R \rangle$

`ab(ab)*`

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

allgemein: $R\texttt{(}R\texttt{)*}$

- $\langle R \rangle \langle \texttt{(}R\texttt{)*} \rangle = \langle R \rangle \langle R \rangle^* = \langle R \rangle^+$
- gelegentlich Abkürzung $\texttt{(}R\texttt{)+}$ bzw. $R\texttt{+}$

Weitere Beispiele für $\langle R \rangle$

`ab(ab)*`

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

allgemein: $R\texttt{(}R\texttt{)*}$

- $\langle R \rangle \langle \texttt{(}R\texttt{)*} \rangle = \langle R \rangle \langle R \rangle^* = \langle R \rangle^+$
- gelegentlich Abkürzung $\texttt{(}R\texttt{)+}$ bzw. $R\texttt{+}$

`abc|∅*`

- $\langle \texttt{abc|∅*} \rangle = \cdots = \langle \texttt{abc} \rangle \cup \langle \texttt{∅*} \rangle = \langle \texttt{abc} \rangle \cup \langle \texttt{∅} \rangle^*$
  $= \{\texttt{abc}\} \cup \{\}^* = \{\texttt{abc}, \varepsilon\}$

Weitere Beispiele für $\langle R \rangle$

`ab(ab)*`

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

allgemein: $R\texttt{(}R\texttt{)*}$

- $\langle R \rangle \langle \texttt{(}R\texttt{)*} \rangle = \langle R \rangle \langle R \rangle^* = \langle R \rangle^+$
- gelegentlich Abkürzung $\texttt{(}R\texttt{)+}$ bzw. $R\texttt{+}$

`abc|∅*`

- $\langle \texttt{abc|∅*} \rangle = \cdots = \langle \texttt{abc} \rangle \cup \langle \texttt{∅*} \rangle = \langle \texttt{abc} \rangle \cup \langle \texttt{∅} \rangle^*$
  $= \{\texttt{abc}\} \cup \{\}^* = \{\texttt{abc}, \varepsilon\}$

allgemein: $R\texttt{|∅*}$:

- $\langle R\texttt{|∅*} \rangle = \langle R \rangle \cup \langle \texttt{∅*} \rangle = \langle R \rangle \cup \{\varepsilon\}$
- m. a. W.: das Vorkommen eines Wortes aus $\langle R \rangle$ ist „optional“
- verschiedentlich Abkürzungen wie z. B. $R\texttt{?}$ oder …

Wo sind wir?

RFC 5322: Internet Message Format

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, 21 Jan 2015 12:08:47 +0100`

RFC 5322, Abschnitt 3.3: Date and Time Specification, fast wörtlich:

```
date-time   = [ day-of-week "," ] date time [CFWS]

day-of-week = ([FWS] day-name)
day-name    = "Mon" / "Tue" / "Wed" / "Thu" /  ...

date        = day month year
day         = ([FWS] 1*2DIGIT FWS)
month       = "Jan" / "Feb" / "Mar" / "Apr" /  ...
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)
```

Datums- und Zeitangaben in Emails (2)

Beispiel

```
time-of-day = hour ":" minute [ ":" second ]
hour        = 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:
DIGIT = 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9, also
⟨DIGIT⟩ = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

2DIGIT ist Abkürzung für DIGIT DIGIT, also z. B.
⟨2DIGIT⟩ = {00, 01, 02, . . . , 99}

also z. B. ⟨hour⟩ = {00, 01, 02, . . . , 99}

Datums- und Zeitangaben in Emails (3)

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

Wo sind wir?

Charakterisierungen regulärer Sprachen

Satz

Für jede formale Sprache L sind 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.

Solche Sprachen heißen regulär.

Beweis in «Theoretische Grundlagen der Informatik»

rechtslineare Grammatiken kommen gleich

- sie sind kontextfrei,
  also ist jede reguläre Sprache eine kontextfreie Sprache
- *aber nicht umgekehrt*
  - Gegenbeispiel: $\{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\}$

Zum Beweis des Satzes

konstruktiv:

- von endlichem Akzeptor A zu regulärem Ausdruck R mit $L(A) = \langle R \rangle$
  - „mittel schwer“, z. B. inspiriert vom Algorithmus von Warshall

Zum Beweis des Satzes

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“

Zum Beweis des Satzes

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“

Zum Beweis des Satzes

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 Rückrichtungen keine Konstruktionen nötig

Was ist wichtig

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

Wo sind wir?

Reguläre Ausdrücke

Rechtslineare Grammatiken (Typ 3)

Kantorowitsch-Bäume und strukturelle Induktion

Motivation

kontextfreie Grammatiken erzeugen jedenfalls zum Teil andere formale Sprachen, als man mit endlichen Akzeptoren erkennen kann.

Beispiel:

- $G = (\{X\}, \{\mathtt{a}, \mathtt{b}\}, X, \{X \to \mathtt{a}X\mathtt{b} \mid \varepsilon\})$ erzeugt $\{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\}$
- und diese Sprache ist nicht regulär.

Kann man kontextfreie Grammatiken so einschränken, dass sie zu endlichen Akzeptoren passen?

Rechtslineare Grammatiken: Definition

Eine *rechtslineare Grammatik* ist eine kontextfreie Grammatik $G = (N, T, S, P)$, die der folgenden Einschränkung genügt: Jede Produktion ist

- entweder von der Form $X \to w$ mit $w \in T^*$
- oder von der Form $X \to wY$ mit $w \in T^*$ und $X, Y \in N$.

also auf jeder rechten Seite

- höchstens ein Nichterminalsymbol
- und wenn dann nur als letztes Symbol

Rechtslineare Grammatiken: Beispiele

$$G = (\{X\}, \{\mathtt{a}, \mathtt{b}\}, X, \{X \to \mathtt{ab}X \mid \mathtt{bba}X \mid \varepsilon\}$$

Rechtslineare Grammatiken: Beispiele

$$G = (\{X\}, \{\texttt{a}, \texttt{b}\}, X, \{X \to \texttt{ab}X \mid \texttt{bba}X \mid \varepsilon\}$$

$$L(G) = \langle \texttt{(ab|bba)*} \rangle$$

Rechtslineare Grammatiken: Beispiele

$$G = (\{X\}, \{\mathtt{a}, \mathtt{b}\}, X, \{X \to \mathtt{ab}X \mid \mathtt{bba}X \mid \varepsilon\}$$

$$L(G) = \langle \mathtt{(ab|bba)*} \rangle$$

$$G = (\{X, Y\}, \{\mathtt{a}, \mathtt{b}\}, X,$$
$$\{X \to \mathtt{a}X \mid \mathtt{b}X \mid \mathtt{ababb}Y, Y \to \mathtt{a}Y \mid \mathtt{b}Y \mid \varepsilon\}$$

Rechtslineare Grammatiken: Beispiele

$$G = (\{X\}, \{\texttt{a}, \texttt{b}\}, X, \{X \to \texttt{ab}X \mid \texttt{bba}X \mid \varepsilon\}$$

$$L(G) = \langle \texttt{(ab|bba)*} \rangle$$

$$G = (\{X, Y\}, \{\texttt{a}, \texttt{b}\}, X,$$
$$\{X \to \texttt{a}X \mid \texttt{b}X \mid \texttt{ababb}Y, Y \to \texttt{a}Y \mid \texttt{b}Y \mid \varepsilon\}$$

$$L(G) = \langle \texttt{(a|b)*ababb(a|b)*} \rangle$$

Rechtslineare Grammatiken: Nichtbeispiel

$G = (\{X\}, \{\mathtt{a}, \mathtt{b}\}, X, \{X \to \mathtt{a}X\mathtt{b} \mid \varepsilon\})$ ist nicht rechtslinear,

- denn in $X \to \mathtt{a}X\mathtt{b}$ steht das Nichtterminalsymbol X nicht am rechten Ende.

Da die erzeugte formale Sprache $\{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\}$ von keinem endlichen Akzeptor erkannt wird,
kann es auch gar keine rechtslineare Grammatik geben.

Sprechweisen

Rechtslineare Grammatiken heißen auch
Typ-3-Grammatiken.

Sprechweisen

Rechtslineare Grammatiken heißen auch *Typ-3-Grammatiken*.

Kontextfreien Grammatiken heißen auch *Typ-2-Grammatiken*.

Sprechweisen

Rechtslineare Grammatiken heißen auch *Typ-3-Grammatiken*.

Kontextfreien Grammatiken heißen auch *Typ-2-Grammatiken*.

Es gibt auch noch

- Typ-1-Grammatiken und
- Typ-0-Grammatiken,

die wir hier nicht weiter betrachten werden.

Sprechweisen

Rechtslineare Grammatiken heißen auch *Typ-3-Grammatiken*.

Kontextfreien Grammatiken heißen auch *Typ-2-Grammatiken*.

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.

Vorteil rechtslinearer Grammatiken

Wozu rechtslineare Grammatiken?

Vorteil rechtslinearer Grammatiken

Wozu rechtslineare Grammatiken?

gegenüber deterministischen endlichen Akzeptoren: manchmal deutlich kürzer und übersichtlicher hinzuschreiben

genaueres Verständnis dafür: im 3. Semester bei nichtdeterministischen endliche Akzeptoren

Wo sind wir?

Reguläre Ausdrücke

Rechtslineare Grammatiken (Typ 3)

Kantorowitsch-Bäume und strukturelle Induktion

Ziel dieses Abschnittes

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?

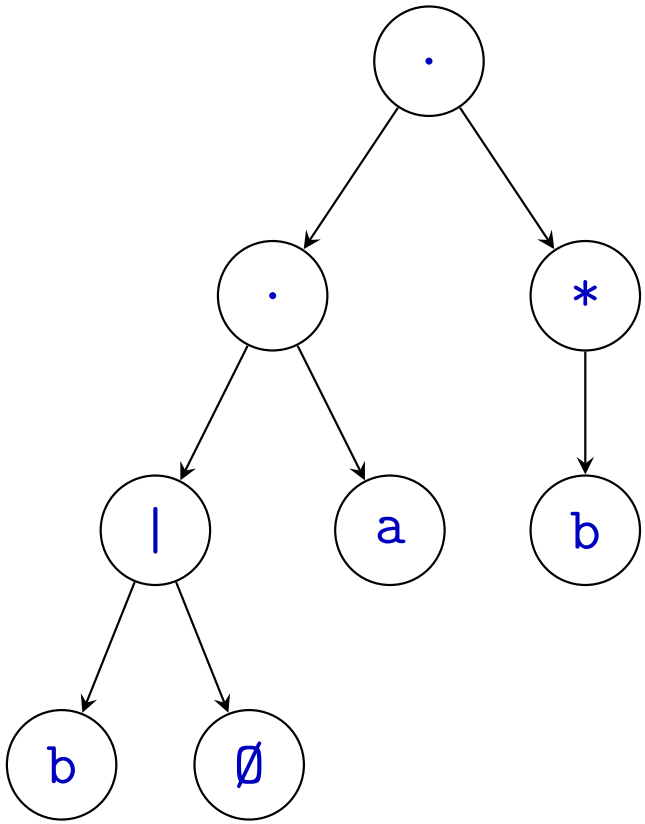
eine Möglichkeit: *strukturelle Induktion*

- 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

Mit Kantorowitsch-Bäumen kann man z. B. reguläre Ausdrücke repräsentieren



regulärer Ausdruck: `((b|∅)a)(b*)`

Darstellung als sogenannter Kantorowitsch-Baum

- innere Knoten: «Operatoren»
- Blätter: «Operanden»

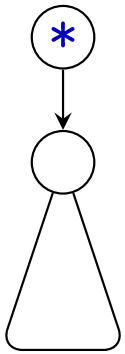
«Regex-Bäume»

Beachte:

- das ist *kein* Ableitungsbaum gemäß einer Grammatik
- aber «genauso gut» und kompakter

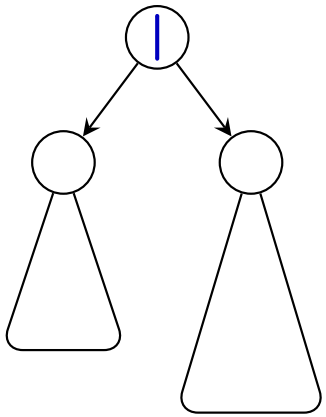
Regex-Bäume — etwas genauer

Es sei A Alphabet.

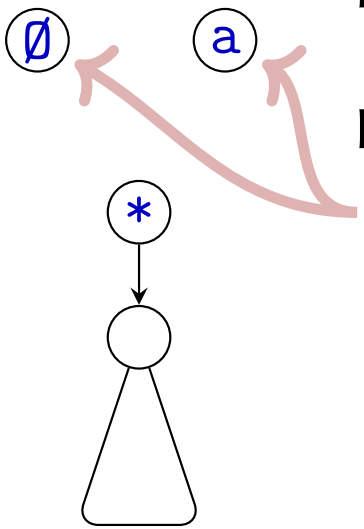


Ein Baum ist ein Regex-Baum, wenn gilt:

- entweder Wurzel ist Blatt, mit $x \in A$ oder ∅ beschriftet,
- oder Wurzel mit $*$ beschriftet und hat
  genau einen Nachfolger, der Wurzel eines Regex-Baumes ist
- oder Wurzel mit $\cdot$ oder $|$ beschriftet und hat
  genau zwei Nachfolger, die Wurzeln zweier Regex-Bäume sind.



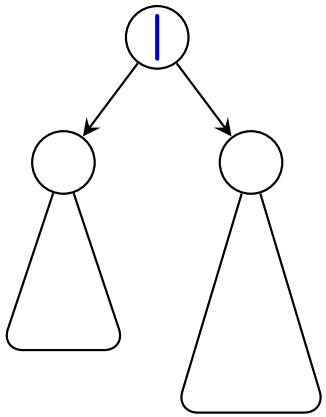
Regex-Bäume — etwas genauer



Es sei A Alphabet.

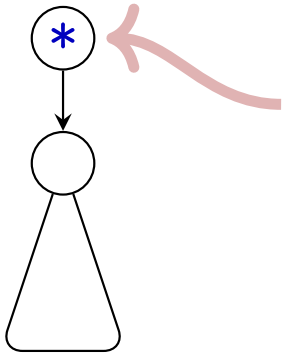
Ein Baum ist ein Regex-Baum, wenn gilt:

- entweder Wurzel ist Blatt, mit $x \in A$ oder $\emptyset$ beschriftet,
- oder Wurzel mit $*$ beschriftet und hat genau einen Nachfolger, der Wurzel eines Regex-Baumes ist
- oder Wurzel mit $\cdot$ oder $|$ beschriftet und hat genau zwei Nachfolger, die Wurzeln zweier Regex-Bäume sind.



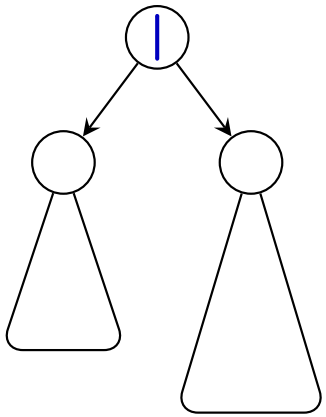
Regex-Bäume — etwas genauer

Es sei A Alphabet.

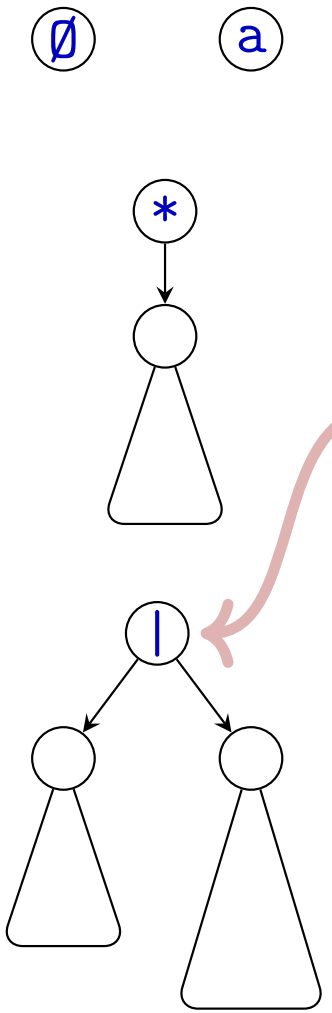


Ein Baum ist ein Regex-Baum, wenn gilt:

- entweder Wurzel ist Blatt, mit $x \in A$ oder $\emptyset$ beschriftet,
- oder Wurzel mit $*$ beschriftet und hat
  genau einen Nachfolger, der Wurzel eines Regex-Baumes ist
- oder Wurzel mit $\cdot$ oder $|$ beschriftet und hat
  genau zwei Nachfolger, die Wurzeln zweier Regex-Bäume sind.



Regex-Bäume — etwas genauer



Es sei A Alphabet.

Ein Baum ist ein Regex-Baum, wenn gilt:

- entweder Wurzel ist Blatt, mit $x \in A$ oder $\emptyset$ beschriftet,
- oder Wurzel mit $*$ beschriftet und hat
  genau einen Nachfolger, der Wurzel eines Regex-Baumes ist
- oder Wurzel mit $\cdot$ oder $|$ beschriftet und hat
  genau zwei Nachfolger, die Wurzeln zweier Regex-Bäume sind.

Vollständige Induktion über die Baumhöhe

Größere Bäume sind „aus kleineren zusammengesetzt“, und zwar auf eindeutige Weise.

Bijektion Regex-Bäume $\leftrightarrow$ reguläre Ausdrücke

Vollständige Induktion über die Baumhöhe

Größere Bäume sind „aus kleineren zusammengesetzt“, und zwar auf eindeutige Weise.

Bijektion Regex-Bäume $\leftrightarrow$ reguläre Ausdrücke

Höhe $h(T)$ eines Baumes

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

Beweis einer Aussage für alle regulären Ausdrücke:
durch Beweis der Aussage für alle Regex-Bäume

Beweis einer Aussage für alle Regex-Bäume:
vollständige Induktion über Höhe der Regex-Bäume

Vollständige Induktion über die Baumhöhe — ein Problem

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: siehe Kapitel 6

Erinnerung: Verallgemeinerung vollständiger Induktion

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 durch vollständige Induktion: $\forall n \in \mathbb{N}_0 : \mathcal{A}(n)$

- das reicht, denn aus $\mathcal{A}(n)$ folgt $\mathcal{B}(n)$

Induktion über Höhe der Regex-Bäume

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

Für jeden Regex-Baum R der Höhe n gibt es eine rechtslineare Grammatik G gibt 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.

Induktion über Höhe der Regex-Bäume

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

Für jeden Regex-Baum R der Höhe n gibt es eine rechtslineare Grammatik G gibt 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.
- Induktionsschritt: $n \rightsquigarrow n+1$: es sei $n \in \mathbb{N}_0$ so, dass die Induktionsvoraussetzung gilt: $\forall i \leq n : \mathcal{B}(i)$,

Induktion über Höhe der Regex-Bäume

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

Für jeden Regex-Baum R der Höhe n gibt es eine rechtslineare Grammatik G gibt 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.
- Induktionsschritt: $n \rightsquigarrow n+1$: es sei $n \in \mathbb{N}_0$ so, dass die Induktionsvoraussetzung gilt: $\forall i \leq n : \mathcal{B}(i)$,
  d. h. für jeden Regex-Baum R' der Höhe $i \leq n$
  gibt es eine T3G G mit $\langle R' \rangle = L(G)$.

Induktion über Höhe der Regex-Bäume

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

Für jeden Regex-Baum R der Höhe n gibt es eine rechtslineare Grammatik G gibt 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.
- Induktionsschritt: $n \rightsquigarrow n+1$: es sei $n \in \mathbb{N}_0$ so, dass die
  Induktionsvoraussetzung gilt: $\forall i \leq n : \mathcal{B}(i)$,
  d. h. für jeden Regex-Baum R' der Höhe $i \leq n$
  gibt es eine T3G G mit $\langle R' \rangle = L(G)$.
  zeige: $\mathcal{B}(n+1)$ gilt

Induktion über Höhe der Regex-Bäume

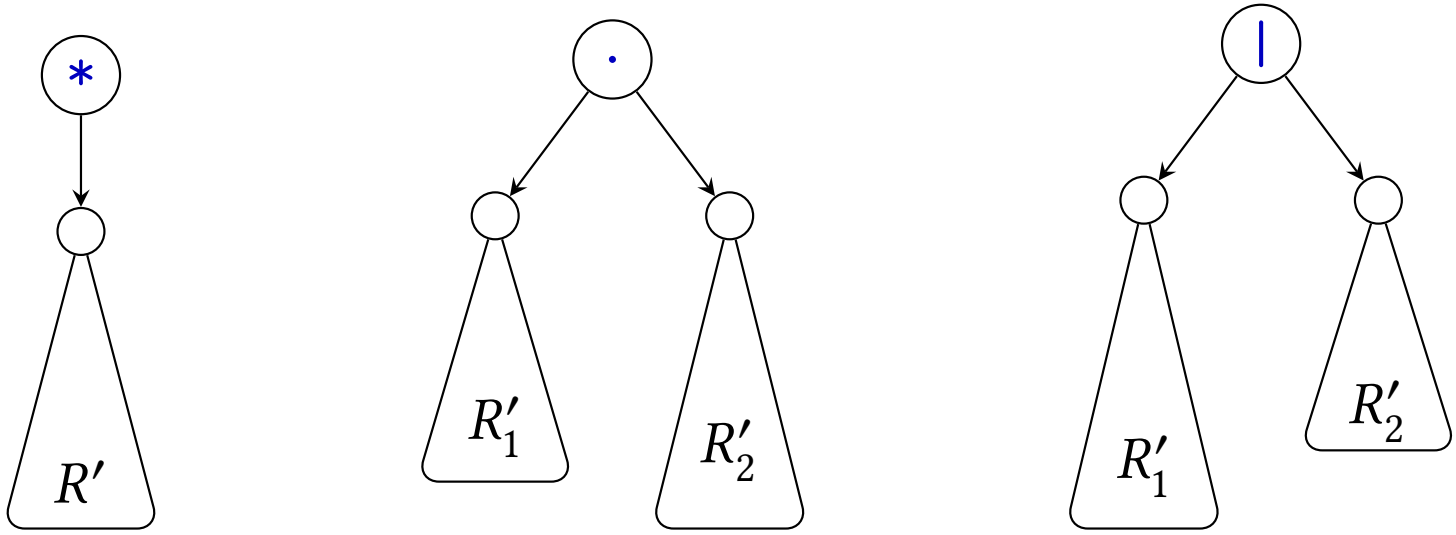
Aussage $\mathcal{B}(n)$:

Für jeden Regex-Baum R der Höhe n gibt es eine rechtslineare Grammatik G gibt 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.
- Induktionsschritt: $n \rightsquigarrow n+1$: es sei $n \in \mathbb{N}_0$ so, dass die
  Induktionsvoraussetzung gilt: $\forall i \leq n : \mathcal{B}(i)$,
  d. h. für jeden Regex-Baum R' der Höhe $i \leq n$
  gibt es eine T3G G mit $\langle R' \rangle = L(G)$.
  zeige: $\mathcal{B}(n+1)$ gilt
  d. h. für jeden Regex-Baum R der Höhe $n+1$
  existiert T3G G mit $\langle R \rangle = L(G)$.

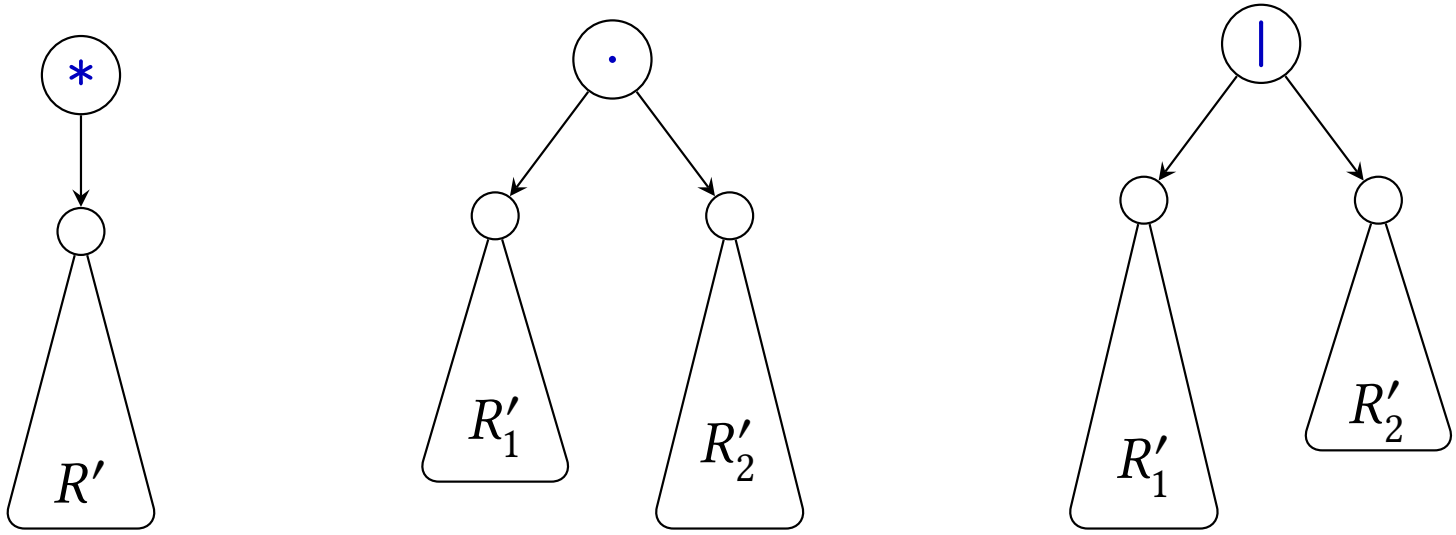
Skizze des Induktionsschritts (1)

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



Skizze des Induktionsschritts (1)

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



Induktionsvoraussetzung: für alle Unterbäume gibt es T3G

Aus T3G für Unterbäume kann man T3G für ganzen Regex-Baum konstruieren.

Skizze des Induktionsschritts (2)

Beispiel: R ist $R_1 \mid R_2$

Skizze des Induktionsschritts (2)

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. $L(G_2) = \langle R_2 \rangle$

es sei $N_1 \cap N_2 = \emptyset$

- keine Beschränkung der Allgemeinheit
- wichtig für die Konstruktion

Skizze des Induktionsschritts (2)

Beispiel: R ist $R_1 \,|\, R_2$

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. $L(G_2) = \langle R_2 \rangle$

es sei $N_1 \cap N_2 = \emptyset$

- keine Beschränkung der Allgemeinheit
- wichtig für die Konstruktion

wähle „neues“ Nichtterminalsymbol $S \notin N_1 \cup N_2$

Behauptung:

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

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

Skizze des Induktionsschritts (2)

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. $L(G_2) = \langle R_2 \rangle$

es sei $N_1 \cap N_2 = \emptyset$

- keine Beschränkung der Allgemeinheit
- wichtig für die Konstruktion

wähle „neues“ Nichtterminalsymbol $S \notin N_1 \cup N_2$

Behauptung:

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

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

- Rechtslinearität: klar
- $L(G) = L(G_1) \cup L(G_2)$: macht Arbeit, aber nicht hier

Strukturelle Induktion

gegeben «Gebilde» (eben reguläre Ausdrücke), und zwar

- kleinste «atomare»/«elementare» Gebilde
  - bei RA: $\emptyset$ und x für $x \in A$
- Konstruktionsvorschrift(en), nach denen man aus kleineren Gebilden größere bauen kann
  - bei RA: $R*$, $R_1 \mid R_2$, $R_1 \cdot R_2$

Strukturelle Induktion

gegeben «Gebilde» (eben reguläre Ausdrücke), und zwar

- kleinste «atomare»/«elementare» Gebilde
  - bei RA: $\emptyset$ und x für $x \in A$
- Konstruktionsvorschrift(en), nach denen man aus kleineren Gebilden größere bauen kann
  - bei RA: $R*$, $R_1 \mid R_2$, $R_1 \cdot R_2$

Aufgabe:

Zeige, dass alle Gebilde eine gewisse Eigenschaft haben.

Strukturelle Induktion:

- Induktionsanfang: zeige:
  alle «atomaren» Gebilde haben die Eigenschaft

Strukturelle Induktion

gegeben «Gebilde» (eben reguläre Ausdrücke), und zwar

- kleinste «atomare»/«elementare» Gebilde
  - bei RA: $\emptyset$ und x für $x \in A$
- Konstruktionsvorschrift(en), nach denen man aus kleineren Gebilden größere bauen kann
  - bei RA: $R*$, $R_1 \mid R_2$, $R_1 \cdot R_2$

Aufgabe:

Zeige, dass alle Gebilde eine gewisse Eigenschaft haben.

Strukturelle Induktion:

- Induktionsanfang: zeige:
  alle «atomaren» Gebilde haben die Eigenschaft
- Induktionsschritt: zeige:
  «zusammengesetzten» Gebilde haben die Eigenschaft, weil alle Untergebilde die Eigenschaft haben
  - gleich welche Konstruktionsvorschrift benutzt wurde

Was ist wichtig

Das sollten Sie mitnehmen:

- Definition rechtslineare Grammatiken

Das sollten Sie üben:

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

Zusammenfassung

reguläre Ausdrücke

- werden von diversen «Werkzeugen» unterstützt
- in manchen Programmiersprachen zur Textverarbeitung praktisch

rechtslineare Grammatiken

strukturelle Induktion