Grundbegriffe der Informatik

Einheit 15: Reguläre Ausdrücke und rechtslineare Grammatiken

Thomas Worsch

Karlsruher Institut für Technologie, Fakultät für Informatik

Wintersemester 2009/2010

Was kann man mit endlichen Akzeptoren?

- manche Sprachen kann man mit endlichen Akzeptoren erkennen
  - z. B. $\{\mathtt{a}\}^+\{\mathtt{b}\} \cup \{\mathtt{b}\}^+\{\mathtt{a}\}$
- manche Sprachen *nicht*
  - 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

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

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

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

Zum Begriff

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
  - verschiedene Varianten (bei Syntax, bei Semantik)
  - emacs, grep, sed, ...
  - Java (java.util.regex), Python, Perl, ...

Definition regulärer Ausdrücke (1)

- sei A ein Alphabet, das kein Zeichen aus Z enthält
- sei Z das Alphabet $Z = \{\mathtt{|}, \mathtt{(}, \mathtt{)}, \mathtt{*}, \mathtt{\emptyset}\}$ („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:
  - $\mathtt{\emptyset}$ 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 $\mathtt{(}R_1 \mathtt{|} R_2\mathtt{)}$ und $\mathtt{(}R_1 R_2\mathtt{)}$ reguläre Ausdrücke.
  - Wenn R ein regulärer Ausdruck ist, dann auch $\mathtt{(}R\mathtt{*)}$.
  - Nichts anderes sind reguläre Ausdrücke.

Klammereinsparungsregeln

- „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
  - $((R_1 \,|\, R_2) \,|\, R_3)$

Beispiele

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

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

Mit Klammereinsparungsregeln:

- `ab`
- `aba`
- `abaa`
- `ab(aa)`
- `∅|b`
- `a|b`
- `a(a|b)|b`
- `(a|(b|(a|a)))`
- `∅*`
- `a*`
- `bab*`
- `(bab)*`
- `a**`
- `(abb)**|∅*`

Nichtbeispiele

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

`(|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 regulärer Ausdrücke (2)

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

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

Überblick

Die *von einem regulären Ausdruck R beschriebene formale Sprache* $\langle R\rangle$ ist wie folgt definiert:

- $\langle\emptyset\rangle = \{\}$ (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 $\langle R*\rangle = \langle R\rangle^*$.

- die Definition folgt der für reguläre Ausdrücke

Die *von einem regulären Ausdruck R beschriebene formale Sprache* $\langle R\rangle$ ist wie folgt definiert:

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

- die Definition folgt der für reguläre Ausdrücke

Beispiele für $\langle R \rangle$

Betrachten wir drei einfache Beispiele:

- $R = \texttt{a|b}$: 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
  $$\begin{aligned}\langle R \rangle &= \langle \texttt{(a*b*)*} \rangle = \langle \texttt{a*b*} \rangle^* \\ &= (\langle \texttt{a*} \rangle \langle \texttt{b*} \rangle)^* = (\langle \texttt{a} \rangle^* \langle \texttt{b} \rangle^*)^* = (\{\texttt{a}\}^* \{\texttt{b}\}^*)^* .\end{aligned}$$
- Nachdenken: $(\{\texttt{a}\}^* \{\texttt{b}\}^*)^* = \{\texttt{a}, \texttt{b}\}^*$

Beispiele für $\langle R \rangle$

Betrachten wir drei einfache Beispiele:

- $R = \texttt{a|b}$: 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
  $$\begin{aligned}\langle R \rangle &= \langle \texttt{(a*b*)*} \rangle = \langle \texttt{a*b*} \rangle^* \\ &= (\langle \texttt{a*} \rangle \langle \texttt{b*} \rangle)^* = (\langle \texttt{a} \rangle^* \langle \texttt{b} \rangle^*)^* = (\{\texttt{a}\}^* \{\texttt{b}\}^*)^* .\end{aligned}$$
- Nachdenken: $(\{\texttt{a}\}^* \{\texttt{b}\}^*)^* = \{\texttt{a}, \texttt{b}\}^*$

Beispiele für $\langle R \rangle$

Betrachten wir drei einfache Beispiele:

- $R = \texttt{a|b}$: 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
  $$\begin{aligned} \langle R \rangle &= \langle \texttt{(a*b*)*} \rangle = \langle \texttt{a*b*} \rangle^* \\ &= (\langle \texttt{a*} \rangle \langle \texttt{b*} \rangle)^* = (\langle \texttt{a} \rangle^* \langle \texttt{b} \rangle^*)^* = (\{\texttt{a}\}^* \{\texttt{b}\}^*)^* \ . \end{aligned}$$
- Nachdenken: $(\{\texttt{a}\}^* \{\texttt{b}\}^*)^* = \{\texttt{a}, \texttt{b}\}^*$

Beispiele für $\langle R \rangle$

Betrachten wir drei einfache Beispiele:

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

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 einem der nächsten Kapitel
  - d. h. jedenfalls: alle bisher bekannten (!) Algorithmen sind im allgemeinen sehr sehr sehr langsam, Laufzeit z. B. 2^{n^2} …
- Man weiß nicht, ob es vielleicht doch Algorithmen mit polynomieller Laufzeit für das Problem gibt, aber man sie „nur noch nicht gefunden“ hat.

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|Ø*`
  - $\langle \texttt{abc|Ø*} \rangle = \cdots = \langle \texttt{abc} \rangle \cup \langle \texttt{Ø*} \rangle = \langle \texttt{abc} \rangle \cup \langle \texttt{Ø} \rangle^*$
    $= \{\texttt{abc}\} \cup \{\}^* = \{\texttt{abc}, \varepsilon\}$
- allgemein: $R\texttt{|Ø*}$:
  - $\langle R\texttt{|Ø*} \rangle = \langle R \rangle \cup \langle \texttt{Ø*} \rangle = \langle R \rangle \cup \{\varepsilon\}$
  - m. a. W.: das Vorkommen eines Wortes aus $\langle R \rangle$ ist „optional“
  - auch dafür verschiedene Abkürzungen (je nach Anwendung) z. B. $\texttt{(}R\texttt{)?}$ bzw. $R\texttt{?}$ oder $\texttt{[}R\texttt{]}$ oder …

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

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

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

Überblick

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, 13 Jan 2010 09:50:17 +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" /
              "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)
```

Datums- und Zeitangaben in Emails (2)

- 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 = wo was wie ein regulärer Ausdruck, der statt Teilausdrücken deren Namen benutzen kann
- an anderer Stelle definiert:
  $\texttt{DIGIT} = 0|1|2|3|4|5|6|7|8|9$, also
  $\langle\texttt{DIGIT}\rangle = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$
- 2DIGIT ist Abkürzung für $\texttt{DIGIT}\,\texttt{DIGIT}$, also z. B.
  $\langle\texttt{2DIGIT}\rangle = \{00, 01, 02, \ldots, 99\}$
- also auch
  $\langle\texttt{hour}\rangle = \langle\texttt{minute}\rangle = \langle\texttt{second}\rangle = \{00, 01, 02, \ldots, 99\}$

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 erlaubt
  - aber `39:71` wäre syntaktisch korrekt

Überblick

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 $\{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\}$ 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 $\{\mathtt{a}^k\mathtt{b}^k \mid k \in \mathbb{N}_0\}$ sieht.

Zum Beweis des Satzes

konstruktiv:

- zu gegebenem endlichen Akzeptor A ein regulärer Ausdruck R mit $\langle R \rangle = L(A)$
  - „mittel schwer“, z. B. inspiriert vom Algorithmus von Warshall
- zu gegebenem regulären Ausdruck R eine rechtslineare Grammatik G mit $L(G) = \langle R \rangle$:
  - „relativ leicht“
  - gleich noch ein bisschen genauer,
    wenn rechtslineare Grammatiken eingeführt sind
- zu gegebener rechtslinearer Grammatik G ein endlicher Akzeptor A mit $L(A) = L(G)$:
  - „am schwierigsten“
- beachte: für die umgekehrten Richtungen braucht man keine zusätzlichen expliziten Konstruktionen

konstruktiv:

- zu gegebenem endlichen Akzeptor A ein regulärer Ausdruck R mit $\langle R \rangle = L(A)$
  - „mittel schwer", z. B. inspiriert vom Algorithmus von Warshall
- zu gegebenem regulären Ausdruck R eine rechtslineare Grammatik G mit $L(G) = \langle R \rangle$:
  - „relativ leicht"
  - gleich noch ein bisschen genauer,
    wenn rechtslineare Grammatiken eingeführt sind
- zu gegebener rechtslinearer Grammatik G ein endlicher Akzeptor A mit $L(A) = L(G)$:
  - „am schwierigsten"
- beachte: für die umgekehrten Richtungen braucht man keine zusätzlichen expliziten Konstruktionen

konstruktiv:

- zu gegebenem endlichen Akzeptor A ein regulärer Ausdruck R mit $\langle R \rangle = L(A)$
  - „mittel schwer", z. B. inspiriert vom Algorithmus von Warshall
- zu gegebenem regulären Ausdruck R eine rechtslineare Grammatik G mit $L(G) = \langle R \rangle$:
  - „relativ leicht"
  - gleich noch ein bisschen genauer, wenn rechtslineare Grammatiken eingeführt sind
- zu gegebener rechtslinearer Grammatik G ein endlicher Akzeptor A mit $L(A) = L(G)$:
  - „am schwierigsten"
- beachte: für die umgekehrten Richtungen braucht man keine zusätzlichen expliziten Konstruktionen

Zum Beweis des Satzes

konstruktiv:

- zu gegebenem endlichen Akzeptor A ein regulärer Ausdruck R mit $\langle R \rangle = L(A)$
  - „mittel schwer“, z. B. inspiriert vom Algorithmus von Warshall
- zu gegebenem regulären Ausdruck R eine rechtslineare Grammatik G mit $L(G) = \langle R \rangle$:
  - „relativ leicht“
  - gleich noch ein bisschen genauer,
    wenn rechtslineare Grammatiken eingeführt sind
- zu gegebener rechtslinearer Grammatik G ein endlicher Akzeptor A mit $L(A) = L(G)$:
  - „am schwierigsten“
- beachte: für die umgekehrten Richtungen braucht man keine zusätzlichen expliziten Konstruktionen

Das sollten Sie mitnehmen:

- Definition „klassischer“ regulärer Ausdrücke
  - atomare:
    - `Ø`
    - $a \in A$
  - zusammengesetzte:
    - $(R_1 \mid R_2)$
    - $(R_1 R_2)$
    - $(R)*$
- 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

Überblick

Motivation

- Mit kontextfreien Grammatiken kann man jedenfalls zum Teil andere formale Sprachen erzeugen, 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

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

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

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

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

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

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

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

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

Rechtslineare Grammatiken: Beispiele

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

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

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

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

Rechtslineare Grammatiken: Beispiele

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

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

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

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

- $G = (\{X\}, \{\mathtt{a}, \mathtt{b}\}, X, \{X \to \mathtt{a}X\mathtt{b} \mid \varepsilon\})$ 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.

- 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.

Sprechweisen

- 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.

Sprechweisen

- 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.

Sprechweisen

- 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.

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

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

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

Beweisskizze für das folgende

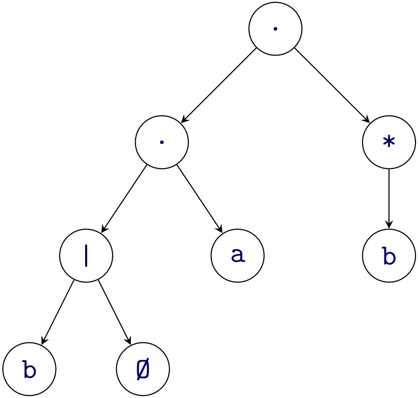
Lemma.

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

Reguläre Ausdrücke als Kantorowitsch-Bäume

- regulärer Ausdruck: `((b|∅)a)(b*)`
- Darstellung als sogenannter Kantorowitsch-Baum



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

„Regex-Bäume“

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

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

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

das Problem bei vollständiger Induktion über die Baumhöhe

- 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: $\Big(\forall i \leq n : \mathcal{B}(i)\Big) \wedge \mathcal{B}(n+1)$
    - $\forall i \leq n : \mathcal{B}(i)$: gilt nach Induktionsvoraussetzung
    - $\mathcal{B}(n+1)$: hier muss man was tun,
      aber man kann $\forall i \leq n : \mathcal{B}(i)$ *benutzen*

Einfache 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 $\forall n \in \mathbb{N}_0 : \mathcal{A}(n)$:
  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: $\Big(\forall i \leq n : \mathcal{B}(i)\Big) \wedge \mathcal{B}(n+1)$
    - $\forall i \leq n : \mathcal{B}(i)$: gilt nach Induktionsvoraussetzung
    - $\mathcal{B}(n+1)$: hier muss man was tun,
      aber man kann $\forall i \leq n : \mathcal{B}(i)$ benutzen

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 $\mathcal{B}(0)$, also finde rechtslineare Grammatiken, die die formalen Sprachen $\{x\} = \langle x \rangle$ für $x \in A$ und $\{\} = \langle \emptyset \rangle$ erzeugen.
  Das ist leicht.
- Induktionsvoraussetzung: für beliebiges aber festes $n \in \mathbb{N}_0$ gelte die Aussage $\forall i \leq n : \mathcal{B}(i)$, d. h. für jeden Regex-Baum R' mit einer Höhe $i \leq n$ gibt es eine rechtslineare Grammatik G gibt mit $\langle R' \rangle = L(G)$.
- Induktionsschluss: Es bleibt zu zeigen, dass auch $\mathcal{B}(n+1)$ gilt, dass also für jeden Regex-Baum R der Höhe $n+1$ eine rechtslineare Grammatik G mit $\langle R \rangle = L(G)$ existiert.

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 $\mathcal{B}(0)$, also finde rechtslineare Grammatiken, die die formalen Sprachen $\{x\} = \langle x \rangle$ für $x \in A$ und $\{\} = \langle \emptyset \rangle$ erzeugen. Das ist leicht.
- Induktionsvoraussetzung: für beliebiges aber festes $n \in \mathbb{N}_0$ gelte die Aussage $\forall i \leq n : \mathcal{B}(i)$, d. h. für jeden Regex-Baum R' mit einer Höhe $i \leq n$ gibt es eine rechtslineare Grammatik G gibt mit $\langle R' \rangle = L(G)$.
- Induktionsschluss: Es bleibt zu zeigen, dass auch $\mathcal{B}(n+1)$ gilt, dass also für jeden Regex-Baum R der Höhe $n+1$ eine rechtslineare Grammatik G mit $\langle R \rangle = L(G)$ existiert.

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 $\mathcal{B}(0)$, also
  finde rechtslineare Grammatiken, die die formalen Sprachen $\{x\} = \langle x \rangle$ für $x \in A$ und $\{\} = \langle \emptyset \rangle$ erzeugen.
  Das ist leicht.
- Induktionsvoraussetzung: für beliebiges aber festes $n \in \mathbb{N}_0$ gelte die Aussage $\forall i \leq n : \mathcal{B}(i)$, d. h. für jeden Regex-Baum R' mit einer Höhe $i \leq n$ gibt es eine rechtslineare Grammatik G gibt mit $\langle R' \rangle = L(G)$.
- Induktionsschluss: Es bleibt zu zeigen, dass auch $\mathcal{B}(n+1)$ gilt, dass also für jeden Regex-Baum R der Höhe $n+1$ eine rechtslineare Grammatik G mit $\langle R \rangle = L(G)$ existiert.

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 $\mathcal{B}(0)$, also
finde rechtslineare Grammatiken, die die formalen Sprachen $\{x\} = \langle x \rangle$ für $x \in A$ und $\{\} = \langle \emptyset \rangle$ erzeugen.
Das ist leicht.
- Induktionsvoraussetzung: für beliebiges aber festes $n \in \mathbb{N}_0$ gelte die Aussage $\forall i \leq n : \mathcal{B}(i)$, d. h. für jeden Regex-Baum R' mit einer Höhe $i \leq n$ gibt es eine rechtslineare Grammatik G gibt mit $\langle R' \rangle = L(G)$.
- Induktionsschluss: Es bleibt zu zeigen, dass auch $\mathcal{B}(n+1)$ gilt, dass also für jeden Regex-Baum R der Höhe $n+1$ eine rechtslineare Grammatik G mit $\langle R \rangle = L(G)$ existiert.

Skizze des Induktionsschritts (1)

- sei R beliebiger Regex-Baum der Höhe $n+1$
  mögliche Fälle:
  1. Die Wurzel von R ist ein $*$-Knoten und hat genau einen Unterbaum R' der Höhe n.
  2. Die Wurzel von R ist ein $|$-Knoten und hat genau zwei Unterbäume R_1 und R_2. Da R Höhe $n+1$ hat, hat einer der beiden Unterbäume Höhe n, der andere hat eine Höhe $i \leq n$.
  3. Die Wurzel von R ist ein „Konkatenations-Knoten“ und hat genau zwei Unterbäume R_1 und R_2. Da R Höhe $n+1$ hat, hat einer der beiden Unterbäume Höhe n, der andere hat eine Höhe $i \leq n$.
- In 2. und 3. existieren nach Induktionsvoraussetzung für *beide* Unterbäume rechtslineare Grammatiken der gewünschten Art.
- Aus Grammatiken für Unterbaum bzw. Unterbäume kann man Grammatik für ganzen Regex-Baum konstruieren.

Skizze des Induktionsschritts (1)

- sei R beliebiger Regex-Baum der Höhe $n+1$
  mögliche Fälle:
  1. Die Wurzel von R ist ein $*$-Knoten und hat genau einen Unterbaum R' der Höhe n.
  2. Die Wurzel von R ist ein $|$-Knoten und hat genau zwei Unterbäume R_1 und R_2. Da R Höhe $n+1$ hat, hat einer der beiden Unterbäume Höhe n, der andere hat eine Höhe $i \leq n$.
  3. Die Wurzel von R ist ein „Konkatenations-Knoten" und hat genau zwei Unterbäume R_1 und R_2. Da R Höhe $n+1$ hat, hat einer der beiden Unterbäume Höhe n, der andere hat eine Höhe $i \leq n$.
- In 2. und 3. existieren nach Induktionsvoraussetzung für *beide* Unterbäume rechtslineare Grammatiken der gewünschten Art.
- Aus Grammatiken für Unterbaum bzw. Unterbäume kann man Grammatik für ganzen Regex-Baum konstruieren.

Skizze des Induktionsschritts (1)

- sei R beliebiger Regex-Baum der Höhe $n+1$
  mögliche Fälle:
  1. Die Wurzel von R ist ein $*$-Knoten und hat genau einen Unterbaum R' der Höhe n.
  2. Die Wurzel von R ist ein $|$-Knoten und hat genau zwei Unterbäume R_1 und R_2. Da R Höhe $n+1$ hat, hat einer der beiden Unterbäume Höhe n, der andere hat eine Höhe $i \leq n$.
  3. Die Wurzel von R ist ein „Konkatenations-Knoten" und hat genau zwei Unterbäume R_1 und R_2. Da R Höhe $n+1$ hat, hat einer der beiden Unterbäume Höhe n, der andere hat eine Höhe $i \leq n$.
- In 2. und 3. existieren nach Induktionsvoraussetzung für *beide* Unterbäume rechtslineare Grammatiken der gewünschten Art.
- Aus Grammatiken für Unterbaum bzw. Unterbäume kann man Grammatik 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:

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

  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:

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

  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:

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

  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:

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

  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.

Strukturelle Induktion

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.

Strukturelle Induktion

1. generelle Situation:
   irgendwelche „Gebilde“ (eben reguläre Ausdrücke). Es gibt
   - kleinste „atomare“ oder „elementare“ Gebilde (eben reguläre Ausdrücke x für $x \in A$ und Ø) 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.

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 „Unix-Tools“ genutzt
  - in manchen Programmiersprachen zur Textverarbeitung praktisch
- rechtslineare Grammatiken
- strukturelle Induktion