Automaten, die auf Bäumen spazierengehen

Johannes Waldmann
Institut für Informatik
Universität Leipzig

Warum Baumsprachen

Daten mit hierarchischer Struktur

Programmiersprachen:
algebraische Datentypen (Records)

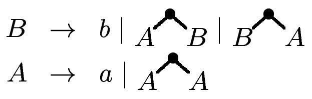
strukturierte Dokumente (XML):

```
<web-app>
    <servlet>
        <servlet-name>
            einsch
        </servlet-name>
        <servlet-class>
            Einsch
        </servlet-class>
    </servlet>
   <servlet-mapping>
        <servlet-name>
            einsch
        </servlet-name>
        <url-pattern>
            /
        </url-pattern>
    </servlet-mapping>
</web-app>
```

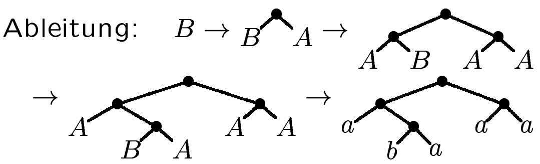

Reguläre Baum–Sprachen (REG)

Alphabet: $\{a, b, \wedge\}$

Grammatik:



Ableitung:



$t \in L(G) \iff \mathsf{yield}(t) \in a^*ba^*$

Blattsprache:

$\mathsf{yield}(x) = x, \quad \mathsf{yield}(l \wedge r) = \mathsf{yield}(l)\ \mathsf{yield}(r)$

Baum–Automaten

Grammatik $\approx$ Automat

- endlich
- nichtdeterministisch
- parallel
- top–down

bottom–up?

deterministisch?

sequentiell?

Tree Walking Automata (TWA)

endliche Zustandsmenge,
nur Lesen, nicht Schreiben,
beginnt in Wurzel,
akzeptiert durch Endzustand

Fragen/Kopfbewegungen:

- READ: lies aktuelles Zeichen
- EXISTS-UP: existiert Vorgänger?
- GO-UP: gehe nach oben
- EXISTS-DOWN(i) existiert Kind Nr. i?
- GO-DOWN(i) gehe zu Kind Nr. i

Befehlssatz nicht ausreichend, deswegen

- AM-I(i): bin ich Kind Nr. i?

jetzt ausreichend für Tiefensuche

Tiefensuche

```
moving := down;
repeat
  case moving of
    down :
         if EXISTS-DOWN (1) then GO-DOWN (1);
         else moving := up;
    up : if      AM-I (1)
         then GO-UP;
              if EXISTS-DOWN (2)
              then moving := down; GO-DOWN (2);
         else if AM-I (2)
         then GO-UP;
              if EXISTS-DOWN (3)
              then moving := down; GO-DOWN (3);
         else if AM-I (3)
         then GO-UP;
until not (AM-I (1) || AM-I (2) || AM-I (3))
```


Auswertung binärer Schaltkreise

Alphabet: $\{0, 1, \cdot\vee\cdot, \cdot\wedge\cdot\}$

$$\text{Egal} \rightarrow 0 \mid 1 \mid \text{Egal} \vee \text{Egal} \mid \text{Egal} \wedge \text{Egal}$$

$$\text{True} \rightarrow 1 \mid \text{True} \vee \text{Egal} \mid \text{Egal} \vee \text{True} \mid \text{True} \wedge \text{True}$$

True $\in$ REG.

True $\in$ TWA.

Schaltkreis ausrechnen

Zustände: `eval, wert-0, wert-1, links-0, links-1`

```
procedure rauf-k =
  zustand := if AM-I(1) then links-k else wert-k;
  GO-UP;

repeat
  case zustand of
    eval : case READ of
          0 : rauf-0;
          1 : rauf-1;
          or, and : GO-DOWN (1);
    wert-k : rauf-k;
    links-0 :
      case READ of
        or  : zustand := eval; GO-DOWN (2);
        and : rauf-0;
    links-1 :
      case READ of
        or  : rauf-1;
        and : zustand := eval; GO-DOWN (2);
```

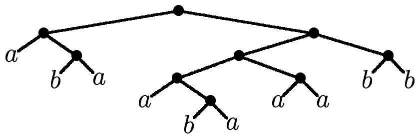
Grenzen von TWA?

Alphabet: $\{a, b,$  $\}$

$L := \{t : \mathsf{yield}(t) \in a^*ba^*\}$

$L \in$ REG, $L \in$ TWA.

$R \to a \mid b \mid$ 

Bsp: 

$\in R$.

$R \in$ REG.

Vermutung (Engelfriet, 1998): $R \notin$ TWA.

Pebble–TWA (pTWA)

Markierungszeichen (Pebble),
zusätzliche Operation/Frage:

- DROP: Pebble ablegen
- LOOK: liegt Pebble hier?
- TAKE: Pebble aufnehmen

$R \in$ pTWA(1).

pTWA(n): Pebbles $1, 2, \ldots, n$
Befehle DROP(i), LOOK(i), TAKE(i)
mit *geschachtelten* Lebensdauern

pTWA $:= \bigcup_{n \geq 0}$ pTWA(n).

Satz (Engelfriet, 1998): pTWA $\subseteq$ REG
Beweis–Idee: Trips on Trees

Vermutung (E.) : alle Inklusionen sind echt:
TWA = pTWA(0) $\subset$ pTWA(1) $\subset \ldots$ pTWA $\subset$ REG.

(Lokale) Logik auf Bäumen

Universum = Adressen im Baum

einstellige Prädikate: $a(p), b(p), \ldots$:
ist Zeichen a auf Adresse p?

zweistellige Prädikate: $p <_1 q, p <_2 q, \ldots$:
das erste (zweite, ...) Kind von p ist q

maximale Arität k ist fixiert,
Abkürzung $p < q \;\equiv\; p <_1 q \lor \ldots \lor p <_k q$

$$\begin{aligned}
\mathsf{blatt}(p) &\equiv \neg\exists q : p < q \\
\mathsf{wurzel}(q) &\equiv \neg\exists p : p < q \\
F_L &\equiv \exists p : \mathsf{blatt}(p) \land b(p) \\
&\qquad \land \forall q : \mathsf{blatt}(q) \land (p \neq q) \to a(p)
\end{aligned}$$

$\mathsf{Mod}(F_L) = L = \{t : \mathsf{yield}(t) \in a^*ba^*\}$.

LFO = lokale Logik erster Ordnung, $\quad L \in$ LFO

Satz (Engelfriet): LFO $\subset$ TWA.
Beweis: Tiefensuche + kleine Umgebungen

FO : Logik erster Ordnung

Prädikat $p \ll q$: p ist Vorgänger von q
$\exists s_1 \ldots s_n : p = s_1 < \ldots < s_n = q$

$\mathsf{rechts}(p) \equiv \forall v : v \ll p \land v \neq p \to \exists r : v <_2 r \land r \ll p$

Relativierung (bezüglich s):

$$\begin{aligned}
F_L^s &\equiv \exists p : \boxed{s \ll p \land} \; (\mathsf{blatt}(p) \land b(p)) \\
&\qquad \land \forall q : \boxed{s \ll q \to} \; (\mathsf{blatt}(q) \land (p \neq q) \to a(p)) \\
F_R &\equiv \forall r : \mathsf{rechts}(r) \land \neg\mathsf{blatt}(r) \to \exists s : r <_1 s \land F_L^s
\end{aligned}$$

$\mathsf{Mod}(F_R) = R, \qquad R \in$ FO.

Satz (Engelfriet): FO $\subseteq$ pTWA.
Beweis–Idee: Quantor $\approx$ Pebble.

bekannt: LFO $\subset$ FO $\subset$ (EMSO =)REG

Mehrheits–Schaltkreise

(Neven, Schwentick, ICALP 2000)

Alphabet $\{0, 1, m(\cdot, \ldots, \cdot)\}$, m ist n–stellig

- $\mathsf{wert}(0) = 0, \mathsf{wert}(1) = 1$
- $\mathsf{wert}(m(t_1, \ldots, t_n)) =$
  if wenigstens die Hälfte der $\mathsf{wert}(t_i) = 1$
  then 1 **else** 0

$T_n^1 = \{t : \mathsf{wert}(t) = 1\}$.

klar: $\forall n : T_n^1 \in \mathsf{REG}$

$T_2^1 = \{t : \mathsf{yield}(t) \in (0+1)^*1(0+1)^*\} \in \mathsf{TWA}$

Def.: rTWA = restricted TWA $\subset$ TWA

Sätze: LFO $\subset$ rTWA, $T_3^1 \notin$ rTWA, $T_3^1 \notin$ FO

Vermutung: $T_3^1 \notin$ TWA, $T_3^1 \notin$ pTWA.

Literatur/Hinweise

- J. Engelfriet, H. J. Hoogeboom: *Tree-walking Pebble Automata*, 1999 `http://www.wi.leidenuniv.nl/home/engelfri/`
- J. Engelfriet, H. J. Hoogeboom, J.-P. van Best: *Trips on Trees*, Acta Cybern. 1999
- F. Neven, T. Schwentick: *On the Power of Tree Walking Automata*, ICALP 2000, `http://www.minet.uni-jena.de/www/fakultaet/inf/index.html`
- F. Neven, T. Schwentick: *Expressive and Efficient Pattern Languages for tree-structured data*

Vortrag am Mittwoch, dem 16. Mai:

Helmut Seidl (Uni Trier):
Baum-Automaten für XML

Die *Heilige Dreifaltigkeit*

diese Bezeichnung © Wolfgang Thomas

- Grammatiken erzeugen
- Automaten akzeptieren
- Logiken beschreiben

Mengen (d. h. formale Sprachen)

von Wörtern, Bäumen, Graphen, . . .