Kontextfreie Grammatiken in AutomataTutor

Bachelorarbeit in Informatik

Martin Helfrich

Technische Universität München

Aufgabensteller: Prof. Javier Esparza
Betreuer: Julia Krämer, Salomon Sickert

12.09.2017

AutomataTutor

- online Übungstool
- Themengebiet: Grundlagen der Automatentheorie
- Aufgabentypen zu DFAs, NFAs und regulären Ausdrücken

AutomataTutor

Vorteile

- individuelles Feedback
- flexibles Arbeiten (Ort / Zeit)
- ressourcenschonende Betreuung der Studierenden

$\rightarrow$ Erweiterung für kontextfreie Grammatiken

neue Aufgabentypen

- *Wörter für Grammatik*
  → Finden von Wörtern innerhalb/außerhalb einer Grammatik
- *Beschreibung nach Grammatik*
  → Konstruieren einer passenden Grammatik
- *Grammatik nach CNF*
  → Umwandlung in Chomsky-Normalform
- *CYK-Algorithmus*
  → Ausfüllen der CYK-Tabelle (Wortproblem)

Algorithmen

- CYK-Algorithmus
- limitierter Äquivalenztest **(neu)**

Frage: $L(G_1) = L(G_2)$
$\rightarrow$ unentscheidbar[1]

Idee:

- Gegenbeispiel suchen
- kürzeste Wörter überprüfen ($\rightarrow$ limitiert)

[1] Uwe Schöning, "Theoretische Informatik kurzgefasst"

Definition (Wörter mit Länge k)

$$L_k(G) = \{w \mid w \in L(G) \wedge |w| = k\}$$

Lemma

$$\bigcup_{k=0}^{\infty} L_k(G) = L(G)$$

Algorithmus 1 : limitierter Äquivalenztest

```
if (ε ∈ G1) ≠ (ε ∈ G2) then return ε;                    // Test ε
k ← 1;
while TERMINATIONCONDITION is false do
    W1 ← generateWordsWithLength(G1, k, dp1);            // W1 = Lk(G1)
    W2 ← generateWordsWithLength(G2, k, dp2);            // W2 = Lk(G2)
    forall w in W1 do                                    // Try: find w ∈ (W1 \ W2)
        if w ∉ W2 then
            return w;
    forall w in W2 do                                    // Try: find w ∈ (W2 \ W1)
        if w ∉ W1 then
            return w;
    k ← k + 1;
```

Funktion generateWordsWithLength(G, l, dp)

if $l = 1$ **then**
    forall $v \in V$ **do** $dp[v][1] \leftarrow \{t \in \Sigma \mid (v \to t) \in P\}$;
else
    forall $(C \in V)$ **do**
        forall $(C \to AB) \in P$ **do**
            for $k = 1$ *to* $l - 1$ **do** // Try all length combinations!
                forall $w_1 \in dp[A][k]$ **do** // $|w_1| = k$
                    forall $w_2 \in dp[B][l-k]$ **do** // $|w_2| = l - k$
                        $dp[C][l] \leftarrow dp[C][l] \cup \{w_1 w_2\}$; // $|w_1 w_2| = l$
return $dp[S][l]$; // return $L_l(G)$

Beispiel: (Klammerausdrücke)

$$S \to SS \mid AB \mid AC \qquad A \to a \qquad B \to b \qquad C \to SB$$

Länge	S	A	B	C
1	$\emptyset$	$\{a\}$	$\{b\}$	$\emptyset$
2	$\{ab\}$	$\emptyset$	$\emptyset$	$\emptyset$
3	$\emptyset$	$\emptyset$	$\emptyset$	$\{aba\}$
4	$\{aabb, abab\}$	$\emptyset$	$\emptyset$	$\emptyset$
5	$\emptyset$	$\emptyset$	$\emptyset$	$\{aabbb, ababb\}$
6	$\{aaabbb, aababb, aabbab, abaabb, ababab\}$	$\emptyset$	$\emptyset$	$\emptyset$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Auswertung

- 15 Teilnehmer
- Aufbau:
  1. Vortest (1 Aufgabe, auf Papier)
  2. Haupttest (2 Aufgaben)
     Kontrollgruppe $\rightarrow$ auf Papier
     Testgruppe $\rightarrow$ mit AutomataTutor
  3. Nachtest (1 Aufgabe, auf Papier)

Auswertung

Ergebnisse

Frage: Übung mit AutomataTutor **“besser”** als auf Papier?
$\rightarrow$ **nicht signifikant**

aber:

- hilft bei anspruchsvollen Aufgaben
- viel positives Feedback
- allgemeine Vorteile von AutomataTutor

Abschluss

Vielen Dank für Ihre Aufmerksamkeit!