Id: titel.tex,v 1.2 2002/10/29 09:51:26 joe Exp

Formalsprachliche Beschreibung von Nachfolgermengen in Wort-Ersetzungs-Systemem

Alfons Geser (ICASE Hampton/VA),
Dieter Hofbauer (Kassel), Johannes Waldmann (Leipzig)

Id: rewrite.tex,v 1.3 2002/11/01 12:39:18 joe Exp

Wort-Ersetzung: außen, innen

Regel-System $R = \{(l_1, r_1), \ldots\}$.

Grammatiken, usw.: ersetze *innen*: $w = ulv \to_i urv = w'$

Büchi-Systeme: ersetze an *einem* Rand: $w = ul \to_r ur = w'$

Post-Systeme: links weg, rechts dran: $w = lu \to_p ur = w'$

Id: nachfolg.tex,v 1.2 2002/10/29 09:51:26 joe Exp

Nachfolgermengen (ein Schritt)

$R(L) := \{w' : \exists w \in L : w \to_R w'\}$

$R^-(L) := \{w' : \exists w \in L : w' \to_R w\}$

Wenn $L \in \mathsf{REG}$, dann $R(L) \in \mathsf{REG}$ für $\to_i$, $\to_r$, $\to_p$.

Beweis: Automaten-Konstruktion.

(Zustände und Pfeile hinzufügen)

Id: nachfolgmehr.tex,v 1.1 2002/10/29 09:51:26 joe Exp

Nachfolgermengen (viele Schritte)

$R^*(L) := \{w' : \exists w \in L : w \to_R^* w'\}$

beschreiben Verlauf (Zwischenschritte) und (nach Schnitt mit Normalformen) Ergebnisse von Rechnungen.

$R^*(L) \cap \mathsf{Err} = \emptyset$: es tritt kein Fehler auf

$R^*(L) \cap \mathsf{Nf}(R) \subseteq \mathsf{Ok}$: Ergebnisse sind Ok.

Rewriting- ($\to_i^*$) und Post-Systeme ($\to_p^*$) simulieren Turing-Maschinen $\Rightarrow$ $R^*(L)$ sind kompliziert.

Reguläre Approximation $\Rightarrow$ Typsysteme ...

Suffix-Rewriting (Büchi-Systeme)

Für Büchi-Systeme $\to_r$ gilt: Wenn $L \in$ REG , dann $R^*(L) \in$ REG.

Verallgemeinerung auf unendliche Regelmengen der Form $A_1 \times B_1 \cup \ldots \cup A_n \times B_n$ für endlich viele Paare $A_i, B_i \in$ REG.

Beweis: passende Epsilon-Übergänge vom Automaten für L zu den Automaten für $B_1, \ldots, B_n$ hinzufügen.

Verallgemeinerung: Ground-Tree-Transducer.

Test von Nachfolgermengen

$R^*(L)$ ist im allgemeinen nicht oder schwer konstruierbar.
Ist es leichter zu verifizieren? (... autotool)

Beispiel: $R = \{110 \to 001, 011 \to 100\}, L = 0^*10^*$.

$R^{-*}(L)$ sind alle Positionen, für die Solitaire lösbar ist.

Satz: falls R^- terminiert, dann ist diese Frage entscheidbar:

Eingabe: R, $L \in$ REG, $N \in$ REG.
Frage: ist $R^*(L) = N$?

Beweis: Aussage ist äquivalent zu $N = R(N) \cup L$.

Nachfolgermengen und Termination

Beobachtung: $L \in$ REG $\wedge$ R terminiert $\not\Rightarrow$ $R^*(L) \in$ REG.

Termination reicht nicht zur Regularitäts-Erhaltung.

Beispiel (Bubblesort): $R = \{ba \to ab\}, L = (ab)^*$.

Dann gilt $R^*(L) \cap a^*b^* = \{a^n b^n : n \geq 0\} \notin$ REG.

REG-Erhaltung bei besonderen Systemen

R ist *speziell*, wenn für alle Regeln $(l \to r) \in R$ gilt: $r = \epsilon$.

R ist *monadisch*, wenn für alle Regeln $(l \to r) \in R$ gilt: $|r| \leq 1$.

Satz (Book, Jantzen, Wrathall 82):
für monadische Systeme bleibt $R^*(L)$ regulär.

Beweis: Automaten-Konstruktion
(Pfeile hinzufügen, aber keine Zustände.)

Id: mono.tex,v 1.3 2002/10/29 09:51:26 joe Exp

Verkürzende Systeme

Definition: R ist *verkürzend* $\iff \forall(l \to r) \in R : |l| > |r|$.

Beobachtung: $L \in$ REG $\wedge$ R verkürzend $\not\Rightarrow R^*(L) \in$ REG.

Beispiel (Otto 96): $R = \{acc \to_1 ca, bcc \to_2 cb, bac \to_3 ab\}$.

Für alle $u, v \in \{a, b\}^*$ gilt $u\ v\ c^{2^n} \to_{1,2}^* u\ c\ v$.

$R^*((ab)^*c^*) \cap a^*b^*c^* = \{a^nb^n : n \geq 0\}c^* \notin$ REG.

Id: bound.tex,v 1.3 2002/11/01 12:39:18 joe Exp

Beschränkte Systeme (Beispiel)

Solitaire $R = \{110 \to 001, 011 \to 100\}$

Beobachtung: Jede Position wird höchstens viermal geändert.

Satz (Ravikumar 97): wenn R längen-erhaltend und änderungs-beschränkt, dann $L \in$ REG $\Rightarrow R^*(L) \in$ REG.

Anwendung: lösbare Positionen $R^{-*}(0^*10^*) \in$ REG.

Id: forward.tex,v 1.4 2002/11/01 16:56:43 alfons Exp

Forward Closures

Definition (Lankford, Musser 78): FC(R) $\subseteq \Sigma^* \times \Sigma^*$:

- $R \subseteq$ FC(R)
- $(u, v) \in$ FC(R) $\wedge (l, r) \in R \wedge w \in$ OVL(v, l) $\Rightarrow$
  $(u \cdot (l$ nach $w), (v$ vor $w) \cdot r) \in$ FC(R)
- $(u, v) \in$ FC(R) $\wedge (v \to_R v') \Rightarrow (u, v') \in$ FC(R)

Definition RFC(R) $= \{v : (u, v) \in$ FC(R)$\}$.

Id: anwend.tex,v 1.4 2002/11/01 16:56:43 alfons Exp

Forward Closures — Termination und Schleifen

Satz (Dershowitz 81):
wenn es ein $w \in \Sigma^*$ mit unendlicher Reduktion gibt,
dann auch ein $w' \in$ RFC(R) mit unendlicher Reduktion.

Beweis-Idee:
betrachte den am weitesten linken Redex.

Folgerung: R terminiert (auf Σ^*) $\iff$ R terminiert auf RFC(R).

Satz (Geser, Zantema 99):
R hat Schleife $\iff \exists(u, v) \in$ FC(R) : $(u \to^* v)$ ist Schleife.

Beweis-Idee: falls in w „unbenutzte“ Stellen, dann wähle Teilwort als Start. Eliminiere nicht-FC Überlappungen.

Id: term.tex,v 1.2 2002/10/29 09:51:26 joe Exp

Closures, REG, und Termination

Definition: right closures RC(R):
wie FC, aber ohne „innere“ Reduktionen.

RRC(R) $:= \{v : (u, v) \in$ RC(R)$\}$.

Satz: RRC(R) $\in$ REG. Beweis: Büchi-System.

Fakt: RRC(R) $\subseteq$ Nf(R) $\iff$ RFC(R) $\subseteq$ Nf(R)

Folgerung: $\ldots \Rightarrow R$ terminiert.

Id: todo.tex,v 1.1 2002/10/29 09:51:26 joe Exp

Erweiterungen (geplant)

Für Forward Closures:

- Wie weit kann man RFC(R) $\not\subseteq$ RRC(R) tolerieren?
- Wie kann man RFC(R) aus RRC(R) konstruieren?
  (Antwort: durch innere Reduktionen.
  Aber erhalten diese die Regularität?)

Allgemein:

- Zusammenhänge zwischen Termination und REG-Erhaltung
- Wort-Ersetzung $\Rightarrow$ Term-Ersetzung

Id: mark.tex,v 1.1 2002/11/01 12:39:18 joe Exp

Ansatz: Markierte Reduktionen

statt R auf Σ betrachte R_B auf $\Sigma \times \{0, \ldots, B\}$

Reduktionsschritt: wenn $(l, r) = (l_1 \ldots l_m, r_1 \ldots r_n)$,
dann $w = u(l_1, b_1) \ldots (l_m, b_m)v \to u(r_1, b) \ldots (r_n, b)v$

mit neuer Markierung $b = 1 + \min\{b_1, \ldots, b_m\}$.

System R heißt *beschränkt*, falls
$\exists B$ mit $\forall u \to_R^* v : u \times 0 \to_{R_B}^* v'$ so daß Proj$(v') = v$.

Fakt: R beschränkt $\Rightarrow R$ terminiert.

Vermutung: R beschränkt $\wedge\, L \in$ REG $\Rightarrow R^*(L) \in$ REG.

Id: delete.tex,v 1.1 2002/11/01 12:39:18 joe Exp

Ansatz: Löschende Systeme

R heißt *löschend*, falls $\exists c \in \Sigma : \forall (l \to r) \in R : |l|_c > |r|_c = 0$.

Vermutung: R löschend $\Rightarrow R$ erhält Regularität.

Beachte: $|l|_c > |r|_c$ reicht nicht.
Siehe früheres Beispiel $\{acc \to_1 ca, bcc \to_2 cb, bac \to_3 ab\}$.

Zusammenhang: bei beschränkten Systemen
werden Buchstaben $(c, \min\{b_1, \ldots, b_m\})$ gelöscht.