Korrektheit des Vervollständigungsverfahrens und Huets Algorithmus

Vortrag von Favio Miranda
Proseminar Termersetzung
Betreuer: Dr. Ralph Matthes
Institut für Informatik, Universität München

31.01.2001

Die Vervollständigung beschäftigt sich, wie schon erwähnt, mit folgendem Problem:

Eingabe: ein Gleichungssystem E.

Gesucht: ein konvergentes Termersetzungssystem R, äquivalent zu E.

Im Abschnitt 7.1 wurde der Grundlegende Vervollständigungsalgorithmus (GVA) vorgestellt. Dieses GVA ist sehr ineffizient weil die Mengen R_i zu groß werden. (Es gilt $R_i \subseteq R_{i+1}$).
Im Abschnitt 7.2 wurde ein verallgemeinertes Vervollständigungsverfahren vorgestellt, das auf einem Inferenzsystem C basiert.

Ziel dieses Vortrags ist die Korrektheit eines fairen Vervollständigungsverfahrens zu zeigen (Satz 7.2.8), und Huets Vervollständigungsverfahren vorzustellen.

7.3 Beweisordnungen

Der Beweis der Korrektheit ist komplizierter als der für den GVA, wegen die Regeln $L - Simp$ und $R - Simp$, besonders für unendliche Läufe.
In einem fairen Lauf, der nicht fehlschlägt, werden alle kritische Paare berechnet, weil $CP(R_\omega) \subseteq \bigcup_{i\geq 0} E_i$ gilt.
Diese kritische Paare sind zusammenführbar bzgl. der Menge $\bigcup_{i\geq 0} R_i$ aller berechneten Regeln, weil $E_\omega = \varnothing$ gilt.
Aber es ist nicht klar, daß die kritische Paare auch zusammenführbar bzgl. R_ω sind. In ähnlicher Weise, obwohl die Mengen $E_i \cup R_i$ äquivalent mit E_0 sind, ist es nicht klar, daß R_ω äquivalent mit E_0 ist.

Ein praktisches Instrument um die Korrektheit eines beliebigen Vervollständigungsverfahrens zu beweisen sind die sogenannten *Beweisordungen*. Darüber sprechen wir in diesem Abschnitt, aber erstens wird der Begriff *Beweis* definiert.

In folgenden nehmen wir einmal an, daß

$$(E_0, \varnothing) \vdash_C (E_1, R_1) \vdash_C (E_2, R_2) \vdash_C (E_3, R_3) \vdash_C \ldots$$

ein beliebiger, fixierter, fairer Lauf eines Vervollständigungsverfahrens ist, der nicht fehlschlägt.
Wir definieren:

$$E_\infty = \bigcup_{i\geq 0} E_i \quad R_\infty = \bigcup_{i\geq 1} R_i$$

Definition 7.3.1 Ein *Beweis* einer Gleichung $s \approx t$ bzgl. $E_\infty \cup R_\infty$ ist eine endliche Folge $(s_0, \ldots, s_n)$ mit $s_0 = s, s_n = t$ und für alle $1 \leq i \leq n$

1. $s_{i-1} \longleftrightarrow_{E_\infty} s_i$, oder
2. $s_{i-1} \longrightarrow_{R_\infty} s_i$, oder
3. $s_i \longrightarrow_{R_\infty} s_{i-1}$

Für $i = 1, \ldots, n$ heißen die Paare (s_{i-1}, s_i) *Beweisschritte.*
Zwei Beweise heißen *äquivalent* wenn sie die selbe Gleichung beweisen.
Ein Beweis $(s_0, \ldots, s_n)$ bzgl. $E_\infty \cup R_\infty$ heißt *V-Beweis*[1] bzgl. R_ω genau dann wenn es ein k, $0 \leq k \leq n$ gibt, mit $s_{i-1} \longrightarrow_{R_\omega} s_i$ für alle $1 \leq i \leq k$ und $s_i \longleftarrow_{R_\omega} s_{i+1}$ für alle $k \leq i \leq n$.

Beweise lassen sich vergleichen durch die Multimengenerweiterung einer lexikographische Ordnung, die durch die Multimengenerweiterung der Reduktionsordnung $>$, den strikte Anteil der Umschließungsordnung $\sqsupset$ und die Reduktionsordnung $>$ definiert wird.

Definition 7.3.2 Sei $\succ_C$ diese sogenannte *Beweisordnung.*

Lemma 7.3.3 *$\succ_C$ ist eine wohlfundierte Ordnung.*

Lemma 7.3.4 *Sei P ein Beweis bzgl. $E_\infty \cup R_\infty$ der kein V-Beweis ist. Dann gibt es einen Beweis P' bzgl. $E_\infty \cup R_\infty$ so daß gilt: P' äquivalent mit P und $P \succ_C P'$.*

Satz 7.3.5 *Sei $(E_0, \varnothing) \vdash_C (E_1, R_1) \vdash_C (E_2, R_2) \vdash_C (E_3, R_3) \vdash_C \ldots$ ein fairer Lauf eines Vervollständigungsverfahrens, der nicht fehlschlägt. Dann gilt*

1. *Jeder Beweis bzgl. $E_\infty \cup R_\infty$ ist äquivalent mit eine V-Beweis bzgl. R_ω.*
2. *R_ω ist äquivalent mit E_0.*
3. *R_ω ist konvergent.*
4. *Wenn R_ω endlich ist, dann ist das Wortproblem für E_0 entscheidbar. Sonst liefert der Lauf eine Semi-Entscheidungsprozedur für $\approx_{E_0}$.*

Als Korollar zeigen wir den Satz 7.2.8:

Korollar 7.3.6 (Satz 7.2.8) *Jedes faire Vervollständigungsverfahren ist korrekt.*

7.4 Huets Vervollständigungsverfahren

Huet war der erste, der einen Beweis der Korrektheit eines Vervollständigungsverfahren angab, das die Vereinfachung der Regeln erlaubt (siehe [Hue81]). In diesem Abschnitt stellen wir Huets Vervollständigungsverfahren als ein konkretes Beispiel eines korrekten Vervollständigungsverfahrens vor.

Die folgenden Lemmas zeigen die Korrektheit des Huet-Verfahrens unter der Voraussetzung, daß es eine bestimmte Strategie benutzt, um die kritischen Paare zu berechnen. Außerdem schlägt ein Lauf dieses Verfahrens fehl genau dann wenn das Verfahren mit Ausgabe `Fail` terminiert.

Lemma 7.4.1 *Huets Vervollständigungsverfahrens ist eine Vervollständigungsverfahren im Sinne der Definition 7.2.3.*

[1] Auf Englisch *rewrite proof.*

Lemma 7.4.2 *Ein Lauf von Huets Vervollständigungsverfahren schlägt fehl genau dann wenn das Verfahren mit Ausgabe* `Fail` *terminiert.*

Lemma 7.4.3 *Sei ein Lauf von Huets Prozedur gegeben ohne unmarkierte persistente Regeln. Sei weiter* $s_{i-1} \longleftarrow_{R_\omega} s_i \longrightarrow_{R_\omega} s_{i+1}$ *eine Spitze auf Grund von einer kritischen Überlappung. Dann gibt es Terme* s'_{i-1} *und* s'_{i+1} *so daß* $s_i \longrightarrow_{R_\infty} s'_{i-1} \longrightarrow^\star_{R_\infty} s_{i-1}$, $s_i \longrightarrow_{R_\infty} s'_{i+1} \longrightarrow^\star_{R_\infty} s_{i+1}$ *und* $s'_{i-1} \longleftrightarrow_{E_\infty} s'_{i+1}$.

Satz 7.4.4 *Huets Prozedur ist ein korrektes Vervollständigungsverfahren, wenn wir ausschließen können, daß es unmarkierte persistente Regeln gibt. Dies ist möglich, wenn eine passende Strategie verwendet wird zur Auswahl der unmarkierten Regeln, für die die kritischen Paare berechnet werden.*

Literatur

[Ave95] J. Avenhaus. *Reduktionssysteme.* Springer Verlag 1995.

[BaNi98] F. Baader, T. Nipkow. *Term Rewriting and All That.* Cambridge University Press. 1998.

[Hue81] G. Huet. *A complete proof of correctness of the Knuth-Bendix completion algorithm.* Journal of Computer and System Sciences. 23(1):11-21. 1981.

Verfahren 1 Huets Vervollständigungsverfahren

Eingabe: Eine endliche Menge E von Σ-Gleichungen und eine Reduktionsordnung $>$ auf $T(\Sigma, V)$.

Ausgabe: Ein endliches konvergentes Termersetzungssystem R_i, das äquivalent zu E ist, wenn die Prozedur mit Erfolg terminiert; "`Fail`", wenn die Prozedur mit Fehlschlag terminiert. Wenn die Prozedur nicht terminiert, generiert sie ein unendliches Limessystem R_ω, das konvergent und äquivalent zu E ist.

$R_0 := \varnothing; E_0 := E; i := 0$
while $E_i \neq \varnothing$ oder es gibt eine unmarkierte Regel in R_i **do**
  while $E_i \neq \varnothing$ **do**
    Wähle eine Gleichung $s \approx t \in E_i$ aus;
    Reduziere s, t zu R_i-Normalformen $\widehat{s}, \widehat{t}$;
    if $\widehat{s} = \widehat{t}$ **then**
      $R_{i+1} := R_i$;
      $E_{i+1} := E_i - \{s \approx t\}$;
      $i := i + 1$;
    else
      if $\widehat{s} \not> \widehat{t}$ und $\widehat{t} \not> \widehat{s}$ **then**
        Terminiere mit Ausgabe `Fail`;
      else
        Sei l, r so daß gilt $\{l, r\} = \{\widehat{s}, \widehat{t}\}$ und $l > r$;
        $R_{i+1} := \{g \longrightarrow \widehat{d} \mid g \longrightarrow d \in R_i,\ g$ lässt sich nicht reduzieren mit $l \longrightarrow r$ und $\widehat{d}$
                ist eine Normalform von d bzgl. $R_i \cup \{l \longrightarrow r\}\}$
            $\cup \{l \longrightarrow r\}$; /* $g \longrightarrow \widehat{d}$ *erbt die Marke von* $g \longrightarrow d$ */
                /* $l \longrightarrow r$ *ist nicht markiert* */
        $E_{i+1} := (E_i - \{s \approx t\}) \cup$
            $\{g' \approx d \mid g \longrightarrow d \in R_i$ und g lässt sich reduzieren zu g' mit $l \longrightarrow r\}$;
        $i := i + 1$;
      end if
    end if
  end while/* *Ende der inneren while-Schleife* */
  if es gibt eine unmarkierte Regel in R_i **then**
    Sei $l \longrightarrow r$ solche Regel.
    $R_{i+1} := R_i$;
    $E_{i+1} := \{s \approx t \mid \langle s, t\rangle$ ist ein kritisches Paar von $l \longrightarrow r$ mit sich selbst
                oder mit einer markierten Regel in $R_i\}$;
    $i := i + 1$;
    Markiere $l \longrightarrow r$;
  end if
end while/* *Ende der äußere while-Schleife* */
Gib R_i aus.