Theorie der Programmierung SS 2015

Übungsblatt 1

Rev.8357

Abgabe der Lösungen: Tutorium in der Woche 11.05-15.05

Übung 1 Aufwärmübung

Es sei $\Sigma = \{A, B, C, D, \cdot\}$ eine Signatur mit vier Konstanten A, B, C und D sowie einem binären Funktionssymbol $\cdot$, welches wir als Infix-Operator verwenden (d.h., wir schreiben $s \cdot t$ anstelle von $\cdot(s, t)$). Wir betrachten das folgende Termersetzungssystem:

$$A \cdot x \to_0 B \cdot (C \cdot x) \tag{1}$$
$$C \cdot (D \cdot x) \to_0 B \cdot (C \cdot x) \tag{2}$$
$$B \cdot (x \cdot y) \to_0 A \cdot (D \cdot x) \tag{3}$$
$$B \cdot (B \cdot x) \to_0 D \cdot x \tag{4}$$

Zeigen Sie, daß dieses System nicht stark normalisierend ist, indem Sie einen Σ-Term t sowie eine unendliche Derivation $t \to t_1 \to t_2 \to t_3 \ldots$ angeben.

Übung 2 Applikative vs. normale Reduktion

Wir betrachten das folgende funktionale Programm (gegeben in Haskell-Syntax):

```
1  data List a = Nil | Cons a (List a)
2
3  error = error
4
5  head Nil = error
6  head (Cons a as) = a
7
8  tail Nil = Nil
9  tail (Cons a as) = as
```

1. Drücken Sie die für dieses Programm möglichen Transitionen als Termersetzungssystem über der Signatur $\Sigma = \{Nil, Cons, head, tail, error\}$ aus.

2. Identifizieren Sie alle Redexe in dem folgenden (als Σ-Term zu interpretierenden) Ausdruck:

$$head\ (Cons\ (head\ (Cons\ (tail\ Nil)\ (head\ Nil)))\ (Cons\ (tail\ Nil)\ error))$$

3. Die *Auswertungsstrategie* einer Programmiersprache ist der Teil ihrer Semantik, der die Reihenfolge festlegt, in welcher Teile von zusammengesetzten Ausdrücken bzw. Aussagen berechnet werden. Eine Möglichkeit, dies zu formalisieren, ist die Angabe einer *Reduktionsstrategie*, das heißt einer Festlegung, welcher der Redexe eines gegebenen Terms als nächstes reduziert werden soll.

(a) Die bei Programmiersprachen am weitesten verbreitete Auswertungsstrategie ist die sogenannte *strikte* (engl. *eager* oder *strict*) Auswertung. Sie entspricht üblicherweise der *leftmost-innermost-Reduktion*, die auch als *applikative Reduktion* bezeichnet wird: Es wird jeweils der erste Redex reduziert, der bei einer *post-order*-Traversierung des Term-Baumes auftritt. Zeigen Sie, daß der obige Ausdruck unter leftmost-innermost-Reduktion divergiert.

(b) Einige Programmiersprachen (darunter auch Haskell) verwenden eine nicht-strikte Auswertungsstrategie. Eine solche Strategie entsteht bespielsweise bei Verwendung der *lefmost-outermost-Reduktion* (auch als *normale Reduktion* bezeichnet), bei welcher der erste bei einer *pre-order*-Traversierung des Term-Baumes auftretende Redex reduziert wird. Zeigen Sie, daß der obige Ausdruck unter leftmost-outermost-Reduktion zu einer Normalform reduziert.

Übung 3 SKI

Es sei $\Sigma = \{I, K, S, \cdot\}$ für die drei Konstantensymbole I, K und S sowie den binären Infix-Operator $\cdot$. Wir betrachten das folgende Termersetzungssystem:

$$I \cdot x \to_0 x \tag{5}$$
$$(K \cdot x) \cdot y \to_0 x \tag{6}$$
$$((S \cdot x) \cdot y) \cdot z \to_0 (x \cdot z) \cdot (y \cdot z) \tag{7}$$

Aus Gründen der Lesbarkeit schreiben wir die folgenden Terme ohne den Operator $\cdot$ und entfernen nach links geschachtelte Klammern. Beispielsweise schreiben wir die letzte Reduktionsregel als $Sxyz \to_0 (xz)(yz)$. Entscheiden Sie, welche der folgenden Terme durch applikative bzw. normale Reduktion zu einer Normalform reduziert werden:

1. $SII(SII)$ (wir nennen diesen Term Ω)
2. $SK\Omega a$ (a ist in Normalform)
3. $S(SK)(SK)(KKK)$
4. $S\ (K(SI))\ K\ (SII)\ (SK(SII))$ (die eingefügten Freizeichen dienen nur der Lesbarkeit)

Übung 4 Terminationsbeweis mittels Polynomordnung

Wir betrachten das folgende Termersetzungssystem für $\Sigma = \{\oplus, \odot\}$:

$$x \odot (y \oplus z) \to_0 (x \odot y) \oplus (x \odot z) \tag{8}$$
$$(x \oplus y) \oplus z \to_0 x \oplus (y \oplus z) \tag{9}$$

Wir möchten mithilfe von Polynomordnungen zeigen, daß dieses System normalisierend ist, und verwenden dazu die folgenden Polynome:

$$p_\odot(x, y) := xy$$
$$p_\oplus(x, y) := 2x + y + 1$$

Es genügt nun, eine geeignete Domäne (in diesem Fall also eine Teilmenge von $\mathbb{N}$) für die Ordnung zu finden und sicherzustellen, daß die Reduktionsregeln bezüglich der Ordnung absteigend sind.

1. Es sei $\mathcal{A} = \langle \mathbb{N}, p_\odot, p_\oplus \rangle$ die einfachste auf $p_\odot$ und $p_\oplus$ basierende polynomielle Interpretation von Σ. Begründen Sie, daß die von $\mathcal{A}$ induzierte Polynomordnung $\succ_\mathcal{A}$ nicht geeignet ist, um die Termination des Systems zu zeigen.

2. Finden Sie eine geeignete Domäne $B \subset \mathbb{N}$ für welche die durch die Interpretation $\mathcal{B} = \langle B, p_\odot, p_\oplus \rangle$ induzierte Polynomordnung $\succ_\mathcal{B}$ die Termination des Systems zeigt.

3. Wir ersetzen die zweite Reduktionsregel durch

$$x \oplus (y \oplus z) \to_0 (x \oplus y) \oplus z \tag{10}$$

   Zeigen Sie unter Verwendung einer Polynomordnung, daß das so erhaltene System ebenfalls terminierend ist.

Übung 5 Ein weiterer Beweis mittels Polynomordnung

Wir betrachten das folgende Termersetzungssystem für $\Sigma = \{f, a, b\}$:

$$f(f(x, y), z) \to_0 f(x, f(y, z)) \tag{11}$$
$$f(y, f(x, z)) \to_0 f(x, x) \tag{12}$$
$$f(a, b) \to_0 a \tag{13}$$

Zeigen Sie mithilfe einer Polynomordnung, daß dieses System terminierend ist.

Hinweis: Das Polynom $x^2 + xy$ könnte hilfreich sein!

Übung 6 Woher kommen diese Polynome?

Wir betrachten das folgende in Haskell-Syntax gegebene Programm:

```
1   data Nat = Z | S Nat
2
3   x + Z = x
4   x + (S y) = S (x + y)
5
6   d Z = Z
7   d (S x) = S (S (d x))
8
9   q Z = Z
10  q (S x) = q x + S (d x)
```

1. Drücken Sie dieses Programm als Termersetzungssystem über der Signatur $\Sigma = \{0, s, +, d, q\}$ aus.

2. Verwenden Sie eine Polynomordnung um zu beweisen, daß das System normalisierend ist. *Hinweis:* Wählen Sie $p_s(x) := x + 1$ und leiten Sie hieraus geeignete Werte für p_+, p_d, p_q und p_0 her.

3. Können wir also schließen, daß jedes aus den obigen Funktionen zusammengesetze Programm terminierend sein wird? Wird die Termination eines solchen Programmes davon abhängen, ob eine strikte oder eine nicht-strikte Auswertungstrategie verwendet wird?

Übung 7 Konfluenz-Quiz

Entscheiden Sie, ob die folgenden Aussagen jeweils wahr oder falsch sind und *begründen* Sie ihre Entscheidung entsprechend.

1. „Jedes terminierende TES ist konfluent.“
2. „Es gibt ein nicht-terminierendes TES, das konfluent, aber nicht lokal konfluent ist.“
3. „Es gibt ein nicht-terminierendes TES, das lokal konfluent, aber nicht konfluent ist.“
4. „Wenn ein TES terminierend ist, dann ist es konfluent genau dann wenn es lokal konfluent ist.“

Übung 8 Kritische Paare berechnen

Wir betrachten erneut das Termersetzungssystem aus Übung 1. Bestimmen Sie nun alle kritischen Paare des Systems und geben Sie dazu jeweils die involvierten Regeln sowie den entsprechenden allgemeinsten Unifikator an. (Achten Sie hierbei darauf, Variablen nötigenfalls umzubenennen, um unbeabsichtigte Namensgleichheit zu vermeiden!)

Übung 9 Konfluenz mittels Newmans Lemma

Wir betrachten erneut das Termersetzungssystem aus Übung 4. Nun möchten wir zeigen, daß es konfluent ist. Da wir bereits gezeigt haben, daß das System terminierend ist, genügt es nach Newmans Lemma zu zeigen, daß es *lokal konfluent* ist. Ihre Aufgabe ist es also, die lokale Konfluenz des Systems zu zeigen, d.h. alle kritischen Paare des Systems zu bestimmen und anschliessend für jedes Paar zu zeigen, daß es zusammengeführt werden kann.

Übung 10 Ein nicht-konfluentes System

Zeigen Sie, daß das System aus Übung 5 *nicht* konfluent ist. *Hinweis:* Das System ist nicht einmal lokal konfluent. Wir suchen also ein kritisches Paar, das nicht zusammengeführt werden kann.

Übung 11 *Hausaufgabe I* *(20 Punkte)*

Es sei Σ ein Vokabular der Prädikatenlogik erster Stufe, d.h. es sei $\Sigma \supseteq \{\neg, \wedge, \vee, \Rightarrow, \exists, \forall\}$, wobei wir vereinbaren, daß $\neg$ unär ist und die restlichen Operatoren binär sind. Wir verwenden $\exists x.y$ als abkürzende Schreibweise für $\exists(x, y)$. Wir betrachten das folgende Termersetzungssystem:

$$\begin{aligned}
\neg\neg x &\to_0 x \\
(x \wedge y) &\to_0 \neg(\neg x \vee \neg y) \\
\neg(x \wedge y) &\to_0 \neg x \vee \neg y \\
(x \Rightarrow y) &\to_0 (\neg x \vee y) \\
x \vee (y \vee z) &\to_0 (x \vee y) \vee z \\
\forall x.y &\to_0 \neg(\exists x.\neg y)
\end{aligned}$$

1. Zeigen Sie mittels einer Polynomordnung, daß das obige Systen terminiert.

   Hinweis: Wählen Sie $p_{\neg}(x) := x + 1$.

2. Zeigen Sie, daß das System konfluent ist.

3. Wenn wir mit der Reduktion bei einem Term beginnen, der eine wohlgeformte Formel repräsentiert, was wird die Normalform des Terms repräsentieren?

Übung 12 *Hausaufgabe II* *(20 Punkte)*

Wir betrachten das folgende Termersetzungssystem über der Signatur $\Sigma = \{f, g\}$:

$$f(f(x)) \to_0 g(x) \tag{14}$$
$$g(g(x)) \to_0 g(x) \tag{15}$$
$$f(g(x)) \to_0 f(x) \tag{16}$$
$$g(f(x)) \to_0 f(g(x)) \tag{17}$$

1. Zeigen Sie mittels einer Polynomordnung, daß dieses System terminierend ist. Zeigen Sie weiter, daß es konfluent ist.

2. Es sei t ein beliebiger nichttrivialer Term (d.h. $t \notin V$). Wie sieht die Normalform von t aus? Begründen Sie Ihre Antwort.