Universität
des
Saarlandes
FR Informatik

MPI Informatik

Harald Ganzinger
Andreas Podelski
Uwe Waldmann

2. Mai 2001

Übungen zur Vorlesung „Programmiersprachen“
Übungsblatt 4

Programmieraufgabe 4.1: (*2 P.*)
Wir betrachten die Menge aller Terme der Logik erster Stufe, die aus Funktionssymbolen f und Variablen x aufgebaut sind. Diese Terme können wir mittels der folgenden ML-Datentypen repräsentieren:

```
datatype function = F of string;

datatype term = V of string
              | @@ of function * term list;
infix @@;
```

Dabei bezeichnet der infix-Operator `@@` die Anwendung eines Funktionssymbols auf eine Liste von Argumenten. (Terme sind also hier *nicht* curryfiziert.) Konstanten sind Funktionssymbole mit leerer Argumentliste. Der Term `f(a,g(x))` würde beispielsweise durch

```
F "f" @@ [F "a" @@ [], F "g" @@ [V "x"]]
```

codiert. Substitutionen sind Listen von Paaren, die jeweils aus einem Variablennamen und einem Term bestehen, also beispielsweise

```
[("x",F "a" @@ []),("y",F "g" @@ [V "z"])]
```

für die Substitution $[a/x, g(z)/y]$.

Schreiben Sie eine Funktion `matchingSubst : term -> term -> (string * term) list`, die zwei Argumente t und t' bekommt und die Anpaßsubstitution für t auf t' liefert, das heißt, diejenige Substitution σ mit $t\sigma = t'$. Sie können davon ausgehen, daß t linear ist, also daß keine Variable mehrfach in t vorkommt. Falls keine Anpaßsubstitution für t auf t' existiert, so liefert `matchingSubst` eine Exception `nomatch`.

Programmieraufgabe 4.2: (*2 P.*)
Schreiben Sie eine Funktion `applySubst : (string * term) list -> term -> term`, die zwei Argumente σ und t bekommt und $t\sigma$ liefert.

Programmieraufgabe 4.3: *(2 P.)*
Wir repräsentieren eine Termersetzungsregel $l \to r$ durch ein Paar `(l,r)` und ein Reduktionssystem durch eine Liste von solchen Paaren. Implemetieren Sie eine Funktion `applyRulesOnceAtTop : (term * term) list -> term -> term`, die ein Reduktionssystem R und einen Term t als Argumente bekommt und die erste passende der Regeln aus R *einmal* an der Spitze auf t anwendet. Das bedeutet, falls $l \to r$ die erste Regel aus R mit $l\sigma = t$ ist, so liefert `applyRulesOnceAtTop` den Term $r\sigma$. Falls keine solche Regel existiert, liefert `applyRulesOnceAtTop` eine Exception `topirreducible`. Sie können davon ausgehen, daß die linken Seiten der Regeln in R linear sind.

Programmieraufgabe 4.4: *(9 P.)*
Schreiben Sie eine Funktion `normalize : (term * term) list -> term -> term`, die ein Reduktionssystem R und einen Term t als Argumente bekommt und eine Normalform von t unter der Relation $\to_R$ berechnet. Sie können davon ausgehen, daß die linken Seiten der Regeln in R linear sind, und daß $\to_R$ terminierend ist; falls $\to_R$ nicht terminiert, so braucht auch `normalize` nicht zu terminieren.

Verwenden Sie die Strategie „innerste Redexe zuerst". Beachten Sie, daß die Anwendung einer Regel weitere Redexe *unterhalb* des aktuellen Redex produzieren kann; beispielsweise wird nach Anwendung der ersten Regel von $R = \{f(g(x)) \to f(h(x)),\, h(a) \to h(b)$ an der Spitze von $f(g(a))$ die zweite Regel unterhalb der Spitze anwendbar.

Schreiben Sie die Funktion `normalize` so, daß die Datenstruktur des zu normalisierenden Terms soweit möglich nicht kopiert, sondern wiederverwendet wird. Wird zum Beispiel eine Regel $f(x) \to g(x)$ auf einen Term $h(g(a), f(b))$ angewendet, dann muß der Redex $f(b)$ durch einen „neuen" Term $g(b)$ ersetzt werden, und auch die Funktionssymbole auf dem Pfad von der Wurzel zum Redex (hier h) müssen neu angelegt werden; Teilterme jedoch, die sich mit dem Redex nicht überschneiden (hier $g(a)$) können ohne Kopieren in den neuen Term eingebaut werden. Benutzen Sie dazu eine Hilfsfunktion `normalize'`, die, falls t reduzibel ist, eine Normalform von t und sonst eine Exception liefert, sowie eine weitere Hilfsfunktion `normalizeList'`, die analog zu `normalize'` auf Listen von Termen operiert.

Aufgabe 4.5: *(4 P.)*
Gegeben seien eine Funktion f und ein Element e, so daß für alle x, y, z:

$$f(x, f(y, z)) = f(f(x, y), z)$$
$$f(e, x) = x$$

Zeigen Sie, daß für alle l und x:

$$f(\mathit{foldr}\ f\ e\ l,\ x) = \mathit{foldr}\ f\ x\ l$$

Schicken Sie die ausgetesteten Lösungen der Programmieraufgaben (inklusive einiger Testbeispiele mit Ergebnissen) bis zum 9. 5., 13:00 Uhr per Mail an Ihren Übungsgruppenleiter, Subject: `PS Blatt 4`. Bereiten Sie die übrigen Aufgaben so vor, daß Sie die Lösung in der nächsten darauffolgenden Übungsstunde vortragen können.