INFORMATIK

Universität
des
Saarlandes
FR Informatik

Harald Ganzinger
Andreas Podelski
Uwe Waldmann

18. April 2001

Übungen zur Vorlesung „Programmiersprachen"
Übungsblatt 2

Programmieraufgabe 2.1: (*3 P.*)
Schreiben Sie in ML eine Sortierfunktion `sort` mit dem Typ `('a -> 'a -> bool) -> 'a list -> 'a list`. Das erste Argument von `sort` ist eine totale Ordnung *smaller*, das zweite Argument die Liste, die gemäß *smaller* sortiert werden soll (d. h., falls *smaller* x y = `true`, dann steht x in der sortierten Liste vor y). Verwenden Sie das Mergesort-Verfahren: Listen mit weniger als zwei Elementen sind sortiert, Listen $[x_1, x_2, \ldots, x_n]$ mit mindestens zwei Elementen werden zunächst in zwei Teillisten $[x_1, x_3, \ldots]$ und $[x_2, x_4, \ldots]$ aufgespalten, dann werden beide Listen rekursiv sortiert und die sortierten Listen gemäß *smaller* zusammengemischt.

Programmieraufgabe 2.2: (*2 P.*)
Schreiben Sie eine ML-Funktion `uniq`, die aufeinanderfolgende Duplikate aus einer Liste entfernt.

Beispiel:

```
uniq  ["a","a","b","c","c","c","a"]
```

liefert das Ergebnis `["a","b","c","a"]`.

Programmieraufgabe 2.3: (*5 P.*)
Sei $T = \{\tau\}$ und $\Sigma = \{a : \to \tau,\ b : \to \tau,\ f : \tau \to \tau,\ g : \tau \times \tau \to \tau\}$. Wir repräsentieren Variable durch Strings und Terme über Σ und Variablen durch den folgenden ML-Datentyp:

```
datatype term = var of string | a | b | f of term | g of term * term;
```

Implementieren Sie folgende Funktionen in ML:

- `vars : term -> string list` berechnet die Liste aller Variablen(-namen), die in einem Term vorkommen (ohne Duplikate).
- `isground : term -> bool` ermittelt, ob ein Term ein Grundterm (d. h., variablenfrei) ist.

- `islinear : term -> bool` ermittelt, ob ein Term linear ist, d. h., ob jede Variable in dem Term höchstens einmal vorkommt.
- Positionen in Termen kann man formal als Listen (oder Worte oder Strings) natürlicher Zahlen definieren. Das oberste Funktionssymbol eines Terms hat die Position [ ]; falls ein Symbol in t_i die Position p hat, dann hat es in $f(t_1, \ldots, t_i, \ldots, t_n)$ die Position $i :: p$.

  Die Funktion `positionsof : term -> term -> int list list` berechnet die Liste aller Positionen p, an denen das erste Argument t im zweiten Argument t' als Subterm vorkommt (d. h., $t'/p = t$).

(Sie können die Funktionen `sort` und `uniq` aus den vorherigen Aufgaben und die vordefinierte Funktion `map` verwenden.)

Aufgabe 2.4: (*3 P.*)
Unter welcher Bedingung ist die folgende Definition in ML zulässig und was bewirkt sie dann?

```
val f = fn [] => 0 | (_::l) => 1 + f(l);
```

Aufgabe 2.5: (*3 P.*)
Parallele Wertdefinitionen

`val` x_1 `=` e_1 `and` $\ldots$ `and` x_n `=` e_n

kann man immer durch eine äquivalente einzelne Wertdefinition ersetzen. Wie?

Aufgabe 2.6: (*4 P.*)
Das Termersetzungssystem R besteht aus der einzigen Regel $f(f(x)) \to f(g(f(g(x))))$. Ist die Relation $\to_R$ konfluent? Ist sie terminierend? Geben Sie einen Beweis bzw. ein Gegenbeispiel an.

Schicken Sie die ausgetesteten Lösungen der Programmieraufgaben (inklusive einiger Testbeispiele mit Ergebnissen) bis zum 25. 4., 13:00 Uhr per Mail an Ihren Übungsgruppenleiter, Subject: `PS Blatt 2`. Bereiten Sie die übrigen Aufgaben so vor, daß Sie die Lösung in der nächsten darauffolgenden Übungsstunde vortragen können.