

Konzepte höherer Programmiersprachen Sommersemester 2005

Übungsblatt 3

Aufgabe 1

Stellen Sie endliche Mengen als Listen dar

```
type 'a set = 'a list
```

und implementieren Sie die folgenden Funktionen

- a. einen Mengenkonstruktor `set: 'a list -> 'a set`
- b. Funktionen `insert` und `delete` zum Hinzufügen bzw. Entfernen eines einzelnen Elements aus einer Menge
- c. Mengenoperationen `union`, `intersection` und `difference`
- d. Testfunktionen `member`, `is_empty` und `disjoint`
- e. Vergleichsoperatoren `equal` und `subset`
- f. höhere Funktionen `map`, `exists`, `forall` und `filter` analog zu den gleichnamigen Listenfunktionen
- g. eine geeignete Funktion zur “Ausgabe” einer Menge

Überlegen Sie sich selbst, welche Darstellungsinvariante man wählen könnte. Es gibt unterschiedliche Möglichkeiten, die zu mehr oder weniger effizienten Implementierungen führen. Ihre Implementierung sollte nicht zu ineffizient sein.

(Hinweis: Es gibt polymorphe Vergleichsoperationen in O’Caml.)

Aufgabe 2

Bestimmen Sie jeweils *alle* möglichen β -Reduktionen (d.h. Folgen von β -Reduktionsschritten) für die folgenden Ausdrücke

a. $((\lambda x. \lambda y. + x y) 2) ((\lambda x. x) 1)$

b. $(\lambda y. ((\lambda x. \lambda y. + x y) y) 1) 2$

Aufgabe 3

Wiederholen und ergänzen Sie Aufgabe 3 von Blatt 2.

Benutzen Sie dabei die Mengenoperationen aus Aufgabe 1.

Implementieren eine Funktion `beta: expression -> expression`, die einen β -Reduktionsschritt durchführt (falls möglich).

Schreiben Sie ein Programm, das alle möglichen β -Reduktionen für einen Ausdruck e durchführt und in halbwegs lesbarer Form ausgibt.

Überprüfen Sie mit diesem Programm, ob Sie Aufgabe 2 richtig gelöst haben.