Christoph Lüth
Dominik Dietrich
Marcus Ermler
Klaus Hartke
Christian Maeder
Dominik Luecke
Ewaryst Schulz

4. Übungsblatt

Ausgabe: 15.12.08
Abgabe: 12.01.09

9 *Eine ganze Menge* *14 Punkte*

In dieser Aufgabe soll der abstrakte Datentyp (ADT) der *endlichen Mengen* implementiert werden. Eine endliche Menge ist parametrisiert über dem Typ ihrer Elemente, also wird eine Menge von Elementen des Typs `a` durch den abstrakten Typ `Set a` repräsentiert.

Auf diesem abstrakten Typen soll es folgende Operationen geben:

- die leere Menge (`empty`);
- das Hinzufügen eines Elementes zu einer Menge von Elementen desselben Typs (`add`);
- das Löschen eines Elementes aus einer Menge von Element desselben Typs (`delete`);
- der Test, ob ein Element in einer Menge von Elementen desselben Typs vorhanden ist (`isElem`);
- die Vereinigung von zwei Mengen von Elementen desselben Typs (`union`);
- die Teilmengenbeziehung (`subset`) zwischen zwei Mengen von Elementen desselben Typs;
- die Gleichheit zweier Mengen (`==`) zwischen zwei Mengen von Elementen desselben Typs.

Diese Operationen sollen unter anderem folgende Eigenschaften erfüllen:

- Ein Element ist in der Menge, zu der es hinzugefügt wurde, enthalten;
- Ein Element ist in der Menge, zu der ein anderes Element hinzugefügt wurde, genau dann enthalten, wenn es in der Menge ohne das hinzugefügte Element enthalten ist;
- Wenn ein Element zu einer Menge hinzugefügt wird, in der es bereits enthalten ist, bleibt die Menge unverändert;
- Eine Menge ist genau dann eine Teilmenge von einer Menge von Elementen desselben Typs, wenn alle Elemente, die in der erste Menge enthalten sind, auch in der zweiten Menge enthalten sind;
- Ein Element ist in der Vereinigung zweier Mengen enthalten, wenn es in mindestens einer der beiden zu vereinigenden Mengen enthalten ist.

1. Geben Sie eine Signatur für die Operationen an, und formalisieren Sie die oben aufgeführten Eigenschaften als Axiome. Geben Sie mindest noch drei weitere sinnvolle und nicht-triviale Axiome an. Formulieren Sie alle Axiome als Eigenschaften für `quickCheck`. (*7 Punkte*)

2. Implementieren Sie den ADT, indem Sie Mengen als streng aufsteigend geordnete Listen implementieren, und testen Sie Ihre Implementation mit den Eigenschaften in `quickCheck`. Sie können annehmen, dass auf den Elementen der Menge eine Ordnung existiert (d.h. diese sind eine Instanz der Typklasse `Ord`). (*7 Punkte*)

Hinweise zu ***`quickCheck`****:* In `quickCheck` können Sie bedingte Eigenschaften A `==>` B formulieren, wobei A und B vom Typ `Bool` sein müssen. Bei dieser Eigenschaft wird erst A ausgewertet und nur wenn A wahr ist, wird B getestet.

Um Eigenschaften auf Mengen testen zu können, müssen Sie zufällig Mengen erzeugen können. Dafür benutzen Sie folgendes Stück Code, welches zufällige Mengen von bis zu 50 Elementen eines beliebigen Typs erzeugt (die Funktionen `add` und `empty` müssen definiert sein):

```
instance (Ord a, Arbitrary a)=> Arbitrary (Set a) where
  arbitrary = do
    size <- choose (0, 50::Int)
    xs <- vector size
    return (foldr add empty xs)
```

10 *Wahr oder Falsch?* *6 Punkte*

Zeigen Sie folgende Behauptungen, oder geben Sie ein Gegenbeispiel an:

```
length (reverse xs)  ==  length xs                    (1)
filter p (map f xs)  ==  map f (filter p xs)          (2)
 map f (reverse xs)  ==  reverse (map f xs)           (3)
```

Hinweis: Folgende in der Vorlesung gezeigte Lemmas können hilfreich sein:

- `length (xs ++ ys) == length xs + length ys`
- `map f (xs ++ ys) == map f xs ++ map f ys`

Bonausaufgabe (3 Punkte): Zeigen oder widerlegen Sie folgende Behauptung:

```
reverse (xs++ ys) == reverse ys++ reverse xs          (4)
```