Berthold Hoffmann <hof>
Christian Maeder <maeder>
Klaus Lüttich <luettich>
Pascal Schmidt <pharao90>
Felix Beckwermert <foetus>
Dennis Walter <dw>
Franck Ngueuleu <nngf>

4. Übungsblatt

Ausgabe: 01.12.2003

Bearbeitungszeit: bis zum 9.1.2004 um 13 Uhr

9 *Abbildungen als sortierte Paarlisten* *14 Punkte*

Eine *Abbildung* ist eine Funktion von `a` nach `b` oder eine (rechts-)eindeutige Relation. Jedem Element aus `a` ist höchstens ein Element aus dem Wertebereich `b` zugeordnet. (Eine Relation kann man sich als eine Menge von Paaren vorstellen.) Beispiele für endliche Abbildungen sind z.B. die Symboltabellen eines Übersetzers oder ein Lexikon, welches jedem Wort einen erklärenden Eintrag zuordnet.

Geben Sie einen *abstrakten Datentyp* (ADT) als Modul `MapBySortedList` für Abbildungen an, das keine Konstruktoren sondern nur die nachfolgend beschriebenen Funktionen und den polymorphen Datentyp `Map` exportiert.

Der Datentyp '`Map a b`' ist mit den Typen `a` und `b` parametrisiert, wobei lediglich der Definitionsbereich `a` mit einer totalen Ordnung '`Ord a`' ausgestattet sein muss. (Dadurch kann eine Menge von Paaren nicht durch einen abstrakten Datentyp für Mengen repräsentiert werden, der zusätzlich eine totale Ordnung auch für `b` benötigen würde.)

Zu Testzwecken kann eine `Show`-Instanz durch ein nachfolgendes '`deriving Show`' erzeugt werden.

Intern soll eine *Liste* von Paaren verwaltet werden, die bezüglich der ersten Komponente (vom Typ `a`) streng aufsteigend sortiert ist.

1. Eine leere Abbildung: `emptyMap`
2. Test auf leere Abbildung: `isEmptyMap`
3. Definieren einer Zuordnung: `set :: Ord a => (a, b) -> Map a b -> Map a b`
4. Löschen einer Zuordnung: `unset :: Ord a => a -> Map a b -> Map a b`
5. Testen einer Zuordnung: `defined :: Ord a => a -> Map a b -> Bool`
6. Wert einer Zuordnung: `get :: Ord a => a -> Map a b -> b`
7. Konvertierungsfunktionen '`toList`' und '`fromList`' zwischen '`Map a b`' und '`[(a, b)]`'
8. Der Definitionsbereich '`domain`' einer Abbildung (als sortierte Liste) vom Typ '`[a]`'
9. Der Wertebereich '`range`' (oder '`codom`') vom Typ '`[b]`'
10. Die Anzahl der Zuordnungen (einer endlichen Abbildung): `size`
11. Test auf disjunkte Definitionsbereiche zweier Abbildungen: `disjoint`
12. Die Vereinigung von disjunkten Abbildungen: `union`

13. Der Durchschnitt von zwei Abbildungen:
`intersection :: (Ord a) => (b -> b -> b) -> Map a b -> Map a b -> Map a b`
Nur Zuordnungen, die in beiden Eingabeabbildungen definiert sind, liefern eine Zuordnung in der Ergebnisabbildung. Die Werte der Ergebnisabbildung werden mit Hilfe des Funktionsarguments aus den Eingabeabbildungen berechnet.

14. Eine Faltung: `foldMap :: (Ord a) => ((a, b) -> c -> c) -> c -> Map a b -> c`

$\boxed{10}$ *Selbstabbildungen* — *6 Punkte*

Spezielle Abbildungen sind Selbstabbildungen '`Map a a`' mit gleichen Grundmengen/Typen für Definitions- und Wertebereich. In einem weiteren Modul importieren Sie `MapBySortedList` und implementieren folgende Funktionen:

- Komposition von zwei Selbstabbildungen. Dieses entspricht einer Funktionskomposition $g \circ f$, die für eine Eingabe x definiert ist, wenn $f(x)$ und $g(y)$ mit $f(x) = y$ definiert sind, wobei definiert im Sinne von obigem `defined` zu verstehen ist.
- Wandeln Sie eine Selbstabbildung in eine listenwertige Umkehrabbildung vom Typ '`Map a [a]`' um, die jedem Wert (2. Komponenten) aus der Eingabeabbildung alle zugehörigen Urbilder (1. Komponenten) zuordnet.
- Test, ob eine Selbstabbildung injektiv ist (d.h. linkseindeutig bzw. eineindeutig)
- Invertieren einer injektiven Selbstabbildung