Übungsaufgaben zum SML-Kurs, WS 2000/2001

Hans Leiß

Abgabe: Di, 28.11.2000

2 Eine einfache Implementierung von Mengen

Aufgabe 2.1 Deklariere einen Typ für (homogene, endliche) Mengen durch die Typabkürzung

⟨*set*⟩≡
```
type 'a set = 'a list
```

und definiere (ggf. nur einige der) Werte und Funktionen mit folgenden Typen und der unten angegebenen Bedeutung.

⟨*set functions*⟩≡
```
val empty : 'a set
val mk : ''a list -> ''a set
val member : ''a -> ''a set -> bool
val add : ''a -> ''a set -> ''a set
val drop : ''a -> ''a set -> ''a setp
val union : ''a set -> ''a set -> ''a set
val Union : ''a set list -> ''a set
val difference : ''a set -> ''a set -> ''a set
val subset : ''a set -> ''a set -> bool
val equal : ''a set -> ''a set -> bool
```

Die Typvariable `''a` statt `'a` deutet an, daß eine Gleichheit `=` zwischen Elementen der Menge benutzt wird.

`(mk bs)` soll aus einer Eingabe*liste* `bs` eine Menge (in Listendarstellung) machen, also doppelt vorkommende Elemente entfernen. `(member a s)` soll testen, ob `a` in der Menge `s` vorkommt, `(add a s)` soll `a` als Element zu `s` hinzufügen, `(drop a s)` soll `a` aus `s` entfernen (und die Restmenge liefern), `(union s1 s2)` soll die Vereinigung und `(difference s1 s2)` die Differenzmenge der beiden Mengen `s1` und `s2` ergeben, `(Union (s1::s2::s3:: .... sk::nil))` soll die Vereinigung aller Mengen `si` der Liste ergeben, und `(subset s1 s2)` bzw. `(equal s1 s2)` sollen `true` antworten, wenn `s1` eine Teilmenge von `s2` repräsentiert, bzw. wenn `s1` und `s2` dieselbe Menge repräsentieren. (12 Punkte)

Aufgabe 2.2 Wie muß man die oben angegebenen Deklarationen ändern, wenn man einen *abstrakten Datentyp* `'a set` erhalten will, wobei nur die angegebenen Funktionen zum Umgang mit Mengen erlaubt sind? (4 Punkte)

Hinweis: die `abstype`-Deklaration (s.Folien) ist keine Typabkürzung, sondern wie eine *Datentyp*deklaration mit verborgenen Wertkonstruktoren und Funktionsdefinitionen.

⟨*abstype*⟩≡
```
abstype 'a set = ...
   with
       val empty = ....
       ...
   end
```