

Praktikum Spezifikation und Verifikation

1 Bäume

Definieren Sie einen Datentyp `'a tree` für Binärbäume, bei denen nur die Blätter Information enthalten können.

`datatype 'a tree`

Definieren Sie eine Funktion `leaves`, die die Information an den Blättern des Baumes von links nach rechts ausgibt.

`leaves :: "'a tree ⇒ 'a list"`

Definieren Sie eine Funktion:

`plant :: "'a list ⇒ 'a tree"`

so dass die folgende Eigenschaft gilt.

`theorem "leaves (plant xs) = xs"`

Definieren Sie eine Funktion `mirror`, die das Spiegelbild eines Baums berechnet.

`mirror :: "'a tree ⇒ 'a tree"`

Beweisen oder widerlegen Sie (mit Gegenbeispiel) nun folgende Behauptungen.

`theorem "leaves (mirror xt) = rev (leaves xt)"`

`theorem "leaves (mirror (plant xs)) = rev xs"`

`theorem "plant (rev (leaves xt)) = mirror xt"`

Sei `Branch` Ihr Konstruktor für Bäume

`theorem "leaves (Branch (plant xs) (plant ys)) = (xs @ ys)"`

`theorem "plant((leaves xt) @ (leaves yt)) = Branch xt yt"`

Jetzt arbeiten wir mit Binärbäumen, deren Blätter und Knoten Information enthalten. Definieren Sie einen Datentype `'a ntree` für solche Binärbäume.

```
datatype 'a ntree
```

Definieren Sie Funktionen `preOrder`, `postOrder` und `inOrder`, die Bäume in der entsprechende Reihenfolge durchgehen.

```
preOrder :: "'a ntree ⇒ 'a list"  
postOrder :: "'a ntree ⇒ 'a list"  
inOrder :: "'a ntree ⇒ 'a list"
```

Definieren Sie eine Funktion `nmirror`, die das Spiegelbild solcher Bäume berechnet.

```
nmirror :: "'a ntree ⇒ 'a ntree"
```

Sei `xOrder` und `yOrder` jeweils eine von den Funktionen `preOrder`, `postOrder` oder `inOrder`. Formulieren und beweisen Sie alle gültigen Eigenschaften der Form `xOrder (mirror xt) = rev (yOrder xt)`.

Definieren Sie Funktionen `root`, `leftmost` und `rightmost`, die die Wurzel, das Element ganz links, bzw. das Elemente ganz rechts im Baum liefern.

```
root :: "'a ntree ⇒ 'a"  
leftmost :: "'a ntree ⇒ 'a"  
rightmost :: "'a ntree ⇒ 'a"
```

Beweisen oder widerlegen Sie (mit Gegenbeispiel) nun folgende Behauptungen. Eventuell müssen Sie dazu erst einige Lemmas beweisen.

```
theorem "last(inOrder xt) = rightmost xt"  
theorem "hd (inOrder xt) = leftmost xt "  
theorem "hd(preOrder xt) = last(postOrder xt)"  
theorem "hd(preOrder xt) = root xt"  
theorem "hd(inOrder xt) = root xt "  
theorem "last(postOrder xt) = root xt"
```

▷ Abgabe: 30. April 2003