

Aufgabe 10-1

- a) Definieren Sie in PVS eine rekursive Funktion $\text{mod}(n:\text{nat}, k:\text{posnat})$ ¹ zur Berechnung des Rests einer natürlichen Zahl n bei der Division durch k .
- b) Zeigen Sie mittels *vollständiger* Induktion (*Noetherscher Induktion*):

```
n,m : VAR nat
k    : VAR posnat

mod_defn : PROPOSITION
  EXISTS m: n = m*k + mod(n,k)
```

Hinweis: Das Prinzip der Noetherschen Induktion auf natürlichen Zahlen ist in der Prelude-Datei von PVS unter dem Namen `NAT_induction` vordefiniert.

Um in einem Induktionsbeweis dieses Induktionsprinzip anzuwenden, muss dem Befehl

```
(induct/$ var &optional (fnum 1) name)
```

über das optionale Argument `:name` der Name `NAT_induction` übergeben werden. Genaueres siehe `HELP` Kommando.

Aufgabe 10-2

Über einem Grundtyp T definieren wir Sequenzen mit fester Länge N als Funktion folgenden Typs:

```
Sequenz : TYPE = [below(N) -> T]
```

- a) Definieren Sie ein Prädikat `permutation_of?(S1,S2)`, das angibt, ob die Sequenz $S1$ eine Permutation der Sequenz $S2$ ist.
- b) Zeigen Sie, dass `permutation_of?` eine Äquivalenzrelation ist.
- Hinweis:** Eigenschaften bijektiver (injektiver / surjektiver) Funktionen aus der PVS-Prelude-Theorie können verwendet werden.
- c) Definieren Sie ein Prädikat `in?(x:T,S:Sequenz)` das angibt, ob ein Element in einer Sequenz enthalten ist.

¹Beachte den Typ des zweiten Arguments!

Zeigen Sie:

```
perm_in? : LEMMA
  permutation_of?(S1,S2) => (in?(x,S1) <=> in?(x,S2))
```

Die Definition von `in?` kann auf zwei Arten geschehen – welche? Welche erscheint Ihnen für den Beweis günstiger?

Aufgabe 10-3

Gegeben sei der folgende Datentyp eines Binärbaums, in dessen Knoten natürliche Zahlen gespeichert sind:

```
BinTree : DATATYPE
BEGIN
  empty : empty?
  node(key:nat, left:BinTree, right:BinTree) : node?
END BinTree
```

- Definieren Sie Funktionen, die die minimale bzw. maximale Höhe eines Baums berechnen (`minheight`, `maxheight`). Die minimale (maximale) Höhe sei dabei die Länge des kürzesten (längsten) Pfads von der Wurzel bis zu einem Blatt.
Beweisen Sie, dass die minimale Höhe eines Baums immer kleiner oder gleich der maximalen Höhe ist.
- Definieren Sie für einen *nicht-leeren* Binärbaum die Funktionen `minval`, `maxval`, die den kleinsten bzw. größten Wert zurückgeben, sowie die Funktionen `leftmost`, `rightmost`, die den Wert im am weitesten links bzw. rechts liegenden Knoten eines Baums liefern.
- Definieren Sie ein Prädikat `sorted?` über Binärbäumen, das genau dann erfüllt ist, wenn für jeden Knoten eines Baumes gilt, dass die Werte im linken Teilbaum kleiner oder gleich dem Knotenwert und die im rechten Teilbaum größer oder gleich diesem Wert sind.

Zeigen Sie: in einem nicht-leeren, sortierten Baum ist der kleinste Wert immer im am weitesten links und der größte im am weitesten rechts liegenden Knoten.

Vergessen Sie nicht, die entstehenden Typkorrektheitsbedingungen zu zeigen!

Hinweis: Insbesondere bei Teilaufgabe **b)** sind zur Definition der rekursiven Funktionen eine Reihe verschiedener Fälle zu betrachten um sicherzustellen, dass rekursive Aufrufe auch nur mit *nicht-leeren* Bäumen erfolgen. Das `CASES`-Konstrukt eignet sich hierfür vermutlich weniger gut. Als Variante für lange `IF-THEN-ELSE`-Ketten kann das `COND`-Konstrukt eingesetzt werden; näheres dazu siehe in der PVS-Sprachbeschreibung.