

**Hinweis:** Für die Beweise dieser Übungsaufgabe benötigen Sie neue PVS-Beweiserbefehle: (INDUCT), (REPLACE) und (LEMMA). Lesen Sie sich dazu die jeweiligen Abschnitte in der ausgeteilten PVS-Kurzanleitung durch! Hilfreich sind außerdem (REWRITE), (USE).

## Aufgabe 5-1

In der Vorlesung wurde die PVS-Deklaration des abstrakten Datentyps `Nat` vorgestellt:

```
Nat : DATATYPE
BEGIN
  nul      : nul?
  suc(prd:Nat) : suc?
END Nat
```

Auf diesem Datentyp wurde die Addition axiomatisch definiert:<sup>1</sup>

```
; + : [Nat,Nat -> Nat]
x,y : VAR Nat
add0 : AXIOM x + nul = x
add1 : AXIOM x + suc(y) = suc(x + y)
```

a) Beweisen Sie in PVS die Assoziativität der so definierten Addition:

```
z : VAR Nat
add_associative : THEOREM x + (y + z) = (x + y) + z
```

b) Beweisen Sie in PVS die Kommutativität der Addition:

```
add_commutative : THEOREM x + y = y + x
```

**Hinweis:** Sie werden in diesem Beweis vermutlich die Hilfsgleichungen  $nul + y = y$  und  $suc(x) + y = suc(x + y)$  benötigen. Beweisen Sie dafür geeignete Lemmata!

c) Geben Sie eine axiomatische Definition der Multiplikation über `Nat` an.

d) Beweisen Sie in PVS die Distributivität der so definierten Multiplikation bzgl. der Addition:

```
mult_distributes_add : THEOREM x * (y + z) = (x * y) + (x * z)
```

e) Beweisen Sie in PVS die Assoziativität der Multiplikation:

```
mult_associative : THEOREM x * (y * z) = (x * y) * z
```

<sup>1</sup>Das Semikolon am Zeilenanfang der Deklaration von `+` trennt die Deklaration syntaktisch von der vorigen ab. Dies muss an dieser Stelle (leider) gemacht werden, da der PVS-Parser sonst `+` als Infix-Operator interpretiert und so durcheinander gerät.