Kurze Hilfe zu PVS – Zusatz

Funktionen über induktiven Datentypen

Betrachten wir folgenden abstrakten Datentyp zur Modellierung natürlicher Zahlen:

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

Eine Additionsfunktion über diesem Typ kann durch Fallunterscheidung mit Hilfe eines `IF-THEN-ELSE`-Ausdrucks definiert werden:

```
add(n,m:Nat) : RECURSIVE Nat =
  IF m = nul THEN n
    ELSE suc(add(n,prd(m)))
  ENDIF
    MEASURE m BY <<
```

Im `ELSE`-Zweig dieser Definition wird auf das Vorgängerelement von `m` mit einem expliziten Aufruf der Selektor-Funktion `prd` verwiesen.

PVS bietet jedoch auch eine syntaktische Variante zum `IF-THEN-ELSE`-Ausdruck an: den `CASES`-Ausdruck, mit dessen Hilfe ein einfaches *pattern matching* erreicht werden kann. Die obige Additionsfunktion lässt sich mit `CASES` auch wie folgt definieren:

```
add(n,m:Nat) : RECURSIVE Nat =
  CASES m OF
    nul    : n,
    suc(x) : suc(add(n,x))
  ENDCASES
    MEASURE m BY <<
```

Hier werden die verschiedenen Fälle für die einzelnen Konstruktoren des Datentyps nacheinander aufgezählt (durch Kommata getrennt). Es ist darüberhinaus möglich, einen `ELSE`-Zweig zu benutzen, der dann alle noch nicht betrachteten Konstruktoren abdeckt.

Näheres zum `CASES`-Ausdruck findet sich in der PVS-Sprachbeschreibung.