

Übungen zur Vorlesung „Grundlagen der Programm- und Systementwicklung“

Aufgabe 1 Vollständige Induktion

Betrachten Sie die Funktion

$$sum : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

gegeben durch

$$sum(x, y) = \begin{cases} x & \text{falls } y = 0 \\ sum(x+1, y-1) & \text{falls } y > 0 \end{cases}$$

(a) Beweisen Sie zunächst

$$sum(x, y+1) = 1 + sum(x, y)$$

für alle $x, y \in \mathbb{N}$ durch vollständige Induktion.

(b) Beweisen Sie nun

$$sum(x, y) = sum(y, x)$$

für alle $x, y \in \mathbb{N}$ ebenfalls durch vollständige Induktion.

Aufgabe 2 Noethersche Induktion

Wir nennen eine Ordnung \prec auf einer Menge M *noethersch*, wenn bezüglich \prec in M keine unendlichen Folgen $(x_j) \in (\mathbb{N} \rightarrow M)$ mit

$$x_{i+1} \prec x_i$$

existieren (für $i \in \mathbb{N}$).

Das Beweisprinzip der Noetherschen Induktion besagt für eine Eigenschaft $E : M \rightarrow \mathbb{B}$ und eine noethersche Ordnung \prec auf einer Menge M folgendes ($x, y, z \in M$):

Um $E(x)$ für alle x nachzuweisen, genügt es $E(z)$ unter der Voraussetzung nachzuweisen, dass $E(y)$ für alle Vorgänger y von z gilt.

In einer Formel lässt sich dieses Prinzip folgendermaßen fassen:

$$\left[\forall z \in M : (\forall y \in M : y \prec z \Rightarrow E(y)) \Rightarrow E(z) \right] \Rightarrow (\forall x \in M : E(x))$$

- (a) Beweisen Sie die Gültigkeit dieses Prinzips.
- (b) Leiten Sie die vollständige und die Terminduktion/strukturelle Induktion als Spezialfälle der Noetherschen Induktion her.
- (c) Beweisen Sie die Aussage aus Aufgabe 1(b) nun mittels noetherscher Induktion. Geben Sie dazu eine geeignete noethersche Ordnung auf Zahlenpaaren an.

(d) Die Ackermannfunktion $ack : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ ist folgendermaßen definiert:

$$ack(x, y) = \begin{cases} y+1 & \text{falls } x = 0 \\ ack(x-1, 1) & \text{falls } x \neq 0 \text{ und } y = 0 \\ ack(x-1, ack(x, y-1)) & \text{sonst} \end{cases}$$

Zeigen Sie mittels Noetherscher Induktion, dass die Ackermannfunktion terminiert.

Aufgabe 3 Vom Programm zur algebraischen Spezifikation, Termininduktion

Betrachten Sie folgenden Ausschnitt aus einem Pascal-Programm zur Verarbeitung mathematischer Ausdrücke über ganzen Zahlen. Ein Ausdruck ist dabei entweder eine ganze Zahl, oder zusammengesetzt aus einem linken Operanden, einem binären Operator („+“, „*“, „-“, oder „/“), und einem rechten Operanden. Beide Operanden sind wiederum Ausdrücke.

```

type Op      = (plus, mal, minus, durch);
type ExpType = (iExp, cExp);
type pExp    = ↑Exp; (* pointer to Exp *)
type Exp     = record
                case t : ExpType of
                    iExp : (iVal : integer);
                    cExp : (lExp : pExp;
                           op   : Op ;
                           rExp : pExp)
                end (*case *);
            end (* record *);

function numCnt(e : Exp) : integer;
begin
    case e.t of
        iExp : numCnt := 1;
        cExp : numCnt := (numCnt(e.lExp)+numCnt(e.rExp));
    end (*case *);
end (* numCnt *);

function opCnt(e : Exp) : integer;
begin
    case e.t of
        iExp : opCnt := 0;
        cExp : opCnt := (1+opCnt(e.lExp)+opCnt(e.rExp));
    end (*case *);
end (* numCnt *);

```

- (a) Übertragen Sie diesen Programmausschnitt in eine entsprechende algebraische Spezifikation.
 (b) Beweisen Sie mittels Termininduktion, dass

$$numCnt(x) = opCnt(x) + 1$$

für diese Spezifikation gilt.