Technische Universität München
Fakultät für Informatik
Prof. Dr. M. Broy

WS 2001
Übungsblatt 5
15. November 2001

Übungen zu Einführung in die Informatik I

Aufgabe 17 Die Signatur der Rechenstruktur der ganzen Zahlen

Zeigen Sie durch Angabe des Signaturgraphen: Die Rechenstruktur der ganzen Zahlen **INT** hat bis auf Umbenennung von Sorten die gleiche Signatur wie die aus der Vorlesung bekannte Rechenstruktur der natürlichen Zahlen **NAT**. Zeigen Sie durch Angabe eines geeigneten Teils der Spezifikationen, dass die Rechenstrukturen verschieden sind.

Aufgabe 18 Kartesisches Koordinatensystem

Wir erweitern die aus der vorhergehenden Aufgabe bekannte Rechenstruktur **INT** folgendermaßen zu der Rechenstruktur **K** der Punkte in einem kartesischen Koordinatensystem mit ganzzahligen Koordinaten: Die Menge S der Sorten der Rechenstruktur **K** sei gegeben durch

$$S_{\mathbf{K}} = \{\mathbf{bool}, \mathbf{int}, \mathbf{point}\}.$$

Zusätzlich zu den Funktionssymbolen der Rechenstruktur **INT** besitze die Rechenstruktur **K** die Symbole

$$F_{\mathbf{K}} = F_{\mathbf{INT}} \cup \{\text{make}, \text{projX}, \text{projY}, \text{moveto}\}$$

mit den Funktionalitäten

fct make = (**int, int**) **point**
fct projX = (**point**) **int**
fct projY = (**point**) **int**
fct moveto = (**point, int, int**) **point**

a) Geben Sie den Signaturgraphen der Rechenstruktur **K** an !

b) Welche der folgenden Terme sind syntaktisch korrekt ?

(i) make(succ(zero), projX(make(zero, succ(succ(zero)))))

(ii) moveto(make(succ(succ(zero))), add(succ(zero), succ(zero)), div(succ(zero), zero))

(iii) make(pred(projY(make(succ(zero), zero))), mult(add(succ(zero), succ(zero))))

Aufgabe 19 Termersetzungsregeln und partielle Korrektheit

In der Vorlesung wurde ein Termersetzungssystem zur Rechenstruktur **NAT** bestehend aus Reduktionsregeln für die Funktionen pred, add und mult eingeführt. Wir erweitern das Termersetungssystem um die Regeln für die strikte Funktion sub mit der Spezifikation:

$$\text{sub}^{\mathbf{NAT}} : \mathbb{N}^{\perp} \times \mathbb{N}^{\perp} \to \mathbb{N}^{\perp}; \quad \text{für } x,y \in \mathbb{N} \text{ gilt,} \quad \begin{array}{lcll} \text{sub}^{\mathbf{NAT}}(x,y) & = & x-y & \text{, falls } x \geq y \\ \text{sub}^{\mathbf{NAT}}(x,y) & = & \perp & \text{, falls } x < y \end{array}$$

a) Geben Sie die Funktionalität von sub und Reduktionsregeln für sub an.

b) vereinfachen Sie folgende Terme mit den eingeführten Regeln bis zu ihrer terminalen Grundform:

- pred(sub(succ(succ(succ(zero))), succ(succ(zero))))
- sub(succ(zero), succ(succ(zero)))

c) Zeigen Sie die partielle Korrektheit der Termersetzungsregeln von sub bezüglich der Rechenstruktur **NAT**.

Aufgabe 20 Termersetzung: Sequenzen

Gegeben ist die Rechenstruktur **SEQ** von Arbeitsblatt 2. Wir betrachten in dieser Aufgabe Sequenzen über der Sorte **bool**, d.h. Sequenzen der Sorte **seq bool**.

Wir nennen zwei Sequenzen aus **seq bool** zueinander komplementär, wenn sie gleich lang sind und die Wahrheitswerte an jeder Position Negierte voneinander sind. Zum Beispiel sind die Sequenzen `conc(make(true ),conc(make(false),conc(make(true ),empty)))` und `conc(make(false),conc(make(true ),conc(make(false),empty)))` zueinander komplementär.
Die Funktion mit der Funktionalität

fct `kompl = ( seq bool, seq bool )` **bool**

gibt an, ob zwei Argumente zueinander komplementär sind.

Aufgabenstellung: Geben Sie ein Termersetzungssystem an, das für alle Grundterme *s*, *t* über **SEQ** den Wert von `kompl(s,t)` berechnet. Der Einfachheit halber können Sie annehmen, dass die Grundterme nur aus `empty` , `make` und `conc` aufgebaut sind. Diese Terme nennen wir "Konstruktorterme".

Aufgabe 21 (P) (H) Markov-Algorithmus — Boolesche Terme

In dieser Aufgabe ist ein Markov-Algorithmus zu entwickeln, der boolesche Terme akzeptiert. Der Algorithmus soll mit ε terminieren, wenn es sich bei der Eingabe um einen vollständig geklammerten, booleschen Term handelt, andernfalls soll der Algorithmus mit einer Ausgabe $\neq \varepsilon$ terminieren. Zur Vereinfachung sei die Menge der Identifikatoren *ID* auf $\{x,y,z\}$ eingeschränkt und als boolesche Operationssymbole seien lediglich $\wedge$, $\vee$ und $\neg$ zugelassen.

a) Geben Sie einen geeigneten Zeichenvorrat V für Ihren Markov-Algorithmus an.

b) Geben Sie die Regelmenge Ihres Markov-Algorithmus an.

c) Geben Sie die Berechnungen Ihres Algorithmus für folgende Eingaben an:

(i) $(((x \wedge y) \vee (\neg y)) \wedge true)$

(ii) $(z \vee (yz)$

d) Implementieren Sie Ihren Algorithmus in Gofer und testen Sie diesen mit den beiden Eingaben aus Teilaufgabe c). Gehen Sie dabei, wie in Aufgabe 16 beschrieben, vor.
Hinweis: Symbolisieren Sie bei der Definition der Regeln in Gofer das Zeichen $\neg$ durch *not*, $\wedge$ durch && und $\vee$ durch ||.