Übung zur Vorlesung

Automatisierte Logik und Programmierung

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Wintersemester 2008/09

Blatt 6 — Abgabetermin: 04.02.09 nach der Übung

Achtung: Am 27. und 28. Januar 2009 finden keine Vorlesungen statt.
Die nächste Veranstaltung ist die Vorlesung am 3. Februar.
Als Termin für die Prüfungsklausur ist Dienstag, der 17.3., 11:00-12:30 vereinbart worden.

Aufgabe 6.1 (Für Tüftler: Lehre der Leere)

Zeigen Sie, das sich der Datentyp Void durch den Ausdruck $\texttt{T} = \texttt{F} \in \mathbb{B}$ simulieren läßt. Dabei sei $\mathbb{B}$ inklusive seiner kanonischen und nicht–kanonischen Elemente wie in Übung 5 definiert.

Die Korrektheit der Simulation von Void *durch* **T=F** $\in \mathbb{B}$ *läßt sich am besten dadurch zeigen, daß man in den Inferenzregeln für* Void *letzteres durch die angegebene Simulation ersetzt und dann unter Zuhilfenahme der Regeln für* $\mathbb{B}$ *zeigt, daß jeweils die selben Unterziele entstehen wie in den ursprünglichen Regeln. Betrachten Sie dabei der Einfachheit halber nur die Regeln* `voidEq` *und* `voidE`*. Wie könnte man* any`(x)` *simulieren?*

Aufgabe 6.2 (Programmsynthese: Minimum–Problem)

Gegeben sei die Spezifikation: $\forall \texttt{i,j}:\mathbb{Z}.\exists \texttt{min}:\mathbb{Z}.\ \texttt{min} \le \texttt{i} \land \texttt{min} \le \texttt{j} \land (\texttt{min=i} \in \mathbb{Z} \lor \texttt{min=j} \in \mathbb{Z})$.

Beweisen Sie dieses Spezifikationstheorem und extrahieren Sie das zugehörige Programm.

Nach Einführung der Allquantoren kann mit der Schnittregel $\texttt{cut 2 i<j} \lor \texttt{i} \ge \texttt{j}$ *eine Fallunterscheidung eingefügt werden. Beweist man das Ziel* $\texttt{i<j} \lor \texttt{i} \ge \texttt{j}$ *mit Hilfe der* `arith`*–Regel, so entsteht der Extrakt–Term* if `i<j` then inl`(`Ax`)` else inr`(`λ`z.`Ax`)`*. Alle weiteren Unterziele, in denen nur noch über die Ordnung zwischen* `i` *und* `j` *entschieden werden muß, können ebenfalls mit* `arith` *bewiesen werden, wobei zur Vereinfachung angenommen werden kann, daß als Extrakt–Term* Ax *entsteht.*

Aufgabe 6.3 (Formalisierung)

Formalisieren Sie die folgenden Konzepte als induktive Datentypen

6.3–a Nichtleere Listen über natürlichen Zahlen

6.3–b Endliche (nicht notwendigerweise binäre) Bäume über natürlichen Zahlen

6.3–c λ-Terme als syntaktisches Konzept

6.3–d Prädikatenlogische Formeln als syntaktisches Konzept

Geben Sie hierfür eine informale Beschreibung der wesentlichen Komponenten dieser Konzepte (ohne den "syntaktischen Zucker") und setzen Sie diese in eine rekursive Typgleichung um.

Aufgabe 6.4 (Entwicklung rekursiver Programme)

Konstruieren Sie verschiedenartige Programme, die den größten gemeinsamen Teiler zweier Zahlen berechnen.

6.4–a Formalisieren Sie ein Prädikat `GGT(i,j,k)`, welches ausdrücken soll, daß `k` der größte gemeinsame Teiler der natürlichen Zahlen `i` und `j` ist.

6.4–b Beschreiben Sie einen *induktiven* Algorithmus der Gestalt

$$\texttt{ggt} \equiv \lambda\texttt{i}.\lambda\texttt{j}.\mathsf{ind}(\texttt{i};\_,\_\_;\ base;\ y,f_y.t),$$

welcher das Problem löst.

6.4–c Beschreiben Sie einen *rekursiven* Algorithmus der Gestalt

$$\texttt{ggT} \equiv \mathsf{letrec}\ f\texttt{(pair)} = (\mathsf{let}\ \langle\texttt{i,j}\rangle = \texttt{pair}\ \mathsf{in}\ t),$$

welcher das Problem löst.

6.4–d Berechnen Sie – durch Reduktion und in Einzelschritten – mit beiden Algorithmen den größten gemeinsamen Teiler von 4 und 6.

Aufgabe 6.5 (Rückblick)

Die folgenden Kontrollfragen dienen der Überprüfung des eigenen Kenntnisstandes. Sie entsprechen in ihrer Thematik dem Spektrum einer mündlichen Prüfung. Die Antworten sind größtenteils im Skript, allerdings selten an auffälliger Stelle. Versuchen sie, diese zunächst ohne Ihre Unterlagen zu beantworten.

6.5–a Nennen Sie die drei Grundbestandteile eines formalen Kalküls

6.5–b Welche Vor- und Nachteile bringt es, eine `cut`-Regel in einen Kalkül hineinzunehmen?

6.5–c Wodurch wird in der Prädikatenlogik den Zusammenhang zwischen einer Formel und ihrer Bedeutung hergestellt?

6.5–d Welches fundamentale Gesetz der klassischen Logik ist intuitionistisch nicht allgemeingültig? Warum?

6.5–e Warum wird bei den Quantorenregeln der Prädikatenlogik eine "Eigenvariablenbedingung" benötigt?

6.5–f Erklären Sie den Unterschied zwischen Korrektheit und Vollständigkeit eines Kalküls für die Prädikatenlogik.

6.5–g Wodurch wird die Semantik von λ-Termen definiert?

6.5–h Mit welchem Hilfsmittel kann man im λ-Kalkül Rekursion einführen?

6.5–i Warum ist der λ-Kalkül nicht stark normalisierbar?

6.5–j Welche berechenbaren Funktionen lassen sich *nicht* durch λ-Terme beschreiben? Warum?

6.5–k Auf welche zwei Arten kann man eine Typdisziplin für den λ-Kalkül einführen?

6.5–l Wie kann man die Gleichheit typisierbarer λ-Terme entscheiden? Warum?

6.5–m Welches typentheoretische Gegenstück gibt es zum *variant record* von PASCAL?

6.5–n Welches semantische Konzept der Typentheorie wird durch Sequenzen repräsentiert?

6.5–o Welche Erkenntnis erlaubt es, auf die explizite Einführung logischer Konnektive in der Typentheorie zu verzichten?

6.5–p Durch welches Konstrukt wird die Typeigenschaft syntaktisch wiedergespiegelt?

6.5–q Welche Programmstruktur entspricht gemäß dem "*Beweise als Programme*"-Prinzip der induktiven Beweisführung?

6.5–r Wozu lassen sich Quotiententypen verwenden? Nennen Sie ein konkretes Beispiel.

6.5–s Welche Schwierigkeiten ergeben sich durch die Hinzunahme eines Gleichheitstyps oder eines leeren Datentyps zur Typentheorie und wie werden sie überwunden?

6.5–t Welches typentheoretische Konzept ist das Analogon zur Schnittelimination in der Logik?

6.5–u Welche Schwierigkeit ist mit der Einführung von Teilmengenoperatoren verbunden?

6.5–v Welche Komplexität können Programme, die aus induktiven Beweisen extrahiert werden, bestenfalls erreichen?

6.5–w Warum kann in der intuitionistischen Typentheorie mit abhängigen Datentypen die Typeigenschaft nicht einfach durch ein Symbol $\mathbb{U}$ repräsentiert werden?

6.5–x Welche Besonderheit zeichnet den Term any`(z)` aus?

6.5–y Nennen Sie drei Formen rekursiver Definitionen, die sich in der Typentheorie formalisieren lassen.

6.5–z Wie ist die Semantik der Gleichung $T = F[T]$ bei induktiven Datentypen definiert?

6.5– Warum dürfen auf der rechten Seite von rekursiven Typgleichungen nicht beliebige Funktionenraumkonstrukte auftauchen

6.5– Durch welche syntaktische Einschränkung kann man dem obigen Problem begegnen?