Übung zur Vorlesung

Automatisierte Logik und Programmierung I

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Wintersemester 2013/14

Blatt 6 — Abgabetermin: 7.2.2014

Das sechste Übungsblatt soll dazu dienen, sich mit Beweisen, Programmextraktion und rekursiven Typen auseinanderzusetzen. Wir werden mögliche Lösungen in der Veranstaltung am 7.2.2014 besprechen.

Achtung: Am 30. und 31. Januar 2014 finden keine Vorlesungen statt.
Die nächste Veranstaltung ist die Vorlesung am 6. Februar.
Als Termin für die Prüfungsklausur ist Freitag, der 14.2., 12:30-14:30, im Hörsaal 3+4 vereinbart worden.

Aufgabe 6.1 (Programmsynthese: Minimum–Problem)

Gegeben sei die Spezifikation: $\forall \mathtt{i,j}{:}\mathbb{Z}.\exists \mathtt{min}{:}\mathbb{Z}.\ \mathtt{min}{\le}\mathtt{i} \wedge \mathtt{min}{\le}\mathtt{j} \wedge (\mathtt{min}{=}\mathtt{i} \in \mathbb{Z} \vee \mathtt{min}{=}\mathtt{j} \in \mathbb{Z})$.

Beweisen Sie dieses Spezifikationstheorem und extrahieren Sie das zugehörige Programm.

Hinweis: *Nach Behandlung der Allquantoren kann mit der Schnittregel* `cut` *eine Fallunterscheidung* $\mathtt{i}{<}\mathtt{j} \vee \mathtt{i}{\ge}\mathtt{j}$ *eingefügt werden. Beweist man diese mit Hilfe der* `arith`*–Regel, so entsteht der Extrakt–Term* $\mathtt{if}\ \mathtt{i}{<}\mathtt{j}\ \mathtt{then}\ \mathsf{inl}(\mathsf{Ax})\ \mathtt{else}\ \mathsf{inr}(\lambda \mathtt{z}.\mathsf{Ax})$. *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 6.2 (Synthese von $\lfloor \log_k n \rfloor$)

Synthetisieren Sie ein Programm zur Berechnung von ganzzahligen Logarithmen.

Geben Sie dazu eine formale (rekursive) Definition der Potenz n^k an, formulieren Sie das erforderliche Spezifikationstheorem, skizzieren Sie einen Beweis, extrahieren Sie daraus den Algorithmus für $\lfloor \log_k n \rfloor$ und schätzen Sie seine Komplexität ab. Falls erforderlich, formulieren Sie ein Induktionslemma.

Aufgabe 6.3 (Entwicklung rekursiver Programme)

Konstruieren Sie verschiedene Programme zur Berechnung des größten gemeinsamen Teilers.

6.3–a Formalisieren Sie ein Prädikat $\mathtt{GGT(i,j,k)}$, welches ausdrückt, daß `k` der größte gemeinsame Teiler der natürlichen Zahlen `i` und `j` ist.

6.3–b Beschreiben Sie einen *induktiven* ggt-Algorithmus der Form
$$\mathtt{ggt} \ \equiv\ \lambda \mathtt{i}.\lambda \mathtt{j}.\mathsf{ind}[\mathit{base};\ \_[\_,\_];\ g[x,y]\,](i).$$

6.3–c Beschreiben Sie einen *rekursiven* ggt-Algorithmus der Form
$$\mathtt{ggT} \ \equiv\ \mathsf{function}\ f(p) = \mathsf{match}\ p\ \mathsf{with}\ \langle \mathtt{i},\mathtt{j}\rangle \mapsto t.$$

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

Aufgabe 6.4 (Formalisierung induktiver Datentypen)

Formalisieren Sie die folgenden Konzepte als induktive Datentypen

6.4–a Nichtleere Listen über natürlichen Zahlen

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

6.4–c λ-Terme als syntaktisches Konzept

6.4–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 6.5 (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 Grundkomponenten der Beschreibung eines formalen Kalküls

6.5–b Erklären Sie skizzenhaft die prädikatenlogische Refinement-Logik

6.5–c Wie kann in der Prädikatenlogik die Semantik einer Formel beschrieben werden?

6.5–d Welche formale Struktur hat Evidenz für die Gültigkeit von Formeln der Art $A \wedge B$, $A \vee B$ und $A \Rightarrow B$?

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

6.5–f Erklären Sie die Inferenzregel `andR`: Auf welche Sequenzen kann sie angewandt werden, welche Teilziele erzeugt sie und welche Evidenz wird durch sie konstruiert?

6.5–g Erklären Sie die Inferenzregel `allR`: Auf welche Sequenzen kann sie angewandt werden, welche Teilziele erzeugt sie und worauf muß man achten?

6.5–h Geben Sie einen Beweis für die Formel $P \Rightarrow (Q \Rightarrow P)$ in Refinement-Logik. Extrahieren Sie aus dem Beweis eine Evidenz für $P \Rightarrow (Q \Rightarrow P)$.

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

6.5–j Welche Vor- und Nachteile bringt es, eine Schnittregel (`cut`) in einen Kalkül hineinzunehmen?

6.5–k Beschreiben Sie den Aufbau des λ-Kalküls

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

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

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

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

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

6.5–q Beschreiben Sie den Aufbau der einfachen Typentheorie. Geben Sie dabei eine genaue Definition der Typzugehörigkeitsrelation an.

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

6.5–s Warum ist die einfache Typentheorie zur Formalisierung berechenbarer Funktionen unzureichend?

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

6.5–u Erklären Sie die Kernaussage von Girard's Paradox

6.5–v Erklären Sie die wichtigsten Grundkonzepte des semantischen Aufbaus der konstruktiven Typentheorie. Beschreiben Sie diese anhand des abhängigen Funktionenraums

6.5–w Warum kann man auf die explizite Einführung logischer Konnektive in der Typentheorie verzichten?

6.5–x Erklären Sie die wichtigsten Grundkonzepte des Refinement-Kalküls konstruktiven Typentheorie.

6.5–y Welches semantische Konzept der Typentheorie wird im Refinement-Kalkül durch Sequenzen repräsentiert?

6.5–z Durch welches Konstrukt wird die Typeigenschaft im Refinement-Kalkül syntaktisch wiedergespiegelt?

6.5– Welche Arten von Refinement-Regeln sollten (wenn möglich) für jeden Datentyp angegeben werden ? Beschreiben Sie diese skizzenhaft anhand des abhängigen Funktionenraums

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

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

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

6.5– Wie kann man den leeren Datentyp durch andere Typen simulieren ?

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

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

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

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

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

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