Übung zur Vorlesung

Automatisierte Logik und Programmierung I

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Wintersemester 2016/17

Blatt 7 — Abgabetermin: 09.02.2017

Das siebte Übungsblatt soll dazu dienen, sich mit rekursiven Programmen und induktiven Datentypen auseinanderzusetzen und wichtige Fragestellungen der bisherigen Veranstaltung zu wiederholen. Wir werden mögliche Lösungen in der Veranstaltung am 09.02.2017 besprechen.

Aufgabe 7.1 (Entwicklung rekursiver Programme)

Konstruieren Sie verschiedene Programme zur Berechnung des größten gemeinsamen Teilers.

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

7.1–b Beschreiben Sie einen *induktiven* ggt-Algorithmus der Form

$$\texttt{ggt} \;\equiv\; \lambda\texttt{i}.\lambda\texttt{j}.h(i) \text{ where } \begin{array}{ll} h(0) & = base \\ \mid\; h(z{<}0) & = \_ \\ \mid\; h(z{>}0) & = g[z, h(z{-}1)/x, y]. \end{array}$$

Geben Sie dazu zwei passende Terme $base$ und g (mit freien Variablen x und y) an.

7.1–c Beschreiben Sie einen *rekursiven* ggt-Algorithmus der Form

$$\texttt{ggT} \;\equiv\; \text{function } f(p) = \text{match } p \text{ with } \langle i, j\rangle \mapsto t.$$

Geben Sie dazu einen passenden Term t mit freien Variablen i und j an.

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

Aufgabe 7.2 (Formalisierung induktiver Datentypen)

Formalisieren Sie die folgenden Konzepte als induktive Datentypen

7.2–a Nichtleere Listen über natürlichen Zahlen

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

7.2–c λ-Terme als syntaktisches Konzept

7.2–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 7.3 (Rückblick I)

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

7.3–a Nennen Sie die drei Grundkomponenten der Beschreibung eines formalen Kalküls.

7.3–b Erklären Sie skizzenhaft die prädikatenlogische Refinement-Logik.

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

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

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

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

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

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

7.3–i Was ist bei einem logischen Kalkül der Unterschied zwischen Korrektheit und Vollständigkeit?

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

7.3–k Welche Aufgaben haben Bibliothek, Benutzerinterface und Inferenzmaschine eines Beweisassistenten?

7.3–l Welche Gestaltungsmöglichkeiten gibt es für die Bibliothek, das Benutzerinterface und die Inferenzmaschine eines Beweisassistenten? Beschreiben Sie Vor-und Nachteile der jeweiligen Gestaltungsform.

7.3–m Beschreiben Sie den Aufbau des λ-Kalküls.

7.3–n Wodurch wird die Semantik von λ-Termen definiert?

7.3–o Mit welchem Hilfsmittel kann man im λ-Kalkül Rekursion einführen?

7.3–p Warum ist der λ-Kalkül nicht stark normalisierbar?

7.3–q Wie kann man im λ-Kalkül boolesche Operatoren, Datenstrukuren wie Paare oder Listen, und Zahlen beschreiben?

7.3–r Welche berechenbaren Funktionen lassen sich *nicht* durch λ-Terme beschreiben? Warum?

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

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

7.3–u Geben Sie je ein Beispiel für einen typisierbaren und einen nicht typisierbaren λ-Term.

7.3–v Mit welcher Strategie kann man die schwache Normalisierbarkeit typisierbarer λ-Terme zeigen?

7.3–w Mit welcher Methode beweist man die starke Normalisierbarkeit typisierbarer λ-Terme?

7.3–x Wie beweist man die Konfluenz typisierbarer λ-Terme?

7.3–y Wie kann man die Gleichheit typisierbarer λ-Terme entscheiden? Warum?

7.3–z Warum ist die einfache Typentheorie zur Formalisierung berechenbarer Funktionen unzureichend?

Aufgabe 7.4 (Rückblick II)

7.4–a Beschreiben Sie den Aufbau der Theorie abhängiger Datentypen. Erklären Se die Unterschieede zur einfachen Typentheorie.

7.4–b Wie kann man in der Theorie abhängiger Datentypen die Typen $\mathbb{B}$, Produkte, Summentypen, $\mathbb{N}$, logische Operationen und Gleichheit beschreiben?

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

7.4–d Erklären Sie die Kernaussage von Girard's Paradox.

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

7.4–f Warum kann man auf die explizite Einführung logischer Konnektive in der Typentheorie verzichten?

7.4–g Erklären Sie die wichtigsten Grundkonzepte des Refinement-Kalküls konstruktiven Typentheorie.

7.4–h Welches semantische Schlüsselkonzept der Typentheorie wird im Refinement-Kalkül durch Sequenzen repräsentiert?

7.4–i Durch welches Konstrukt wird die Typeigenschaft im Refinement-Kalkül syntaktisch wiedergespiegelt?

7.4–j Warum ist der Refinement-Kalkül für die Typentheorie extensional?

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

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

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

7.4–n Welches formal beweisbare Theorem ist die Grundlage für das Prinzip "*Beweise als Programme"*?

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

7.4–p Welche Programmstruktur entspricht gemäß des Prinzips "*Beweise als Programme"* der induktiven Beweisführung?

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

7.4–r Welche Schwierigkeit ist mit der Einführung von Teilmengenoperatoren verbunden?

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

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

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

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

7.4–w Warum dürfen auf der rechten Seite von rekursiven Typgleichungen nicht beliebige Funktionenraumkonstrukte auftauchen? Durch welche syntaktische Einschränkung kann man dem obigen Problem begegnen?