FM Abschlussprüfung Elias

Gerhard hat bei mir die (meisten) Fragen gestellt.

Frei Datentypen

- Wie bei allen: Wie spezifieren wir denn standardmäßig Datentypen? → Sorten, Konstruktoren, frei oder nicht frei, je nach dem Data Specification oder Extensionalitätsaxiom
- Dann zieht Gerhard eine wunderschöne Zeichnung von einem Hypergraphen hervor (Kanten verbinden mehrere Knoten gleichzeitig und evtl. auch mehrfach), den sollte ich spezifizieren. Wollte schon anfangen zu schreiben, aber sie wollten nur eine mündliche Beschreibung.
  - Sorten: node, edge, graph
  - Edge ist einfach eine nicht-leere Multimenge und Graph ansonsten ganz normal
  - Konstruktoren:
    - Node: Keine, wir machen node zu einem Parameter
    - Edge: mkedge(node), edge.addnode(node)
    - Graph: emptygraph, graph.addedge(edge), graph.addnode(node)
  - Weder edge noch graph frei (Einfügereihenfolge egal)
  - Extensionalitätsaxiom für edge: $e1 = e2 \leftrightarrow \forall n.\, e1.count(n) = e2.count(n)$
  - Dann noch die verwendeten Funktionen axiomatisieren, also:
    - $mkedge(n).\, count(n) = 1$
    - $n1 \neq n2 \rightarrow mkedge(n1).\, count(n2) = 0$
    - $e.\, addnode(n).\, count(n) = e.\, count(n) + 1$
    - $n1 \neq n2 \rightarrow e.\, addnode(n1).\, count(n2) = e.\, count(n2)$
  - Sind wir jetzt fertig? – Nein, wir müssen noch das Extensionalitätsaxiom für den Graph aufschreiben. – Ja schon klar, aber dann? – Nein, dann müssen wir noch die Funktionen axiomatisieren, die das Graph-Extensionalitätsaxiom verwendet. – Ja ja, klar, aber sind wir dann fertig? – Joa kommt drauf an, wenn wir noch zusätzliche Funktionen haben müssen wir die – Ja ja, aber wenn wir keine haben? – Hm, wir haben noch Beweisverpflichtungen für das Extensionalitätsaxiom, das muss eine Kongruenzrelation beschreiben – Ja ja ja, aber sind wir mit der Spezifikation fertig? – Ja dann schon. – Ja, endlich. 😆
- Warum muss das Extensionalitätsaxiom denn eine Kongruenzrelation beschreiben? – Weil die Gleichheit sonst keine Gleichheit ist, also $x \neq x$ wäre z.B. blöd. – Ja, aber was geht uns denn dann verloren? Was wir immer wollen? – Ah, Konsistenz.
- Nach welchem Prinzip hast du denn deine Funktionen spezifiziert? – Strukturell rekursiv. – Okay, was gäbs noch? – Wohlfundiert rekursiv, nicht-rekursiv, Existenz und Eindeutigkeit
- Welches Lemma steckt hinter Ex. und Eind.? – Das von Skolem. – Und was ist da die Idee? – Wenn man weiß, dass für alle x ein y existiert, kann man sich das y auch durch eine Funktion geben lassen

DL

- Wofür kann man Wohlfundiertheit noch brauchen, und was heißt das? – Relation ohne unendlich absteigende Ketten; für Induktion, While-Schleifen und Rekursion
- Schreib mal die Induktionsregel für Listen hin
- Dann noch eine Diamond-While-Regel der DL
- Definier mal Box oder Diamond – Ich hab zum Spaß hier gleich die natürliche Semantik verwendet mit der Notation von der Folie zur Äquivalenz (also sowas wie $(v, w) \in [p]_{ns}$). Gerhard kannte die Syntax gar nicht, aber Alex meinte doch, haben wir zwar so gemacht, aber da hätte er auf den Folien geschlampt und man müsste eigentlich schon die Algebra irgendwie noch mitreinbringen
- Du hast die natürliche Semantik jetzt schon ins Spiel gebracht, wofür hatten wir denn dann die relationale überhaupt? – Man spart sich die ganze Theorie drum rum mit Fixpunkten etc.
- Wie ist die relationale Semantik denn definiert? – Rekursiv
- Und die natürliche? – Über Regeln – Wie kommt man von den Regeln auf die Semantik? – Das ist der kleinste Fixpunkt des Regeloperators. – Und was beschreibt der kleinste Fixpunkt? – alles was man als Konklusion von abgeschlossenen Beweisbäumen bekommt
- Was macht der Regeloperator? – Bringt uns alle Konklusionen, die wir aus einer gegebenen Prämissenmenge durch einmalige Regelanwendung erhalten können
- Welche Sätze hatten wir für den kleinsten Fixpunkt? – Knaster-Tarski, Kleene, Stetigkeit, Monotonie, usw.
- Wie zeigt sich Stetigkeit/Monotonie beim Regelsystem? – (Endlich viele) positive Prämissen

Indeterminismus

- Was haben wir für Indeterminismus? – Choose und als Abkürzung or
- Sind wir trotzdem noch stetig? – Ja, aber beim garantierten Terminierungsbereich macht Choose Probleme wegen evtl. unendlich vieler Prämissen, or nicht.
- Wo hatten wir noch unendlich viele Prämissen? – Omega-Kalkül. – Wofür haben wir das da gebraucht? – Da hab ich mich dann nicht mehr so recht dran erinnert. Alex hat mir dann erklärt dass wir da keine Induktion hatten, sondern bei natürlichen Zahlen die Aussage einfach für jede Zahl gezeigt haben und wollte dann wissen, warum was deshalb im Omegakalkül besser ist als in unserem. Wusste ich auch nicht mehr, er meinte Vollständigkeit.

Abstrakte Datentypen

- Was entspricht abstrakten Datentypen in der Praxis? – Klassen? – Joa, ne, … – Objekte? Ganze Systeme? – Noch schlimmer. Wogegen will man normalerweise programmieren? – Ah, Schnittstellen
- Woraus besteht ein abstrakter Datentyp? – einem Zustand, … – einem? – nein mehreren, aber es ist nur einer „aktiv“ – einer Menge von Initialzuständen, und Operationen
- Woraus bestehen die Operationen? – Kontrakte – Ja, und einen Namen etc. haben sie natürlich
- Wie kann man Kontrakte angeben? – Relational, Vor-Nach-Bedingung, Operational
- Wie macht das KIV? – Operational
- Was nützt uns der Kontrakt, wenn wir dann eh schon das Programm hinschreiben müssen? – Wir wählen einfach mit choose das richtige, in der konkreten Implementierung macht man das nicht
- Dann noch das restliche Standardzeug mit Refinement, Traces etc., da erinner ich mich jetzt nicht mehr an die Details. War aber nichts dabei, was sie nicht fast alle fragen

KIV-Beweis

Hatte auch das Terminieren von Mergesort, das hat u.a. Fabian ja schon perfekt beschrieben.