Zwischenprüfung Formale Methoden

Allgemeines

- Gerhard hat die Fragen gestellt
- Ich hab zwar direkt nach der Prüfung angefangen, die Fragen aus dem Gedächtnis aufzuschreiben, aber ich stand so unter Strom, dass ich glaube ich ein paar Fragen vergessen habe.
- Simon hatte wohl mehr oder weniger die gleiche Prüfung.

Prüfungsverlauf

Aussagenlogik

- Beweis einer aussagenlogischen Tautologie in einem Kalkül meiner Wahl (an die genaue Formel kann ich mich leider nicht erinnern, es kam u.a. (→L), (→R), (∧L), (∧R), (A) vor)
- Wir konnten es herleiten und wissen deshalb, dass es sich um eine Tautologie handeln muss. Welche Eigenschaft des Kalküls ist das? *Korrektheit*

Prädikatenlogik

- Schreibe die Definition von (∃L) auf.
- Welche Regeln für Gleichheit kommen bei PL dazu? *(Refl), (Eq). Grobe Bedeutung reicht*
- Wendet der Simplifier auch (Eq) an? *Rewrite-Regeln*
- Welche Simplifierregeln gibt es insgesamt? *Äquivalenz-, Ersetzungsregeln und Assoz./Komm.*
- Schreibe die Form von Äquivalenz-/Ersetzungsregeln hin und erkläre, was der Simplifier mit ihnen macht. *(„Wenn er etwas von der gleichen Form wie t findet, …“ ist ihnen zu ungenau)*

Generiertheit

- Was wollen wir von einer Spezifikation, und was bedeutet es jeweils? *Konsistenz, Monomorphie*
- Warum reichen uns Axiome nicht immer? *Natürliche Zahlen nicht monomorph spezifizierbar*
- Was machen wir deshalb? *Generiertheitsklauseln ergänzen*
- Woraus bestehen diese? *Sorten mit zugehörigen Konstruktoren (Funktionen)*
- Was sagen Generiertheitsklauseln aus? *Auswertung der Konstruktorterme auf die Trägermänge ist surjektiv*

Freie Generiertheit

- Was ist uns noch lieber? *freie Generiertheit*
- Was ist dort anders? *Auswertung zusätzlich injektiv*
- Was sind Beispiele für freie Datentypen? *Aufzählung, Tupel, Listen, Nat, Bäume, …*
- Wie spezifizieren wir freie Datentypen in KIV? *Data specification*
- Spezialfrage für Leute, die schon mal an KIV mitgearbeitet haben: Was ist das Scala-Äquivalent zu freien Datentypen (Alex verdreht die Augen und lacht, Gerhard rechtfertigt die Relevanz des Themas 😄) *sealed abstract classes*. Was ist das Scala-Äquivalent zu den Konstruktoren: *case classes*

- Sind data specifications monomorph/konsistent? *konsitent ja, monomorph nur modulo Parametrisierung/unspezifizierter Selektoren*

Anreicherung

- Was tun wir, wenn wir Funktionalität zu data specifications hinzufügen wollen? *Anreicherung*
- Welche Eigenschaften wollen wir dabei? *Hierarchiepersitenz, Eindeutigkeit*
- Wie sind die beiden definiert?
- Wie erreichen wir das beim Hinzufügen neuer Funktionen? *(nicht-)rekursive Definition*

KIV-Verwendung

- Noch auf Papier: rekursive Definition für den Schnitt zweier Listen
  - $intersect - base : [] \cap y = []$
  - $intersect - rec1 : a \in y \rightarrow (a' + x) \cap y = a' + (x \cap y)$
  - $intersect - rec2 : \neg a \in y \rightarrow (a' + x) \cap y = x \cap y$
- Beweis mit KIV (inkl. Einstellen von Simplifierregeln/Heuristiken) von $a \in x \cap append(y, z) \leftrightarrow a \in x \wedge (a \in y \vee a \in z)$. *Strukturelle Induktion über x, dann Case distinction und Lemmaanwendung*