Vorlesung vom 06.02.06: Zusammenfassung, Abschließende Bemerkungen & Ausblick

Zusammenfassung

- Teil I: *Grundlagen*
  - *Formale Logik*
- Teil II: *Modellierung*
  - *Datentypen*, *imperative* und *funktionale* Programme
- Teil III: *Spezifikation*
  - *CASL*, *Z* und *formale Programmentwicklung*

Teil I: Grundlagen

- *Formale Logik*: Grundlage der Informatik
- *Grundbegriffe*:
  - *Konsistenz*, *Vollständigkeit*, *Entscheidbarkeit*
- *Formales Schließen* in *Kalkülen*
  - *Natürliches Schließen*, *Sequenzenkalkül*

Formale Logiken: von einfach bis kompliziert

- *Offene Aussagenlogik* $A \wedge B$
  - *Entscheidbar*, *vollständig*
- *Prädikatenlogik erster Stufe* $\forall\, x. P(x)$
  - *Unentscheidbar*, *vollständig*
- *Prädikatenlogik höherer Stufe* $\forall\, P. P \longrightarrow P$
  - *Unentscheidbar*, *unvollständig*
- *Gödel's Unvollständigkeitssatz*
  - Natürliche Zahlen + Konsistenz = Unvollständigkeit

Klassisch vs. konstruktiv

- *Klassische* Logik: *tertium non datur*
  - HOL: Klassisches Logik höherer Stufe, einfach getypt (Church)
- *Konstruktivistische* Logik & *Curry-Howard-Isomorphismus*
  - *Finitärer* Konstruktivismus
  - *Konstruktivismus* (Bishop, Dummett)
  - *Beweis* in konstruktiver Logik = Term (*Program*) im λ-Kalkül
- *Intuitionistische* Logik (Brouwer)

Theorembeweiser

- `Isabelle`: klassische Logik höherer Stufe
- Gleiche/ähnliche Logik: `HOL`, `HOLlight`, `ProofPower`
- *Typtheorien*:
  - *Abhängige Typen* (dependent types, z.B.. $\Pi n : \mathbb{N}. Vec(n)$).
  - Calculus of constructions: `Coq`
  - Martin-Löf-Typtheorie: `Alf`
- `PVS`: eigene Logik (abhängige Typen, Subtypen, . . .)

Teil II: Modellierung

- Modellierung von *Datentypen*
  - *Isabelle*: Konstruktion von Datentypen durch *Repräsentation und Abstraktion*
  - Algebraische *Datentypen*
- Modellierung von *rekursiven Funktionen*
  - In HOL: nur totale Funktionen; wohlfundierte Rekursion
  - Cpo's und LCF: alle rekursiven Funktionen, aber Stetigkeitsbeweise
- Modellierung von *imperative Programme*
  - *Denotationelle*, *operationelle* und *axiomatische* Semantik
  - Explizites Modell des *Zustandsübergangs*

Teil III: Formale Softwareentwicklung

- Der *Entwicklungszyklus*
  - *Anforderungsspezifikation*: axiomatisch, unvollständig
  - *Entwurfsspezifikation*: konsistent, vollständig
  - *Ausführbare* Spezifikation
  - *Implementation*
- *Prozessmodelle*:
  - *Wasserfall-Modell*, *V-Modell*, *Spiralmodell*
  - Evolutionäre formale Softwareentwicklung

Spezifikationsformalismen

- *Algebraisch* (CASL)
  - Basiert auf *universeller Algebra*
  - *Axiomatisch*
- *Modellbasiert* (Z)
  - Basiert auf *getypter Mengenlehre*
  - *Schemakalkül*, *imperative* Programme
- Weitere Sprachen und Werkzeuge
  - *VDM*; *B*; *ASM*
  - *Model Checker* — *automatische* Beweisewerkzeuge
  - Viele Werkzeuge und Sprachen für *nebenläufige Systeme* (CSP/FDR, SPIN)

Formale Programmentwicklung

- *Verfeinerung*
  - Theoriemorphisms $\sigma : SP_1 \rightarrow SP_2$
  - Endrekursion
  - Transformationsregeln (z.B. Divide-and-conquer)
- *Datenverfeinerung*
  - *Observational* and *behavioural* equivalence: gleich ist nicht gleich
  - Beispiel: Warteschlangen

Anwendung formaler Methoden in der Praxis

- Z: CICS, B: Meteor (führerlose Metro)
- *Praxis High Integrity Software* (UK): Z & SPARK
- Microsoft:
  - *SLAM*: Automatische Verifikation von Treibern (Kombination aus *model checking* und *Abstraktion*)
  - *Spec#*: Objektorientierte Spezifikation (Interaktives Beweisen); *AsmL*: Abstract state machines (model checking)
- Intel: Verifikation der *Fließkommarithmetik* für Pentium IA-32, IA-64
  - Kombination aus *model checking* (SDT, symbolic trajectory evaluation) und *interaktivem Beweisen* (Forte & FL)

Formale Methoden in Bremen

- *Deadlock- & Livelock-Analyse* für die *ISS*
- Die *CASL-Sprachfamilie*
- Der autonome Rollstuhl *Rolland*
- Sicherheit für *autonome mobile Systeme*

Die Zukunft

- . . . hat gerade erst angefangen
- Formale Methoden werden immer mehr Standard
- Projekt *VeriSoft*

Nächstes Semester

- *Kurs Formale Methoden der Softwareentwicklung II*:
  - Forschungsnäher
  - Verifikation *imperativer Programme*, oder
  - *Formale Programmentwicklung*
- *Seminar Formale Methoden der Softwareentwicklung*
- *Darüber hinaus*:
  - Diplomarbeiten
  - Studentische Hilfskräfte