Rechnergestützte Beweissysteme

H. Peter Gumm

Wintersemester 2007/08

Beweissysteme für Informatiker ?

- Logik ist fundamental für Informatik
  - Programmiersprachen
  - Hardwaredesign
  - Algorithmen (Invarianten)
  - Spezifikationen
- Spezifikationen nachrechnen = Beweisen
  - Benötigt: Beweisassistent
  - Beweisen ist schwerer als „ausrechnen“
  - Dennoch: viele Beweisteile sind „trivial“ und sollten , automatisch erledigt werden

Symbolisches Rechnen

- Mathematiker haben *Mathematica*, *Maple*, *MuPad*, etc.
-
  - können symbolisch rechnen, aber nicht in Logikkalkülen.
  - algebraische Formelmanipulation
  - Systeme denkbar, die analog mit *logischen* Formeln umgehen ?
- Gibt es Logikrechner ?
  - Wo kann man sie einsetzen ?
  - Wie kann man sie bedienen ?
  - Wie sollen sie funktionieren ?

Anwendung von Beweisern

- Korrektheit von Algorithmen
  - kritische Anwendungen
  - Kernalgorithmen (Betriebssysteme, Compiler, ..)
  - unsinnig für Spiele, etc.
- Korrektheit von Datenstrukturen
  - Auffinden von Inkonsistenzen
  - Beweis der korrekten Implementierung
  - Finden von Invarianten
  - Erkennen von Sonderfällen

Verteilte Systeme

- extrem fehleranfällig
- Testen ist Stochern im Nebel
- unwahrscheinliche Kombinationen entgehen den Tests, tauchen aber in der Praxis auf (Murphy's law)

- nur ein formaler Beweis verschafft Sicherheit
  - Mathematische Beweise (Bleistift und Papier) nicht vertrauenswürdig
  - Maschinengeprüfte Beweise sicherer

- oft reicht es nur kritische Teile zu beweisen
    - implizite Annahmen
    - kein deadlock
    - wechselseitiger Ausschluss

Hardware

- Korrektheit von Hardware
  - Testen und Verifizieren vor der Konstruktion
  - zeitabhängige Komponenten
  - Hazards
  - korrektes pipelining
  - cache-coherence
  - keine Blockierung
- Konstruktion von Hardware aus einer Spezifikation
  - Hardwaresynthese
  - Implementierung = Beweis
  - Lambda-System

Beweise im Systementwurf

- Korrekter Systementwurf
  - frühzeitiges Erkennen von Designfehlern
  - Beweise impliziter Annahmen (putative theorems)
  - Tests vor der Implementierung
  - Was wäre wenn ... ?
  - Erkennen von Sonderfällen
  - Randbedingungen

Mathematische Anwendungen

- Erledigung einfacher Beweise
  - Erledigung trivialer Fälle
    - Induktionsanfang
    - evtl. Induktionsschritt
  - daher Konzentration auf das Wesentliche:
    - Induktionshypothese
- Überprüfung kritischer Beweisteile
  - Erkennen impliziter Annahmen
  - Erkennen von Beweislücken