Formale Methoden der Softwaretechnik 1
Vorlesung vom 02.12.09: Beweisprozeduren und Funktionsdefinitionen

Christoph Lüth, Lutz Schröder

Universität Bremen

Wintersemester 2009/10

Fahrplan

- Teil I: Grundlagen der Formalen Logik
- Teil II: Arbeiten mit Isabelle
  - Grundlagen von Isabelle
  - Logik höherer Ordnung in Isabelle
  - Isabelle/HOL
  - Beweise über funktionale Programme in Isabelle/HOL
  - Beweisen mit Isabelle: Simplifikation, Taktiken
- Teil III: Modellierung imperative Programme

Überblick

- Automatische Beweisprozeduren im Überblick
- Definition von Funktionen

Simplifikation

- Simplifikation ist Termersetzung:
  - Gegeben Theorem $s = t$, ersetze s durch t.
- Benutzung: `apply (simp)`
- Nutzt Gleichungen und Ungleichungen:
  - Funktionsdefinitionen
  - Vereinfachungsregeln für Datentypen
  - Deklarierte Theoreme
  - Annahmen des lokalen Subgoals
- Benutzt bedingte Gleichungen: $s_1 = t_1, \ldots, s_n = t_n \Longrightarrow s = t$
  - Ersetzt s durch t, wenn Gleichungen $s_1 = t_1 \ldots s_n = t_n$ rekursiv gezeigt werden können.
- Instantiiert keine Meta-Variablen
- Erzeugt keine neuen Subgoals, nur Vereinfachung

Klassische Beweiser

- Beweisplaner: `blast`
  - Konstruiert Beweis durch Suche
  - Gelingt oder schlägt fehl: keine neuen Subgoals
- Klassischer Beweiser: `clarify`
  - Wendet Einführungs- und Eliminationsregeln systematisch an
  - Keine neuen Subgoals, nur Vereinfachung
  - Sicher: keine unbeweisbaren Subgoals
  - `clarsimp`: Kombination mit Simplifikation
- Vollautomatisch: `auto`
  - Kombination verschiedener Beweiser
  - Instantiiert `Meta-Variablen`, erzeugt neue Subgoals
  - Unsicher: kann unbeweisbare Subgoals erzeugen

Definition von Funktionen

- In HOL: alle Funktionen total
  - Need-to-know-Prinzip: unbeweisbar ist undefiniert
- Der einfache Fall: primitiv rekursiv
- Der allgemeinere Fall
- Immer mit Terminationsbeweis

Einfache Rekursion

fun f :: τ

where

equations

Volle Rekursion

function f :: τ

where

equations

by *completeness-proof*

termination by *termination-proof*

Zusammenfassung

- Automatische Beweisprozeduren: `simp`, `blast`, `clarify`, `auto`
- Funktionsdefinition: `fun` und `function`
  - Fast wie in Haskell, aber immer total.