Rechnerübung zu Theorembeweiser und ihre Anwendungen

Prof. Dr.-Ing. Gregor Snelting
Dipl.-Inf. Univ. Daniel Wasserrab

Lehrstuhl Programmierparadigmen
IPD Snelting
Universität Karlsruhe (TH)

Teil I

Einführung in Isabelle

Installation

Wurde für uns schon gemacht :), aber falls es jemand zu Hause installieren möchte:

- Auf Seite `http://isabelle.in.tum.de/download.html` gehen
- Isabelle (online ist neue Version 2009), ProofGeneral, PolyML und HOL herunterladen und installieren (ist erklärt)

Starten des ProofGeneral in `emacs` (Groß- und Kleinschreibung beachten!):

Poolraum: `/usr/local/Isabelle/bin/Isabelle` (`-p xemacs`)

Download: *`Isabelle-Pfad`*`/isabelle emacs` (`-p xemacs`)

Die Option `-p xemacs` nach dem jeweiligen Befehl (also z.B. `isabelle emacs -p xemacs`) startet ProofGeneral in `xemacs`

ProofGeneral

- Isabelle-GUI, Plugin für (X)Emacs
- verwendet XSymbols
- bietet Isabelle-spezifische Menüs
- bietet Buttons zum Steuern des Beweisprozesses (Retract, Undo, Next, Use, Goto, Stop)

Dateien in Isabelle

- Isabelle-Dateien haben die Endung .thy
- eine Datei beginnt mit:
  theory *Dateiname* **imports** *Basisdateiname* (Standard: **Main**) **begin**
- durch die Buttons leitet man den Beweisprozess,
  blau unterlegter Text wurde bearbeitet
- Dateiende wird durch **end** ausgedrückt

Aussagen

- In Isabelle werden zu zeigende Aussagen mit dem Schlüsselwort **lemma** (auch möglich: **corollary** und **theorem**)
- danach folgt optional ein Name, beendet durch :
- danach folgt die zu zeigende Aussage in Anführungszeichen
- darauf werden mittels **apply** Regeln angewandt

Regeln

- allgemeine Form: $[\![P1;\ P2;\ P3]\!] \Longrightarrow Q$
- $P1$, $P2$, $P3$ sind Prämissen der Regel (Annahmen)
- Q die Konklusion (Schlussfolgerung)
- $\Longrightarrow$ trennt Prämissen und Konklusion
- Also: "Wenn $P1$, $P2$ und $P3$, dann Q"
- Beispiel Modus Ponens: $[\![P \longrightarrow Q;\ P]\!] \Longrightarrow Q$

Operatoren

Es gibt folgende logische Operatoren in Isabelle/HOL:

- Negation $\neg$ (geschrieben `\<not>`)
- Konjunktion $\wedge$ (geschrieben `\<and>` oder kurz: /\)
- Disjunktion $\vee$ (geschrieben `\<or>` oder kurz: \/)
- Implikation $\longrightarrow$, nicht verwechseln mit $\Longrightarrow$!
  (geschrieben `\<longrightarrow>` oder kurz: $--$ >)
- Gleichheit =
- Ungleichheit $\neq$ (geschrieben `\<noteq>` oder kurz: = /)

Introduktion und Elimination

- Jeder Operator besitzt eine Introduktionsregel, wobei der Operator in der Konklusion steht (Standardname $\ldots I$)
  "Was brauche ich, damit die Formel gilt?"
  Beispiel: $conjI$: $[\![P;\ Q]\!] \Longrightarrow P \wedge Q$
- Jeder Operator besitzt eine Eliminationsregel, wobei der Operator in der ersten Prämisse steht (Standardname $\ldots E$)
  "Was kann ich aus der Formel folgern?"
  Beispiel: $conjE$: $[\![P \wedge Q;\ [\![P;\ Q]\!] \Longrightarrow R]\!] \Longrightarrow R$
- Regeln kann ich mir mittels **thm** *lemma-Name* anzeigen lassen

Anwendung von Regeln in Isabelle

- Anwendung von Regeln spaltet das Beweisziel evtl. in subgoals auf
- Jedes subgoal muss erst gezeigt werden, bevor das nächste gezeigt werden kann
- Wenn die Konklusion einer Prämisse entspricht:
  apply assumption
- Wenn man eine (Introduktions-)Regel auf eine Konklusion anwenden möchte: **apply**(rule *Regel-Name*)
  ersetzt Beweis der Konklusion der Regel durch (meist mehrere) Beweise der Prämissen der Regel
  vorher vorhandene Prämissen werden wieder Prämissen der Beweise

Anwendung von Regeln in Isabelle

- Wenn man eine (Eliminations-)Regel auf eine Prämisse anwenden möchte: **apply**(erule *Regel-Name*)
  Vorsicht: Das aktuell zu zeigende subgoal muss mit der Konklusion der Regel unifizierbar sein!
  eliminiert die passende Prämisse und ersetzt Beweis der Konklusion der Regel durch Beweise der weiteren Prämissen der Regel
  andere vorhandene Prämissen bleiben wieder erhalten
- Wenn Isabelle meldet: `No subgoals!`, Beenden des Beweises mittels **done**

Und jetzt Sie!

Viel Spass beim Ausprobieren!