universität innsbruck

Summer Term 2023

Interactive Theorem Proving using Isabelle/HOL

Session 13

René Thiemann

Department of Computer Science

Outline

- Further Topics
- Master Projects

Further Topics

Further Type Definition Principles

- coinductive types such as a type for finite and infinite lists, a type for infinite trees (branching or depth or both), etc.
  more information: `isabelle doc datatypes`
- records = named (and extensible) tuples
  more information: `isabelle doc isar-ref` (Chapter 11.6)

Locales

- more generic than type-classes
  - several parameters
  - several instances
- less automation
- more information: `isabelle doc locales`

Syntax

- Isabelle permits to define custom syntax
- examples
  - infix-operators with precedence (via priority grammars)
  - binders, sums, products, ...
  - lists, strings, numbers, ...
- further information: Chapter 8 (Inner Syntax) of `isabelle doc isar-ref`

Isabelle/ML

- Isabelle/ML: implementation language of Isabelle (standard ML + Isabelle library)
- provides internal representations of terms, types, theorems, theories, ...
- can be used to automate tasks
- examples
  - tactics: apply various methods, control can depend on goal structure
  - definitional packages: `fun`, `datatype`, `lift_definition`, ...
  - ...
  - easy example in Demo: automation for characters
- further information: Isabelle sources (CTRL-click on commands), The Isabelle Cookbook

Archive of Formal Proofs

- https://www.isa-afp.org
- collection of various Isabelle formalizations (computer science, mathematics, ...)
- kept up-to-date with Isabelle versions
- freely available
- everyone can contribute
- organized in the way of a scientific journal

IsaFoR and CeTA

- `http://cl-informatik.uibk.ac.at/software/isafor/`
- Isabelle Formalization of Rewriting and Certified Tool Assertions
- certification approach
  - untrusted tools generate certificate, e.g. termination proof via polynomial interpretation
  - IsaFoR contains (among many other techniques)
    - formal proof that polynomial interpretations ensure termination
    - verified functions to check whether given polynomial interpretation was correctly applied
  - CeTA is Haskell file that contains verified functions of IsaFoR (via code generator)
- main developer: Christian Sternagel and René Thiemann
- IsaFoR: 421 theories, 240 000 lines, 14 years of development
  + several theories that have been moved to AFP
- `CeTA.hs`: 57 000 lines of generated Haskell code
- soundness property: if CeTA accepts proof of some property, then property is satisfied

Master Projects

Master Projects

- if you did not yet find a master project, consider Isabelle related topics
- eligible topics: formalization of any non-trivial result, such as
  - some non-trivial algorithm
  - some non-trivial mathematical result
- examples (listed under available master projects)
  - Formalizing Congruence Closure (in an efficient implementation)
  - Formalizing Kleene's Normal Form Theorem
  - Formalizing Termination Techniques for String Rewriting
- examples (further ideas from my side)
  - verify certain optimizations for linear integer arithmetic (e.g., cubes and equalities)
  - verify certain results w.r.t. Nelson–Oppen combination
    (LRA is convex, EUF is convex, deterministic or non-deterministic algorithm, ...)
  - develop and verify abstract DPLL(T) or CDCL(T)
    (e.g., include conflict graphs, but not implementation details such as 2-watched literals)
- come up with your own Isabelle-related topic and discuss it with me

Thank you for your interest!

Stay safe, use