universität innsbruck

Summer Term 2019

Interactive Theorem Proving using Isabelle/HOL

Session 6

Christian Sternagel

Department of Computer Science

Topics

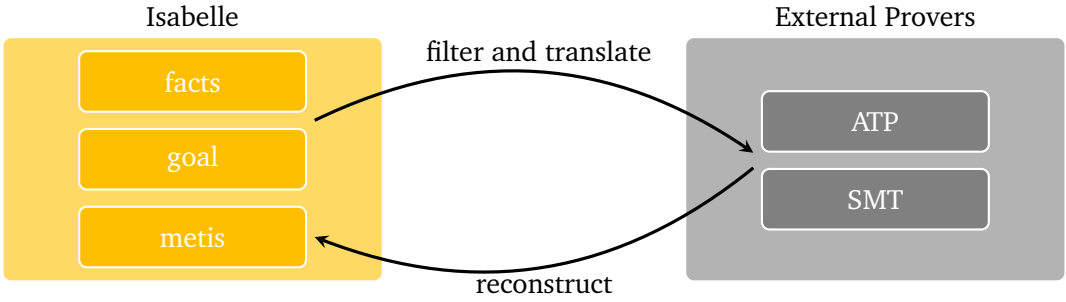
calculational reasoning, case analysis, code generation, computation induction, data type invariants, document preparation, finding theorems, first steps, functional programming in HOL, higher-order logic, history and motivation, induction, inductive definitions, Isabelle basics, Isabelle/Isar, Isabelle/ML, IsaFoR/CeTA, locales, manual termination proofs, multisets, natural deduction, notation, proof methods, PSL: a high-level proof strategy language, rule induction, rule inversion, session management, sets, simplification, sledgehammer, structural induction, structured proof, The Archive of Formal Proofs, the certification approach, total recursive functions, type classes, type definitions, well-foundedness

Overview

- Sledgehammer
- Inductive Definitions
- Rule Inversion and Rule Induction
- Exercises

Sledgehammer

tool that applies automated theorem provers (ATPs) and satisfiability-modulo-theory (SMT) solvers to current subgoal



The metis Method

- metis is Isabelle built-in ATP (first-order ordered resolution and paramodulation)
- its inferences go through Isabelle's proof kernel (correct by construction)
- `metis` *fact** – apply metis using some auxiliary facts

Example – Axiom of Choice

```
lemma "∀x. ∃y. P x y ⟹ ∃f. ∀x. P x (f x)" by metis
```

Isabelle/jEdit's Sledgehammer Panel

- set external provers in "Provers:" input field
- hit "**Apply**" button to run external provers on current subgoal
- hint: set Primary shortcut of command "Sledgehammer panel (Toggle)" to Ctrl+e h

Inductive Predicates

- constant `P :: 'a`$_1$ ⇒ ... ⇒ `'a`$_n$ ⇒ `bool` is n-ary predicate
- inductive predicate `P` is inductively defined, that is, by inference rules
- meaning: input satisfies `P` iff witnessed by arbitrary (finite) application of inference rules
- syntax
  `inductive P :: "'a`$_1$ ⇒ ... ⇒ `'a`$_n$ ⇒ `bool" where ...`
  followed by `|`-separated list of propositions (inference rules)
- generated facts

`P.intros`	inference rules
`P.cases`	case analysis (rule inversion)
`P.induct`	induction (rule induction)
`P.simps`	equational definition

Demo06.thy – Odd Numbers, Inductively

- textual description
  - 1 is odd
  - if n is odd, then also $n + 2$ is odd
- inference rules

$$\frac{}{1 \text{ odd}} \qquad \frac{n \text{ odd}}{n + 2 \text{ odd}}$$

- ```
  inductive is_odd :: "nat ⇒ bool"
    where
      "is_odd (Suc 0)"
    | "is_odd n ⟹ is_odd (Suc (Suc n))"
  ```
  ```

Special Case – Inductively Defined Sets

- given set S, let χ_S be characteristic function such that $\chi_S(x)$ is true iff $x \in S$
- characteristic function is obviously predicate
- inductive sets are common special case and come with special syntax
  `inductive_set S :: "'a`$_1$ `⇒ ... 'a`$_n$ `⇒ 'a set" for c`$_1$ `... c`$_n$ `where`

Demo06.thy – Reflexive Transitive Closure

- (binary) relations encoded by type `('a × 'b) set`
- given relation R, reflexive transitive closure, often written R^*, given by $(x, y) \in R^*$ iff $x\, R\, x_1\, R\, x_2\, R \cdots R\, x_n\, R\, y$ for arbitrary $x_1, x_2, \ldots, x_n$ (think: path in graph)
- ```
  inductive_set star :: "('a × 'a) set ⇒ ('a × 'a) set" for R
    where
      refl [simp]: "(x, x) ∈ star R"
    | step: "(x, y) ∈ R ⟹ (y, z) ∈ star R ⟹ (x, z) ∈ star R"
  ```
  ```

Rule Inversion

- reasoning backwards "which rule could have been used to derive some fact"
- case analysis according to inference rules
- if inductive predicate/set is first of current facts, `cases` applies rule inversion implicitly
- otherwise, use "`cases rule: c.cases`" for inductively defined constant `c`

`Demo06.thy` – Zero is Not Odd

```
lemma is_odd0 [simp]: "is_odd 0 = False" sorry
```

Rule Induction

- induction according to inference rules
- if inductive predicate/set is first of current facts, `induction` applies rule induction implicitly
- otherwise, use "`induction rule: c.induct`" for inductively defined constant `c`
- `case` names are taken from names of inference rules (if any, otherwise numbered)

`Demo06.thy` – If Number is Odd it's Odd

- `lemma is_odd_odd: assumes "is_odd x" shows "odd x" sorry`
- hint: assumed facts (`assumes`) have implicit label `assms`

Demo06.thy – Reflexive Transitive Closure is Transitive

- ```
  lemma
    assumes "(x, y) ∈ star R" and "(y, z) ∈ star R
    shows "(x, z) ∈ star R"
  sorry
  ```
  ```

Exercises (start from `Exercises06.thy`)

URL

http://cl-informatik.uibk.ac.at/teaching/ss19/itp/thys/Exercises06.thy

Further Reading


Jasmin Christian Blanchette.
Hammering Away – A Users' Guide to Sledgehammer for Isabelle/HOL.
Isabelle documentation, 2018.

Important Concepts

- `cases rule:`
- `induction rule:`
- `inductive` (*HOL command*)
- `inductive_set` (*HOL command*)
- `metis` (*HOL method*)
- rule induction
- rule inversion
- `set` (*HOL type*)
- sledgehammer