Isabelle/HOL and the UTP

Part 2: Deduction, Classes and Isar

Simon Foster

University of York

February 1, 2013

Deduction rules

- **Simplification** rules of the form $[\![A_1 \cdots A_n]\!] \Longrightarrow B = C$
- **Introduction** rules
  - of the form $[\![A_1 \cdots A_n]\!] \Longrightarrow B$
  - split a goal matching into n goals, modulo suitable substitutions
  - applied using rule or rule_tac x=A in
- **Destruction/Forward** rules
  - of the form $A \Rightarrow B$
  - adds B as an assumption, removing A if destruction
  - applied using drule and frule
- **Elimination** rules
  - of the form $[\![A; B_1 \Longrightarrow P \cdots B_n \Longrightarrow P]\!] \Longrightarrow P$
  - splits assumption A into n assumptions, from which P must be proved
  - applied using erule
  - induction and case split are key examples

Automated Deduction

- **blast** applies deduction rules recursively with backtracking
- very good solver for set theoretic/first order logic problems
- all-or-nothing – must fully solve goal applied on
- lemmas can be tagged with [intro], [elim] and [dest]
- care is required – **blast** can loop

- **auto** combines **blast** with **simp**
  - not all-or-nothing, applies to all subgoals
- **force** is all-or-nothing **auto** on a single goal
- **clarify** applies rules which do not split the goal
- **safe** applies rules marked as safe (append with !)

Axiomatic type-classes

- A type-class is a polymorphic signature of constants

class *equal* $=$
fixes *eq* $:: \alpha \Rightarrow \alpha \Rightarrow$ *bool* (**infixr** $\approx$ 25)

- Isabelle/HOL allows axioms about these constants

assumes *refl* : $x \approx x$
and *sym* : $x \approx y \Longrightarrow y \approx x$
and *trans* : $[\![x \approx y; y \approx z]\!] \Longrightarrow x \approx z$

- A type-class can extend other type-classes e.g. linorder $\subseteq$ order $\subseteq$ preorder
- instantiation with a type
  - requires declaration of constants + proof of axioms
  - exports all internal definitions and proofs

Sledgehammer

- solve a goal by calling automated theorem provers
- the problem is submitted to 5 ATPs, which may solve it
- the internal theorem prover metis reconstructs the proof
- alternatively, Z3 can be used via smt command
- only useful for first-order problems (e.g. no induction)
- a very helpful tool if you're stuck

Isar

- a natural proof language for Isabelle
- acts as an alternate syntax for proof scripts

Isar	Isabelle

lemma *my_goal* :
  assumes P
  shows $A = B$
proof –
  from *assms* **have** Q
    by *blast*
 	
  thus ?*thesis*
    by *force*
qed
	lemma *my_goal* : $P \Longrightarrow A = B$
	  apply(*subgoal_tac* Q)
	  apply(*force*)
	  apply(*blast*)
	qed