

Isabelle Proof Goals

Proof goals, or judgments:

- The basic shape of proof goal handled by Isabelle
- Local proof state, subgoal

General shape: $\wedge x_1, ..., x_m. \; [\![\; A_1 \; ; \; ... \; ; \; A_n \;]\!] \Rightarrow A$

- x₁,...,x_m: Local variables
- A₁,...,A_n: Local assumptions
- A: local proof goal

Meaning: For all terms $t_1,...,t_m$, if all $A_i[t_1/x_1,...,t_m/x_m]$ are provable then so is $A[t_1/x_1,...,t_m/x_m]$

Global Proof State

An Isabelle proof state consists of number of unproven judgments

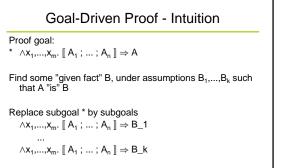
$$1. \land x_{1,1}, \dots, x_{m,1}. \llbracket A_{1,1} \ ; \ \dots \ ; \ A_{n,1} \ \rrbracket \Rightarrow A_1$$

$$k. \land x_{1,k}, \dots, x_{m,k}. \llbracket \mathsf{A}_{1,k}; \dots; \mathsf{A}_{n,k} \rrbracket \Rightarrow \mathsf{A}_k$$

If k = 0 proof is complete

Judgment #1 is the one currently being worked on

Commands to list subgoals, toggle between subgoals, to apply rules to numbered subgoals, etc.



But, "is" is really "is an instance of" so story must be refined

Unification

Substitution:

Mapping σ from variables to terms

[t/x]: Substitution mapping x to t, otherwise the identity

to: Capture-avoiding substitution σ applied to t

Unification:

Try to make terms t and s equal Unifier: Substitution σ on terms s, t such that $s\sigma = t\sigma$ Unification problem: Given t, s, is there a unifier on s, t

Higher-Order Unification

In Isabelle:

Terms are terms in Isabelle = extended λ_{\rightarrow} Terms Equality on terms are modulo α , β , η Variables to be unified are schematic Schematic variables can have function type (= higher order)

Examples:

 $?X \land ?Y =_{\alpha\beta\eta} x \land x$ $P x =_{\alpha\beta\eta} x \wedge x$ $P(?f x) =_{\alpha\beta\eta} ?Y x$

under [x/?X,x/?Y] under [$\lambda x.x \land x/P$] under [\lambda x.x/?f,P/Y]

First Order Unification Decidable Most general unifiers (mgu's) exist: σ is mgu for t and s if σ unifies t and s Whenever σ' unifies t and s then $t\sigma,$ $t\sigma',$ and $s\sigma,$ $s\sigma'$ are both unifiable Exercise 1: Show that [h(?Y)/?X,g(h(?Y))/?Z] is mgu for f(?X,g(?X)) and f(h(?Y),?Z). Applications in e.g. logic programming

Higher Order Unification

HO unification modulo α , β is semi-decidable HO unification modulo α,β,η is undecidable

Higher order pattern:

Term t in β normal form (value in slides for lecture 3) Schematic variables only in head position ?f t₁ ... t_n

Each $t_i \eta$ -convertible to n distinct bound variables

Unification on HO patterns is decidable

Exercises

- Exercise 2: Determine whether each pair of terms is unifiable or not. If it is, exhibit a unifier. If it is not, show why.
- 1. $f(x_1, ?x_2, ?x_2)$ and $f(?y_1, ?y_2, k)$
- 2. $f(x_1, ?x_2, ?x_2)$ and $f(y_1, g ?x_2, k)$
- 3. f (?p x y (h z)) and ?q (g(x,y),h(?r))
- 4. $(p(g x_1) (h x_2) and (q y_2) (h y_1)$
- 5. ?p (g ?q, h z) and f(h ?r, h ?r)

Result: a + (b + c)

Term Rewriting

Use equations t = s as rewrite rules from left to right

Example: Use equations:

- 1. 0 + n = n
- 2. (suc m) + n = suc(m + n)
- 3. (suc $m \leq suc n$) = ($m \leq n$) 4. $(0 \le m) = true$

```
Then:
```

```
0 + suc \ 0 \le (suc \ 0) + x
                                   (by (1))
= suc 0 \le (suc 0) + x
                                  (by (2))
= suc 0 \leq suc (0 + x)
                                  (by (3))
= 0 \le 0 + x
                                  (by (4))
```

= true

More Formally Rewrite rule I = r is applicable to term t[s/x] if: - There is a substitution σ such that $l\sigma$ =_{\alpha\beta\eta} s • σ unifies I and s Result of rewrite is t[so/x] Note: $t[s/x] = t[s\sigma/x]$ Example: Equation: 0 + n = nTerm: a + (0 + (b + c))Substitution: [b+c/n]

Conditional Rewriting

Assume conditional rewrite rule RId: $A_1 \Rightarrow ... \Rightarrow A_n \Rightarrow I = r$

- Rule RId is applicable to term t[s/x] if:
- There is a substitution σ such that $I\sigma =_{\alpha\beta\eta} s$
- σ unifies I and s
- $A_1\sigma,..., A_n\sigma$ are provable

Again result of rewrite is t[s\sigma/x]

Basic Simplification

$$\begin{split} & \text{Goal:} \; [\![\; A_1; \, ... \; ; \; A_m \;]\!] \Rightarrow B \\ & \text{Apply(simp add: eq_1, \, ... \; , \; eq_n)} \\ & \text{Simplify B using} \end{split}$$

- Lemmas with attribute simp
- Rules from primrec and datatype declarations
- Additional lemmas eq1,...,eqn
- Assumptions A_1, \dots, A_m

Variation:

- (simp ... del: ...) removes lemmas from simplification set
- add, del are optional

Termination

Isabelle uses simp-rules (almost) blindly from left to right Termination is *the* big issue

Example: f(x) = g(x), g(x) = f(x)

Rewrite rule

 $[\![\mathsf{A}_1; \, \ldots \, ; \, \mathsf{A}_n]\!] \Rightarrow \mathsf{I} = \mathsf{r}$

suitable for inclusion in simplification set only if rewrite from I to r reduces overall complexity of the global proof state So: I must be "bigger" than r and each A_i

 $n < m = true \Rightarrow (n < suc m) = true$ (may be good) (suc n < m) = true \Rightarrow n < m = true (not good)

Case Splitting

 $\mathsf{P}(\text{if A then s else t}) = (\mathsf{A} \to \mathsf{P}(s)) \land (\neg \mathsf{A} \to \mathsf{P}(t))$ Included in simp by default

$$\begin{split} \mathsf{P}(\text{case t of } 0 \Rightarrow s_1 \mid \mathsf{Suc } n \Rightarrow s_2) \\ &= (t = 0 \rightarrow \mathsf{P}(s_1)) \land (\forall \ n. \ t = \mathsf{Suc } n \rightarrow \mathsf{P}(s_2)) \end{split}$$

Not included - use (simp split: nat.split)

Similar for other datatypes T: T.split

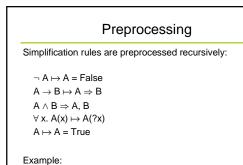
Ordered Rewriting

Problem: ?x + ?y = ?y + ?x does not terminate Isabelle: Use permutative rewrite rules only when term becomes lexicographically smaller Example: ?b + ?a → ?a + ?b but not ?a + ?b → ?b + ?a

For types nat, int, etc.

- Lemmas add_ac sort any sum
- Lemmas times_ac sort any product

Example: (simp add:add_ac) yields (b + c) + a \rightsquigarrow a + (b + c)



. $(p \rightarrow q \land \neg r) \land s$ $\mapsto p = True \Rightarrow q = True, p = True \Rightarrow r = False, s = True$

3