CS477 Formal Software Dev Methods

Elsa L Gunter
2112 SC, UIUC
egunter@illinois.edu
http://courses.engr.illinois.edu/cs477

Slides based in part on previous lectures
by Mahesh Vishwanathan, and by Gul Agha

February 12, 2020

Getting Started with Isabelle

- Possibilities:
  - Use Isabelle on EWS
  - Install on your machine
- On EWS:
  - Assuming you are running an X client, log in to EWS:
    `ssh -Y <netid>@remlnx.ews.illinois.edu`
    - `-Y` used to forward X packets securely
  - To start Isabelle with jedit
    ```
    module load Isabelle/2019
    isabelle jedit
    ```
    - Older versions of Isabelle used emacs and ProofGeneral
    - Will assume `jedit` here
    - First time you use it, it will rebuild all its core theories (takes minutes)

Demo: `my_theory`

System Architecture

Isabelle/jEdit	jEdit based interface
Isar	Isabelle proof scripting language
Isabelle/HOL	Isabelle instance for HOL
Isabelle	generic theorem prover
Standard ML	implementation language

My First Theory File

File name: `my_theory.thy`
Contents:

```
theory my_theory
  imports Main
begin

thm impI

lemma trivial: "A ⟶ A"
  apply (rule impI)
  apply assumption
  done (* end of lemma *)

thm trivial

end (* of theory file *)
```


Overview of Isabelle/HOL

- HOL = Higher-Order Logic
- HOL = Types + Lambda Calculus + Logic
- HOL has
  - datatypes
  - recursive functions
  - logical operators ($\wedge$, $\vee$, $\neg$, $\longrightarrow$, $\forall$, $\exists$, ...)
- Contains propositional logic, first-order logic
- HOL is very similar to a functional programming language
- Higher-order = functions are values, too!
- Booleans are values, too! And predicates are functions.
- We'll start with propositional and first order logic

Formulae (first Approximation)

- Syntax (in decreasing priority):

$$\begin{array}{llll} \textit{form} & ::= & (\textit{form}) & \mid \ \textit{term} = \textit{term} \\ & \mid & \neg \textit{form} & \mid \ \textit{form} \wedge \textit{form} \\ & \mid & \textit{form} \vee \textit{form} & \mid \ \textit{form} \longrightarrow \textit{form} \\ & \mid & \forall x.\ \textit{form} & \mid \ \exists x.\ \textit{form} \end{array}$$

and some others

- Scope of quantifiers: as far to the right as possible

Examples

- $\neg A \wedge B \vee C \equiv ((\neg A) \wedge B) \vee C$
- $A \wedge B = C \equiv A \wedge (B = C)$
- $\forall x.\ P\ x \wedge Q\ x \equiv \forall x.\ (P\ x \wedge Q\ x)$
- $\forall x. \exists y.\ P\ x\ y \wedge Q\ x \equiv \forall x. (\exists y.\ (P\ x\ y \wedge Q\ x))$

Proofs

General schema:

```
lemma name: "..."
apply (...)
:
done
```

First ... theorem statement
(...) are *proof methods*

Top-down Proofs

`sorry`

- "completes" any proof (by giving up, and accepting it)
- Suitable for top-down development of theories:
- Assume lemmas first, prove them later.

Only allowed for interactive proof!

Isabelle Syntax

- Distinct from HOL syntax
- Contains HOL syntax within it
- Also the same as (large subset of) HOL - need to not confuse them

Theory = Module

Syntax:

theory *MyTh*
imports $ImpTh_1$ ... $ImpTh_n$
begin
declarations, definitions, theorems, proofs, ...
end

- *MyTh*: name of theory being built. Must live in file *MyTh*.thy.
- $ImpTh_i$: name of *imported* theories. Importing is transitive.

Meta-logic: Basic Constructs

Implication: $\Longrightarrow$ (==>)
For separating premises and conclusion of theorems / rules

Equality: $\equiv$ (==)
For definitions

Universal Quantifier: $\bigwedge$ (!!)
Usually inserted and removed by Isabelle automatically

Do not use *inside* HOL formulae

Rule/Goal Notation

- $[|A_1;\dots;A_n|] \Longrightarrow B$ abbreviates $A_1 \Longrightarrow \dots \Longrightarrow A_n \Longrightarrow B$
- Means the rule (or potential rule):

$$\frac{A_1;\dots;A_n}{B}$$

- ; $\approx$ "and"
- Note: A theorem is a rule; a rule is a theorem.
- Turn on use of

$$[|A_1;\dots;A_n|]$$

by
`Plugins` $\rightarrow$ `Plugin Options...`$\rightarrow$ `Isabelle` $\rightarrow$ `General` $\rightarrow$ `Print Mode = brackets`

The Proof/Goal State

1. $\bigwedge x_1 \dots x_m.\ [|A_1;\dots;A_n|] \Longrightarrow B$

$x_1 \dots x_m$	Local constants (fixed variables)
$A_1 \dots A_n$	Local subgoals / assumptions
B	Actual goal / conclusion

Proof Basics

- Isabelle uses *Natural Deduction* proofs
  - Uses (modified) *sequent* encoding
- Rule notation:

Rule	Sequent Encoding
$\dfrac{A_1 \dots A_n}{A}$	$[\![A_1,\dots,A_n]\!] \Longrightarrow A$
$\dfrac{A_1 \dots \begin{matrix} B \\ \vdots \\ \hline A_i \end{matrix} \dots A_n}{A}$	$[\![A_1,\dots,B \Longrightarrow A_i,\dots,A_n]\!] \Longrightarrow A$

Natural Deduction

For each logical operator $\oplus$, have two kinds of rules:

Introduction: How can I prove $A \oplus B$?

$$\frac{?}{A \oplus B}$$

Elimination: What can I prove using $A \oplus B$?

$$\frac{\dots A \oplus B \dots}{?}$$

Operational Reading

$$\frac{A_1 \dots A_n}{A}$$

Introduction rule:
To prove A it suffices to prove $A_1 \dots A_n$.

Elimination rule:
If we know A_1 and we want to prove A
it suffices to prove $A_2 \dots A_n$

Natural Deduction for Propositional Logic

$$\frac{\texttt{A}\quad\texttt{B}}{\texttt{A}\wedge\texttt{B}}\ \texttt{conjI} \qquad \frac{\texttt{A}\wedge\texttt{B}\quad [\![\texttt{A};\texttt{B}]\!]\Longrightarrow\texttt{C}}{\texttt{C}}\ \texttt{conjE}$$

$$\frac{\texttt{A}}{\texttt{A}\vee\texttt{B}}\quad\frac{\texttt{B}}{\texttt{A}\vee\texttt{B}}\ \texttt{disjI1/2} \qquad \frac{\texttt{A}\vee\texttt{B}\quad\texttt{A}\Longrightarrow\texttt{C}\quad\texttt{B}\Longrightarrow\texttt{C}}{\texttt{c}}\ \texttt{disjE}$$

$$\frac{\texttt{A}\Longrightarrow\texttt{B}}{\texttt{A}\longrightarrow\texttt{B}}\ \texttt{impI} \qquad \frac{\texttt{A}\longrightarrow\texttt{B}\quad\texttt{A}\quad\texttt{B}\Longrightarrow\texttt{C}}{\texttt{C}}\ \texttt{impE}$$

$$\frac{\texttt{A}\Longrightarrow\texttt{False}}{\neg\texttt{A}}\ \texttt{notI} \qquad \frac{\neg\texttt{A}\quad\texttt{A}}{\texttt{B}}\ \texttt{notE}$$

Natural Deduction for Propositional Logic

$$\frac{\texttt{A}\Longrightarrow\texttt{B}\quad\texttt{B}\Longrightarrow\texttt{A}}{\texttt{A}=\texttt{B}}\ \texttt{iffI}$$

$$\frac{\texttt{A}=\texttt{B}\quad\texttt{A}}{\texttt{B}}\ \texttt{iffD1} \qquad \frac{\texttt{A}=\texttt{B}\quad\texttt{B}}{\texttt{A}}\ \texttt{iffD2}$$

More Rules

$$\frac{\texttt{A}\wedge\texttt{B}}{\texttt{A}}\ \mathit{conjunct1} \qquad \frac{\texttt{A}\wedge\texttt{B}}{\texttt{B}}\ \mathit{conjunct2}$$

$$\frac{\texttt{A}\longrightarrow\texttt{B}\quad\texttt{A}}{\texttt{B}}\ \mathit{mp}$$

Compare to elimination rules:

$$\frac{\texttt{A}\wedge\texttt{B}\quad [\![\texttt{A};\texttt{B}]\!]\Longrightarrow\texttt{C}}{\texttt{C}}\ \mathit{conjE} \qquad \frac{\texttt{A}\longrightarrow\texttt{B}\quad\texttt{A}\quad\texttt{B}\Longrightarrow\texttt{C}}{\texttt{C}}\ \mathit{impE}$$

"Classical" Rules

$$\frac{\neg\texttt{A}\Longrightarrow\texttt{False}}{\texttt{A}}\ \mathit{ccontr} \qquad \frac{\neg\texttt{A}\Longrightarrow\texttt{A}}{\texttt{A}}\ \mathit{classical}$$

- `ccontr` and `classical` are not derivable from the Natural Deduction rules.
- They make the logic *"classical"*, i.e. *"non-constructive* or *"non-intuitionistic"*.

Proof by Assumption

$$\frac{\texttt{A}_1\ldots\texttt{A}_i\ldots\texttt{A}_n}{\texttt{A}_i}$$

- Proof method: `assumption`
- Use:

  `apply assumption`

- Proves:

  $[\![\texttt{A}_1;\ldots;\texttt{A}_n]\!]\Longrightarrow\texttt{A}$

  by unifying A with one of the A_i

Rule Application: The Rough Idea

- Applying rule $[\![A_1;\ldots;A_n]\!]\Longrightarrow A$ to subgoal C:
  - Unify A and C
  - Replace C with n new subgoals: $A_1'\ \ldots\ A_n'$
- Backwards reduction, like in Prolog
- Example:
  Rule: $[\![?P;?Q]\!]\Longrightarrow ?P\wedge ?Q$
  Subgoal: 1. $\texttt{A}\wedge\texttt{B}$
- Resulting Subgoals :
  1. A
  2. B

Rule Application: More Complete Idea

Applying rule $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$ to subgoal C:

- Unify A and C with (meta)-substitution σ
- Specialize goal to $\sigma(C)$
- Replace C with n new subgoals: $\sigma(A_1) \; \ldots \; \sigma(A_n)$

Note: schematic variables in C treated as existential variables
Does there exist value for $?X$ in C that makes C true?
(Still not the whole story)

`rule` Application

Rule: $[\![A_1; \ldots; A_n]\!] \Longrightarrow A$

Subgoal: 1. $[\![B_1; \ldots; B_m]\!] \Longrightarrow C$

Substitution: $\sigma(A) \equiv \sigma(C)$

New subgoals: 1. $[\![\sigma(B_1); \ldots; \sigma(B_m)]\!] \Longrightarrow \sigma(A_1)$
$\vdots$
n. $[\![\sigma(B_1); \ldots; \sigma(B_m)]\!] \Longrightarrow \sigma(A_n)$

Proves: $[\![\sigma(B_1); \ldots; \sigma(B_m)]\!] \Longrightarrow \sigma(C)$

Command: `apply (rule` <*rulename*>`)`

Applying Elimination Rules

`apply (erule` <*elim-rule*>`)`

Like `rule` but also

- Unifies first premise of rule with an assumption
- Eliminates that assumption instead of conclusion

Example

Rule: $[\![?P \wedge ?Q; [\![?P; ?Q]\!] \Longrightarrow ?R]\!] \Longrightarrow ?R$

Subgoal: 1. $[\![X; A \wedge B; Y]\!] \Longrightarrow Z$

Unification: $?P \wedge ?Q \equiv A \wedge B$ and $?R \equiv Z$
$\{?P \mapsto A;\ ?Q \mapsto B;\ ?R \mapsto Z\}$

New subgoal: 1. $[\![X; Y]\!] \Longrightarrow [\![A; B]\!] \Longrightarrow Z$

Same as: 1.$[\![X; Y; A; B]\!] \Longrightarrow Z$

Demo: `my_theory`