### CS477 Formal Software Development 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 6, 2014

伺 ト イヨ ト イヨト

# Getting Started with Isabelle

- Choice
  - Use Isabelle on EWS
  - Install on your machine
  - Both
- 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 /class/cs477/bin/isabelle jedit
- Older versions of Isabelle used emacs and ProofGeneral
- Will assume jedit here

伺下 イヨト イヨト

# My First Theory File

File name: my\_theory.thy Contents:

theory my\_theory imports Main begin

thm impI

```
lemma trivial: "A \longrightarrow A"
apply (rule impI)
apply assumption
done (* of lemma *)
```

```
thm trivial
```

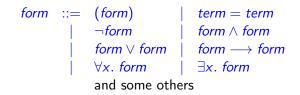
end (\* of theory file \*)

## Overview of Isabelle/HOL

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

▲□ ▶ ▲ □ ▶ ▲ □ ▶

• Syntax (in decreasing priority):



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

個 と くき とくき とう

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

・ 何 ト ・ ヨ ト ・ ヨ ト

General schema:

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

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

◆□ > ◆圖 > ◆臣 > ◆臣 > ─ 臣

#### 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!

- Distinct from HOL syntax
- Contains HOL syntax within it
- Also the same as HOL need to not confuse them

・ 何 ト ・ ヨ ト ・ ヨ ト

Syntax:

theory MyThimports  $ImpTh_1 \ldots ImpTh_n$ begin declarations, definitions, theorems, proofs, ... end

- MyTh: name of theory being built. Must live in file MyTh.thy.
- ImpTh<sub>i</sub>: name of imported theories. Importing is transitive.

▲圖▶ ▲ 圖▶ ▲ 圖▶ …

### Implication: $\implies$ (==>)

For separating premises and conclusion of theorems / rules

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

### Universal Quantifier: $\Lambda$ (!!)

Usually inserted and removed by Isabelle automatically

Do not use inside HOL formulae

abbreviates

 $[|A_1; \ldots; A_n|] \Longrightarrow B$  $A_1 \Longrightarrow \ldots \Longrightarrow A_n \Longrightarrow B$ and means the rule (or potential rule):

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

; 
$$\approx$$
 "and"

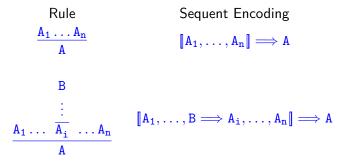
Note: A theorem is a rule; a rule is a theorem.

★個 ▶ ★ 注 ▶ ★ 注 ▶ ● 注

- 1.  $\Lambda x_1 \dots x_m$ .  $[|A_1; \dots; A_n|] \Longrightarrow B$
- $x_1 \dots x_m$ Local constants (fixed variables) $A_1 \dots A_n$ Local assumptionsBActual (sub)goal

### • Isabelle uses Natural Deduction proofs

- Uses (modified) sequent encoding
- Rule notation:



▲口> ▲圖> ▲注> ▲注> 三注

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  $A \oplus B$ ?

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

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

Introduction rule:

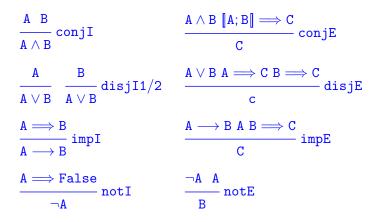
To prove A it suffices to prove  $A_1 \dots A_n$ .

#### Elimination rule:

If we know  $A_1$  and we want to prove Ait suffices to prove  $A_2 \dots A_n$ 

イロト イ理ト イヨト イヨト 二度

### Natural Deduction for Propositional Logic



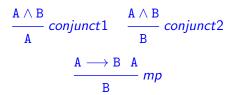
・ロン ・聞と ・ヨン ・ヨン … 油

### Natural Deduction for Propositional Logic

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

.

イロト イ理ト イヨト イヨト



Compare to elimination rules:

 $\frac{A \wedge B \ \|A; B\| \Longrightarrow C}{C} \ conjE \quad \frac{A \longrightarrow B \ A \ B \Longrightarrow C}{C} \ impE$ 

$$\frac{\cancel{A} \Longrightarrow \texttt{False}}{\cancel{A}} \textit{ccontr} \quad \frac{\cancel{A} \Longrightarrow \cancel{A}}{\cancel{A}} \textit{classical}$$

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

・ 同 ト ・ ヨ ト ・ ヨ ト

# Proof by Assumption

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

- Proof method: assumption
- Use:

Proves:

apply assumption  $[\![A_1;\ldots;A_n]\!] \Longrightarrow A$ 

by unifying A with one of the  $A_i$ 

3

▲圖▶ ▲屋▶ ▲屋▶

# Rule Application: The Rough Idea

Applying rule  $[A_1; \ldots; A_n] \implies A$  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:  $[?P; ?Q] \implies ?P \land ?Q$ subgoal: 1.  $A \land B$ 

Result: 1. A2. B

▲圖▶ ▲ 国▶ ▲ 国▶ …

# Rule Application: More Complete Idea

Applying rule  $[A_1; \ldots; A_n] \implies A$  to subgoal *C*:

- Unify A and C with (meta)-substitution  $\sigma$
- 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)

副下 《唐下 《唐下

# rule Application

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

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

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

New subgoals:

Proves: Command:

1. 
$$[\sigma(B_1); ...; \sigma(B_m)] \Longrightarrow \sigma(A_1)$$
  
:  
n.  $[\sigma(B_1); ...; \sigma(B_m)] \Longrightarrow \sigma(A_n)$   
 $\sigma(B_1); ...; \sigma(B_m)] \Longrightarrow \sigma(C)$   
apply (rule )

3

### apply (erule < elim-rule>)

Like rule but also

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

・ 同 ト ・ 三 ト ・ 三 ト

Rule:	$\llbracket ?P \land ?Q; \llbracket ?P; ?Q \rrbracket \Longrightarrow ?R \rrbracket \Longrightarrow ?R$
Subgoal:	$1. \ [\![X; A \land B; Y]\!] \Longrightarrow Z$
Unification:	$P \land Q \equiv A \land B$ and $R \equiv Z$
New subgoal:	$1. \ \llbracket X; Y \rrbracket \Longrightarrow \llbracket A; B \rrbracket \Longrightarrow Z$
Same as:	$1.[\![X;Y;A;B]\!] \Longrightarrow Z$

◆□▶ ◆圖▶ ◆理▶ ◆理▶ ○理