

# $\label{eq:product} What Is Higher Order Logic? \\ Propositional logic \\ No quantifiers \\ All variables have type bool \\ First Order Logic \\ Quantification over values of base type \\ Terms and formulas are syntactically distinct \\ Higher Order Logic \\ Quantification over functions and predicates \\ Consistency by typing \\ Formula = term of type bool \\ Predicate = function with codomain bool \\ \lambda_{\rightarrow} + a few types and constants \\ \end{array}$

## Natural Deduction

Two kinds of rules for each logical operator  $\oplus$  **Introduction rules:** 

### How can $A \oplus B$ be proved?

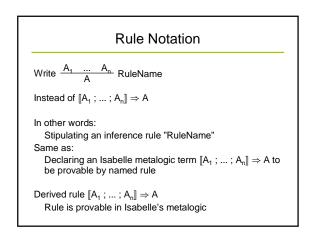
Elimination rules:

What can be inferred from  $A \oplus B$ ?

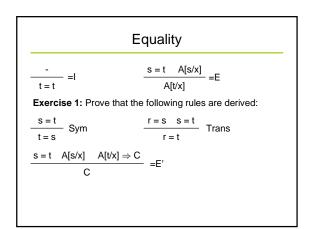
### Natural deduction calculus:

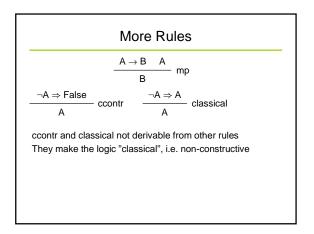
Proof trees may have unproven leaves = assumptions Assumptions can be introduced and discharged Sequent calculus:

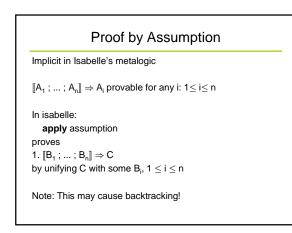
All assumptions (and alternative conclusions) represented explicitly in proof judgments



Natural Deduction	on, Propositional Logic
$\frac{A B}{A \wedge B} \wedge I$	$ \begin{array}{c c} A \land B & \llbracket A;B \rrbracket \Rightarrow C \\ \hline C & \land E \end{array} $
$\frac{A}{A \lor B} \frac{B}{A \lor B} \lor I1/2$	
$\frac{A\RightarrowB}{A\toB}\RightarrowI$	$\frac{A \Rightarrow B  A  B \Rightarrow C}{C} \Rightarrow E$
$\frac{A \Rightarrow B  B \Rightarrow A}{A = B}  \text{iffI}$	$\frac{A = B}{A \Rightarrow B} \frac{A = B}{B \Rightarrow A} \text{ iffD1/2}$
$\frac{A \Rightarrow False}{\neg A} \neg I$	$\frac{\neg A  A}{C} \neg E \qquad D \text{ for } $







Rule Application	
Rule: $\llbracket A_1;; A_n \rrbracket \Rightarrow A$	
Subgoal:	
1. $\llbracket B_1;; B_m \rrbracket \Rightarrow C$	
Substitution:	
$\sigma(A) == \sigma(C)$	
(recall: == means "same term as")	
New subgoals:	
1. $\sigma(\llbracket B_1;; B_m \rrbracket \Rightarrow A_1)$	
n. $\sigma(\llbracket B_1;; B_m \rrbracket \Rightarrow A_n)$	
Command:	
apply (rule <rulename>)</rulename>	



Exercise 2: Prove the following in HOL. Pen and paper is fine. If you use Isabelle, use only basic HOL rules corresponding to rules given in previous slides – no simplifiers

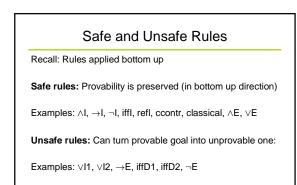
- 1. A  $\vee$  (B  $\vee$  C)  $\rightarrow$  (A  $\vee$  B)  $\vee$  C
- 2. (A  $\rightarrow$  (B  $\rightarrow$  C))  $\rightarrow$  (A  $\wedge$  B)  $\rightarrow$  C
- 3.  $A \lor A \rightarrow A \land A$
- 4.  $A \lor B \rightarrow \neg A \rightarrow B$
- 5. A  $\wedge$  (B  $\vee$  C)  $\rightarrow$  (A  $\wedge$  B)  $\vee$  C
- 6.  $(A \land \neg B) \lor (B \land \neg A) = (A = \neg B)$
- 7.  $\neg(A \land B) \rightarrow (\neg A) \lor (\neg B)$

### Elimination Rules in Isabelle

Tactic erule assumes that first rule premise is assumption to be eliminated: **apply** (erule <RuleName>):

### Example:

$$\begin{split} & \text{Rule: } [\![ ?P \land ?Q ; [\![ ?P; ?Q ]\!] \Rightarrow ?R ]\!] \Rightarrow ?R \\ & \text{Subgoal: } [\![ X ; A \land B ; Y ]\!] \Rightarrow Z \\ & \text{Unifier: } ?R == Z, ?P == A, ?Q == B \\ & \text{New subgoal: } [\![ X ; Y ]\!] \Rightarrow [\![ A ; B ]\!] \Rightarrow Z \\ & \text{Same as: } [\![ X ; Y ; A ; B ]\!] \Rightarrow Z \end{split}$$

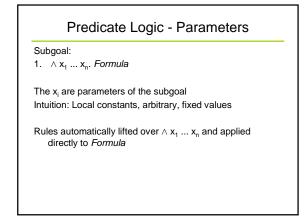


$$\Rightarrow$$
 vs.  $\rightarrow$ 

Theorems should be written as  $\label{eq:alpha} \left[ \begin{array}{c} A_1 \ ; \ \ldots \ ; \ A_n \ \right] \Rightarrow A$  Not as  $A_1 \wedge \ldots \wedge A_n \to A$ 

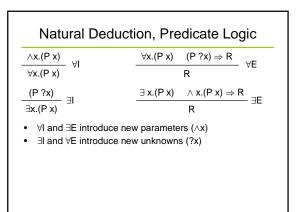
Exception: Induction variable must not occur in premises

 $\label{eq:example: constant} \begin{array}{l} \mbox{Example:} \\ \llbracket \mbox{ A; } B(x) \rrbracket \Rightarrow C(x), \mbox{ not good} \\ \mbox{Use instead: } A \Rightarrow B(x) \rightarrow C(x) \end{array}$ 



Scope of parameters: Whole subgoal Scope of HOL connectives: Never extend to meta-level I.e. ends with ; or  $\Rightarrow$ 

$$\begin{split} &\wedge x \, y. \ [\![ \ \forall y. \ P \ y \to Q \ z \ y; Q \ x \ y] \Rightarrow \exists x. \ Q \ x \ y \\ & \text{means} \\ &\wedge x \, y. [\![ \ \langle \forall y_1. \ P \ y_1 \to Q \ z \ y_1); Q \ x \ y] \Rightarrow \exists x_1. \ Q \ x_1 \ y \end{split}$$

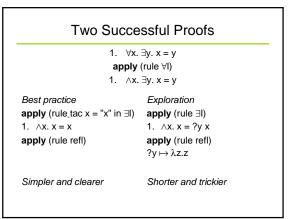


### Instantiating Rules

apply (rule\_tac x = t in <rule>) Acts as <rule>, but ?x in <rule> is instantiated to t before application

erule\_tac is similar

So: x is in <rule>, not in the goal



# Two Unsuccessful Proofs

1. ∃ y. ∀ x. x = y

 apply (rule tac x = ??? in ∃l)
 apply (rule ∃l)

 ???
 1. ∀x. x = ?y

∀x. x = ?y
 apply (rule ∀l)
 ∧ x. x = ?y
 apply (rule refl)
 ?y ↦ x yields ∧x'. x' = x
 ???

# Safe and Unsafe Rules

Safe: ∀I, ∃E Unsafe: ∀E, ∃I Create parameters first, unknowns later

## Exercises, Predicate Logic

**Exercise 3.** Prove or disprove the following formulas. If you prove the formulas, use Isabelle, as in exercise 2. For a disproof it is sufficient to show that the formulas are false in ordinary first-order logic.

- 1.  $\forall x. \forall y. R x y = \forall y. \forall x. R x y$
- 2.  $(\exists x. P x) \lor (\exists y. Q y) = \exists z. (P z) \lor (Q z)$
- $3. \quad \neg \; \forall x. \; \mathsf{P} \; x \Rightarrow \exists y. \neg (\mathsf{P} \; y)$
- 4.  $\exists x.(P x \rightarrow \forall y.P y)$

# **Renaming Parameters**

Careful with Isabelle-generated names

 ∀ x. ∃ y. x = y
 apply (rule ∀I)
 ∧x. ∃y. x = y
 apply (rule tac x = "x" in ∃I) What if the above used in context which already knows some x? Instead:
 apply (rename tac xxx)
 ∧xxx. ∃y. x = y
 apply (rule tac x = "xxx" in ∃I)

### Forward Proof

 $\sigma(B_i) == \sigma(A_1)$ 

 $1. \, [\![ \, \mathsf{B}_1 \, ; \, ... \, ; \, \mathsf{B}_m \, ]\!] \Rightarrow \mathsf{C}$ 

1.  $\sigma(\llbracket B_1; ...; B_n; A \rrbracket \Rightarrow C)$ 

 $A_1 \Rightarrow A$ 

"Forward" rule: Subgoal: Substitution: New subgoal:

### Command:

apply (frule <rule>) Like frule but deletes B<sub>i</sub>: apply (drule <rule>)