

Miscellaneous FOL Examples

December 12, 2021

Contents

1	Natural numbers	1
2	Examples for the manual “Introduction to Isabelle”	2
2.0.1	Some simple backward proofs	2
2.0.2	Demonstration of <i>fast</i>	3
2.0.3	Derivation of conjunction elimination rule	3
2.1	Derived rules involving definitions	3
3	Theory of the natural numbers: Peano’s axioms, primitive recursion	4
3.1	Proofs about the natural numbers	5
4	Theory of the natural numbers: Peano’s axioms, primitive recursion	6
5	Intuitionistic FOL: Examples from The Foundation of a Generic Theorem Prover	7
5.1	Examples with quantifiers	9
6	First-Order Logic: PROLOG examples	10
7	Intuitionistic First-Order Logic	12
7.1	Lemmas for the propositional double-negation translation . .	13
7.2	de Bruijn formulae	13
7.3	Intuitionistic FOL: propositional problems based on Pelletier.	14
7.4	11. Proved in each direction (incorrectly, says Pelletier!!) . . .	15
8	Examples with quantifiers	16
8.1	The converse is classical in the following implications	16
8.2	The following are not constructively valid!	16
8.3	Hard examples with quantifiers	17

9 First-Order Logic: propositional examples (intuitionistic version)	21
10 First-Order Logic: quantifier examples (intuitionistic version)	23
11 Classical Predicate Calculus Problems	25
11.0.1 If and only if	25
11.1 Pelletier's examples	25
11.2 Classical Logic: examples with quantifiers	27
11.3 Problems requiring quantifier duplication	27
11.4 Hard examples with quantifiers	28
11.5 Problems (mainly) involving equality or functions	32
12 First-Order Logic: propositional examples (classical version)	35
13 First-Order Logic: quantifier examples (classical version)	38
13.1 Negation Normal Form	40
13.1.1 de Morgan laws	40
13.1.2 Pushing in the existential quantifiers	40
13.1.3 Pushing in the universal quantifiers	40
14 First-Order Logic: the 'if' example	41

1 Natural numbers

```
theory Natural-Numbers
imports FOL
begin
```

Theory of the natural numbers: Peano's axioms, primitive recursion. (Modernized version of Larry Paulson's theory "Nat".)

```
typedecl nat
instance nat :: <term> ..
```

axiomatization

```
Zero :: <nat>    (<0>) and
Suc  :: <nat => nat> and
rec  :: <[nat, 'a, [nat, 'a] => 'a] => 'a>
where
  induct [case-names 0 Suc, induct type: nat]:
    <P(0) ==> (!x. P(x) ==> P(Suc(x))) ==> P(n)> and
  Suc-inject: <Suc(m) = Suc(n) ==> m = n> and
  Suc-neq-0: <Suc(m) = 0 ==> R> and
  rec-0: <rec(0, a, f) = a> and
```

```

rec-Suc:  $\langle \text{rec}(\text{Suc}(m), a, f) = f(m, \text{rec}(m, a, f)) \rangle$ 

lemma Suc-n-not-n:  $\langle \text{Suc}(k) \neq k \rangle$ 
proof (induct  $\langle k \rangle$ )
  show  $\langle \text{Suc}(0) \neq 0 \rangle$ 
  proof
    assume  $\langle \text{Suc}(0) = 0 \rangle$ 
    then show  $\langle \text{False} \rangle$  by (rule Suc-neq-0)
  qed
next
  fix n assume hyp:  $\langle \text{Suc}(n) \neq n \rangle$ 
  show  $\langle \text{Suc}(\text{Suc}(n)) \neq \text{Suc}(n) \rangle$ 
  proof
    assume  $\langle \text{Suc}(\text{Suc}(n)) = \text{Suc}(n) \rangle$ 
    then have  $\langle \text{Suc}(n) = n \rangle$  by (rule Suc-inject)
    with hyp show  $\langle \text{False} \rangle$  by contradiction
  qed
qed

definition add ::  $\langle \text{nat} \Rightarrow \text{nat} \Rightarrow \text{nat} \rangle$  (infixl  $\langle + \rangle$  60)
  where  $\langle m + n = \text{rec}(m, n, \lambda x y. \text{Suc}(y)) \rangle$ 

lemma add-0 [simp]:  $\langle 0 + n = n \rangle$ 
  unfolding add-def by (rule rec-0)

lemma add-Suc [simp]:  $\langle \text{Suc}(m) + n = \text{Suc}(m + n) \rangle$ 
  unfolding add-def by (rule rec-Suc)

lemma add-assoc:  $\langle (k + m) + n = k + (m + n) \rangle$ 
  by (induct  $\langle k \rangle$ ) simp-all

lemma add-0-right:  $\langle m + 0 = m \rangle$ 
  by (induct  $\langle m \rangle$ ) simp-all

lemma add-Suc-right:  $\langle m + \text{Suc}(n) = \text{Suc}(m + n) \rangle$ 
  by (induct  $\langle m \rangle$ ) simp-all

lemma
  assumes  $\langle !n. f(\text{Suc}(n)) = \text{Suc}(f(n)) \rangle$ 
  shows  $\langle f(i + j) = i + f(j) \rangle$ 
  using assms by (induct  $\langle i \rangle$ ) simp-all

end

```

2 Examples for the manual “Introduction to Isabelle”

```
theory Intro
imports FOL
begin
```

2.0.1 Some simple backward proofs

```
lemma mythm:  $\langle P \vee P \longrightarrow P \rangle$ 
apply (rule impI)
apply (rule disjE)
prefer 3 apply (assumption)
prefer 2 apply (assumption)
apply assumption
done
```

```
lemma  $\langle (P \wedge Q) \vee R \longrightarrow (P \vee R) \rangle$ 
apply (rule impI)
apply (erule disjE)
apply (drule conjunct1)
apply (rule disjI1)
apply (rule-tac [2] disjI2)
apply assumption+
done
```

Correct version, delaying use of *spec* until last.

```
lemma  $\langle (\forall x y. P(x,y)) \longrightarrow (\forall z w. P(w,z)) \rangle$ 
apply (rule impI)
apply (rule allI)
apply (rule allI)
apply (drule spec)
apply (drule spec)
apply assumption
done
```

2.0.2 Demonstration of *fast*

```
lemma  $\langle (\exists y. \forall x. J(y,x) \longleftrightarrow \neg J(x,x)) \longrightarrow \neg (\forall x. \exists y. \forall z. J(z,y) \longleftrightarrow \neg J(z,x)) \rangle$ 
apply fast
done
```

```
lemma  $\langle \forall x. P(x,f(x)) \longleftrightarrow (\exists y. (\forall z. P(z,y) \longrightarrow P(z,f(x))) \wedge P(x,y)) \rangle$ 
apply fast
done
```

2.0.3 Derivation of conjunction elimination rule

```
lemma
```

```

    assumes major:  $\langle P \wedge Q \rangle$ 
    and minor:  $\langle \llbracket P; Q \rrbracket \Longrightarrow R \rangle$ 
    shows  $\langle R \rangle$ 
  apply (rule minor)
  apply (rule major [THEN conjunct1])
  apply (rule major [THEN conjunct2])
done

```

2.1 Derived rules involving definitions

Derivation of negation introduction

```

lemma
  assumes  $\langle P \Longrightarrow False \rangle$ 
  shows  $\langle \neg P \rangle$ 
  apply (unfold not-def)
  apply (rule impI)
  apply (rule assms)
  apply assumption
done

```

```

lemma
  assumes major:  $\langle \neg P \rangle$ 
  and minor:  $\langle P \rangle$ 
  shows  $\langle R \rangle$ 
  apply (rule FalseE)
  apply (rule mp)
  apply (rule major [unfolded not-def])
  apply (rule minor)
done

```

Alternative proof of the result above

```

lemma
  assumes major:  $\langle \neg P \rangle$ 
  and minor:  $\langle P \rangle$ 
  shows  $\langle R \rangle$ 
  apply (rule minor [THEN major [unfolded not-def, THEN mp, THEN FalseE]])
done

```

end

3 Theory of the natural numbers: Peano's axioms, primitive recursion

```

theory Nat
  imports FOL
begin

```

```

typedecl nat
instance nat :: <term> ..

axiomatization
  Zero :: <nat> (<0>) and
  Suc :: <nat  $\Rightarrow$  nat> and
  rec :: <[nat, 'a, [nat, 'a]  $\Rightarrow$  'a]  $\Rightarrow$  'a>
where
  induct: <[[P(0);  $\bigwedge x. P(x) \Rightarrow P(\text{Suc}(x))$ ]]  $\Rightarrow$  P(n)> and
  Suc-inject: <Suc(m)=Suc(n)  $\Rightarrow$  m=n> and
  Suc-neq-0: <Suc(m)=0  $\Rightarrow$  R> and
  rec-0: <rec(0,a,f) = a> and
  rec-Suc: <rec(Suc(m), a, f) = f(m, rec(m,a,f))>

definition add :: <[nat, nat]  $\Rightarrow$  nat> (infixl <+> 60)
  where <m + n  $\equiv$  rec(m, n,  $\lambda x y. \text{Suc}(y)$ )>

```

3.1 Proofs about the natural numbers

```

lemma Suc-n-not-n: <Suc(k)  $\neq$  k>
apply (rule-tac n = <k> in induct)
apply (rule notI)
apply (erule Suc-neq-0)
apply (rule notI)
apply (erule notE)
apply (erule Suc-inject)
done

```

```

lemma <(k+m)+n = k+(m+n)>
apply (rule induct)
back
back
back
back
back
back
oops

```

```

lemma add-0 [simp]: <0+n = n>
apply (unfold add-def)
apply (rule rec-0)
done

```

```

lemma add-Suc [simp]: <Suc(m)+n = Suc(m+n)>
apply (unfold add-def)
apply (rule rec-Suc)
done

```

```

lemma add-assoc: <(k+m)+n = k+(m+n)>

```

```

apply (rule-tac  $n = \langle k \rangle$  in induct)
apply simp
apply simp
done

```

```

lemma add-0-right:  $\langle m+0 = m \rangle$ 
apply (rule-tac  $n = \langle m \rangle$  in induct)
apply simp
apply simp
done

```

```

lemma add-Suc-right:  $\langle m+Suc(n) = Suc(m+n) \rangle$ 
apply (rule-tac  $n = \langle m \rangle$  in induct)
apply simp-all
done

```

```

lemma
  assumes prem:  $\langle \bigwedge n. f(Suc(n)) = Suc(f(n)) \rangle$ 
  shows  $\langle f(i+j) = i+f(j) \rangle$ 
apply (rule-tac  $n = \langle i \rangle$  in induct)
apply simp
apply (simp add: prem)
done

```

```

end

```

4 Theory of the natural numbers: Peano's axioms, primitive recursion

```

theory Nat-Class
  imports FOL
begin

```

This is an abstract version of `Nat.thy`. Instead of axiomatizing a single type `nat`, it defines the class of all these types (up to isomorphism).

Note: The `rec` operator has been made *monomorphic*, because class axioms cannot contain more than one type variable.

```

class nat =
  fixes Zero ::  $\langle 'a \rangle \langle 0 \rangle$ 
  and Suc ::  $\langle 'a \Rightarrow 'a \rangle$ 
  and rec ::  $\langle 'a \Rightarrow 'a \Rightarrow ('a \Rightarrow 'a \Rightarrow 'a) \Rightarrow 'a \rangle$ 
  assumes induct:  $\langle P(0) \Longrightarrow (\bigwedge x. P(x) \Longrightarrow P(Suc(x))) \Longrightarrow P(n) \rangle$ 
  and Suc-inject:  $\langle Suc(m) = Suc(n) \Longrightarrow m = n \rangle$ 
  and Suc-neq-Zero:  $\langle Suc(m) = 0 \Longrightarrow R \rangle$ 
  and rec-Zero:  $\langle rec(0, a, f) = a \rangle$ 
  and rec-Suc:  $\langle rec(Suc(m), a, f) = f(m, rec(m, a, f)) \rangle$ 
begin

```

definition $add :: \langle 'a \Rightarrow 'a \Rightarrow 'a \rangle$ (**infixl** $\langle + \rangle$ 60)
where $\langle m + n = rec(m, n, \lambda x y. Suc(y)) \rangle$

lemma $Suc\text{-}n\text{-}not\text{-}n$: $\langle Suc(k) \neq (k::'a) \rangle$
apply (*rule-tac* $n = \langle k \rangle$ **in** *induct*)
apply (*rule notI*)
apply (*erule Suc-neq-Zero*)
apply (*rule notI*)
apply (*erule notE*)
apply (*erule Suc-inject*)
done

lemma $\langle (k + m) + n = k + (m + n) \rangle$
apply (*rule induct*)
back
back
back
back
back
oops

lemma $add\text{-}Zero$ [*simp*]: $\langle 0 + n = n \rangle$
apply (*unfold add-def*)
apply (*rule rec-Zero*)
done

lemma $add\text{-}Suc$ [*simp*]: $\langle Suc(m) + n = Suc(m + n) \rangle$
apply (*unfold add-def*)
apply (*rule rec-Suc*)
done

lemma $add\text{-}assoc$: $\langle (k + m) + n = k + (m + n) \rangle$
apply (*rule-tac* $n = \langle k \rangle$ **in** *induct*)
apply *simp*
apply *simp*
done

lemma $add\text{-}Zero\text{-}right$: $\langle m + 0 = m \rangle$
apply (*rule-tac* $n = \langle m \rangle$ **in** *induct*)
apply *simp*
apply *simp*
done

lemma $add\text{-}Suc\text{-}right$: $\langle m + Suc(n) = Suc(m + n) \rangle$
apply (*rule-tac* $n = \langle m \rangle$ **in** *induct*)
apply *simp-all*
done

lemma

```

    assumes prem:  $\langle \bigwedge n. f(\text{Suc}(n)) = \text{Suc}(f(n)) \rangle$ 
    shows  $\langle f(i + j) = i + f(j) \rangle$ 
    apply (rule-tac  $n = \langle i \rangle$  in induct)
    apply simp
    apply (simp add: prem)
    done

end

end

```

5 Intuitionistic FOL: Examples from The Foundation of a Generic Theorem Prover

```

theory Foundation
imports IFOL
begin

lemma  $\langle A \wedge B \longrightarrow (C \longrightarrow A \wedge C) \rangle$ 
  apply (rule impI)
  apply (rule impI)
  apply (rule conjI)
  prefer 2 apply assumption
  apply (rule conjunct1)
  apply assumption
  done

```

A form of conj-elimination

```

lemma
  assumes  $\langle A \wedge B \rangle$ 
  and  $\langle A \Longrightarrow B \Longrightarrow C \rangle$ 
  shows  $\langle C \rangle$ 
  apply (rule assms)
  apply (rule conjunct1)
  apply (rule assms)
  apply (rule conjunct2)
  apply (rule assms)
  done

```

```

lemma
  assumes  $\langle \bigwedge A. \neg \neg A \Longrightarrow A \rangle$ 
  shows  $\langle B \vee \neg B \rangle$ 
  apply (rule assms)
  apply (rule notI)
  apply (rule-tac  $P = \langle \neg B \rangle$  in notE)
  apply (rule-tac [2] notI)
  apply (rule-tac [2]  $P = \langle B \vee \neg B \rangle$  in notE)
  prefer 2 apply assumption

```

```

apply (rule-tac [2] disjI1)
prefer 2 apply assumption
apply (rule notI)
apply (rule-tac  $P = \langle B \vee \neg B \rangle$  in notE)
apply assumption
apply (rule disjI2)
apply assumption
done

```

```

lemma
  assumes  $\langle \bigwedge A. \neg \neg A \implies A \rangle$ 
  shows  $\langle B \vee \neg B \rangle$ 
apply (rule assms)
apply (rule notI)
apply (rule notE)
apply (rule-tac [2] notI)
apply (erule-tac [2] notE)
apply (erule-tac [2] disjI1)
apply (rule notI)
apply (erule notE)
apply (erule disjI2)
done

```

```

lemma
  assumes  $\langle A \vee \neg A \rangle$ 
  and  $\langle \neg \neg A \rangle$ 
  shows  $\langle A \rangle$ 
apply (rule disjE)
apply (rule assms)
apply assumption
apply (rule FalseE)
apply (rule-tac  $P = \langle \neg A \rangle$  in notE)
apply (rule assms)
apply assumption
done

```

5.1 Examples with quantifiers

```

lemma
  assumes  $\langle \forall z. G(z) \rangle$ 
  shows  $\langle \forall z. G(z) \vee H(z) \rangle$ 
apply (rule allI)
apply (rule disjI1)
apply (rule assms [THEN spec])
done

```

```

lemma  $\langle \forall x. \exists y. x = y \rangle$ 
apply (rule allI)

```

```

apply (rule exI)
apply (rule refl)
done

```

```

lemma  $\langle \exists y. \forall x. x = y \rangle$ 
apply (rule exI)
apply (rule allI)
apply (rule refl)?
oops

```

Parallel lifting example.

```

lemma  $\langle \exists u. \forall x. \exists v. \forall y. \exists w. P(u, x, v, y, w) \rangle$ 
apply (rule exI allI)
apply (rule exI allI)
apply (rule exI allI)
apply (rule exI allI)
apply (rule exI allI)
oops

```

```

lemma
  assumes  $\langle (\exists z. F(z)) \wedge B \rangle$ 
  shows  $\langle \exists z. F(z) \wedge B \rangle$ 
apply (rule conjE)
apply (rule assms)
apply (rule exE)
apply assumption
apply (rule exI)
apply (rule conjI)
apply assumption
apply assumption
done

```

A bigger demonstration of quantifiers – not in the paper.

```

lemma  $\langle (\exists y. \forall x. Q(x, y)) \longrightarrow (\forall x. \exists y. Q(x, y)) \rangle$ 
apply (rule impI)
apply (rule allI)
apply (rule exE, assumption)
apply (rule exI)
apply (rule allE, assumption)
apply assumption
done

```

end

6 First-Order Logic: PROLOG examples

```

theory Prolog
imports FOL

```

```

begin

typedecl 'a list
instance list :: (⟨term⟩) ⟨term⟩ ..

axiomatization
  Nil    :: ⟨'a list⟩ and
  Cons   :: ⟨['a, 'a list] => 'a list⟩ (infixr ⟨:⟩ 60) and
  app    :: ⟨['a list, 'a list, 'a list] => o⟩ and
  rev    :: ⟨['a list, 'a list] => o⟩
where
  appNil: ⟨app(Nil,ys,ys)⟩ and
  appCons: ⟨app(xs,ys,zs) ==> app(x:xs, ys, x:zs)⟩ and
  revNil: ⟨rev(Nil,Nil)⟩ and
  revCons: ⟨[| rev(xs,ys); app(ys, x:Nil, zs) |] ==> rev(x:xs, zs)⟩

schematic-goal ⟨app(a:b:c:Nil, d:e:Nil, ?x)⟩
apply (rule appNil appCons)
apply (rule appNil appCons)
apply (rule appNil appCons)
apply (rule appNil appCons)
done

schematic-goal ⟨app(?x, c:d:Nil, a:b:c:d:Nil)⟩
apply (rule appNil appCons)+
done

schematic-goal ⟨app(?x, ?y, a:b:c:d:Nil)⟩
apply (rule appNil appCons)+
back
back
back
back
done

lemmas rules = appNil appCons revNil revCons

schematic-goal ⟨rev(a:b:c:d:Nil, ?x)⟩
apply (rule rules)+
done

schematic-goal ⟨rev(a:b:c:d:e:f:g:h:i:j:k:l:m:n:Nil, ?w)⟩
apply (rule rules)+
done

schematic-goal ⟨rev(?x, a:b:c:Nil)⟩

```

```

apply (rule rules)+ — does not solve it directly!
back
back
done

```

```

ML <
  fun prolog-tac ctxt =
    DEPTH-FIRST (has-fewer-prems 1) (resolve-tac ctxt @{thms rules} 1)
  >

```

```

schematic-goal <rev(?x, a:b:c:Nil)>
apply (tactic <prolog-tac context>)
done

```

```

schematic-goal <rev(a:?x:c:?y:Nil, d:?z:b:?u)>
apply (tactic <prolog-tac context>)
done

```

```

schematic-goal <rev(a:b:c:d:e:f:g:h:i:j:k:l:m:n:o:p:Nil, ?w)>
apply (tactic <
  DEPTH-SOLVE (resolve-tac context ([@{thm refl}, @{thm conjI}] @ @{thms
rules}) 1)>)
done

```

```

schematic-goal <a:b:c:d:e:f:g:h:i:j:k:l:m:n:o:p:Nil = ?x ∧ app(?x, ?x, ?y) ∧ rev(?y, ?w)>
apply (tactic <
  DEPTH-SOLVE (resolve-tac context ([@{thm refl}, @{thm conjI}] @ @{thms
rules}) 1)>)
done

```

```

end

```

7 Intuitionistic First-Order Logic

```

theory Intuitionistic
imports IFOL
begin

```

Metatheorem (for *propositional* formulae): P is classically provable iff $\neg\neg P$ is intuitionistically provable. Therefore $\neg P$ is classically provable iff it is intuitionistically provable.

Proof: Let Q be the conjunction of the propositions $A \vee \neg A$, one for each atom A in P . Now $\neg\neg Q$ is intuitionistically provable because $\neg\neg(A \vee \neg A)$ is and because double-negation distributes over conjunction. If P is provable classically, then clearly $Q \rightarrow P$ is provable intuitionistically, so $\neg\neg(Q \rightarrow P)$

is also provable intuitionistically. The latter is intuitionistically equivalent to $\neg\neg Q \rightarrow \neg\neg P$, hence to $\neg\neg P$, since $\neg\neg Q$ is intuitionistically provable. Finally, if P is a negation then $\neg\neg P$ is intuitionistically equivalent to P . [Andy Pitts]

lemma $\langle \neg\neg (P \wedge Q) \longleftrightarrow \neg\neg P \wedge \neg\neg Q \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

lemma $\langle \neg\neg ((\neg P \rightarrow Q) \rightarrow (\neg P \rightarrow \neg Q) \rightarrow P) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

Double-negation does NOT distribute over disjunction.

lemma $\langle \neg\neg (P \rightarrow Q) \longleftrightarrow (\neg\neg P \rightarrow \neg\neg Q) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

lemma $\langle \neg\neg\neg P \longleftrightarrow \neg P \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

lemma $\langle \neg\neg ((P \rightarrow Q \vee R) \rightarrow (P \rightarrow Q) \vee (P \rightarrow R)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

lemma $\langle (P \longleftrightarrow Q) \longleftrightarrow (Q \longleftrightarrow P) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

lemma $\langle ((P \rightarrow (Q \vee (Q \rightarrow R))) \rightarrow R) \rightarrow R \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

lemma
 $\langle (((G \rightarrow A) \rightarrow J) \rightarrow D \rightarrow E) \rightarrow (((H \rightarrow B) \rightarrow I) \rightarrow C \rightarrow J)$
 $\rightarrow (A \rightarrow H) \rightarrow F \rightarrow G \rightarrow (((C \rightarrow B) \rightarrow I) \rightarrow D) \rightarrow (A \rightarrow C)$
 $\rightarrow (((F \rightarrow A) \rightarrow B) \rightarrow I) \rightarrow E \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

7.1 Lemmas for the propositional double-negation translation

lemma $\langle P \rightarrow \neg\neg P \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

lemma $\langle \neg\neg (\neg\neg P \rightarrow P) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

lemma $\langle \neg\neg P \wedge \neg\neg (P \rightarrow Q) \rightarrow \neg\neg Q \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)

The following are classically but not constructively valid. The attempt to prove them terminates quickly!

lemma $\langle (P \rightarrow Q) \rightarrow P \rightarrow P \rangle$
apply (*tactic* $\langle \text{IntPr.fast-tac context } 1 \rangle$)?

apply (*rule asm-rl*) — Checks that subgoals remain: proof failed.
oops

lemma $\langle (P \wedge Q \longrightarrow R) \longrightarrow (P \longrightarrow R) \vee (Q \longrightarrow R) \rangle$
apply (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)?
apply (*rule asm-rl*) — Checks that subgoals remain: proof failed.
oops

7.2 de Bruijn formulae

de Bruijn formula with three predicates

lemma
 $\langle ((P \longleftrightarrow Q) \longrightarrow P \wedge Q \wedge R) \wedge$
 $((Q \longleftrightarrow R) \longrightarrow P \wedge Q \wedge R) \wedge$
 $((R \longleftrightarrow P) \longrightarrow P \wedge Q \wedge R) \longrightarrow P \wedge Q \wedge R \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

de Bruijn formula with five predicates

lemma
 $\langle ((P \longleftrightarrow Q) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \wedge$
 $((Q \longleftrightarrow R) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \wedge$
 $((R \longleftrightarrow S) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \wedge$
 $((S \longleftrightarrow T) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \wedge$
 $((T \longleftrightarrow P) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

Problems from of Sahlin, Franzen and Haridi, An Intuitionistic Predicate Logic Theorem Prover. J. Logic and Comp. 2 (5), October 1992, 619-656.

Problem 1.1

lemma
 $\langle (\forall x. \exists y. \forall z. p(x) \wedge q(y) \wedge r(z)) \longleftrightarrow$
 $(\forall z. \exists y. \forall x. p(x) \wedge q(y) \wedge r(z)) \rangle$
by (*tactic* $\langle \text{IntPr.best-dup-tac } \mathbf{context} \ 1 \rangle$) — SLOW

Problem 3.1

lemma $\langle \neg (\exists x. \forall y. \text{mem}(y, x) \longleftrightarrow \neg \text{mem}(x, x)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

Problem 4.1: hopeless!

lemma
 $\langle (\forall x. p(x) \longrightarrow p(h(x)) \vee p(g(x))) \wedge (\exists x. p(x)) \wedge (\forall x. \neg p(h(x)))$
 $\longrightarrow (\exists x. p(g(g(g(g(x)))))) \rangle$
oops

7.3 Intuitionistic FOL: propositional problems based on Pelletier.

$\neg\neg 1$

lemma $\langle \neg \neg ((P \longrightarrow Q) \longleftrightarrow (\neg Q \longrightarrow \neg P)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg 2$

lemma $\langle \neg \neg (\neg \neg P \longleftrightarrow P) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

3

lemma $\langle \neg (P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg 4$

lemma $\langle \neg \neg ((\neg P \longrightarrow Q) \longleftrightarrow (\neg Q \longrightarrow P)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg 5$

lemma $\langle \neg \neg ((P \vee Q \longrightarrow P \vee R) \longrightarrow P \vee (Q \longrightarrow R)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg 6$

lemma $\langle \neg \neg (P \vee \neg P) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg 7$

lemma $\langle \neg \neg (P \vee \neg \neg P) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg 8$. Peirce's law

lemma $\langle \neg \neg (((P \longrightarrow Q) \longrightarrow P) \longrightarrow P) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

9

lemma $\langle ((P \vee Q) \wedge (\neg P \vee Q) \wedge (P \vee \neg Q)) \longrightarrow \neg (\neg P \vee \neg Q) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

10

lemma $\langle (Q \longrightarrow R) \longrightarrow (R \longrightarrow P \wedge Q) \longrightarrow (P \longrightarrow (Q \vee R)) \longrightarrow (P \longleftrightarrow Q) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

7.4 11. Proved in each direction (incorrectly, says Pelletier!!)

lemma $\langle P \longleftrightarrow P \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg 12$. Dijkstra's law

lemma $\langle \neg \neg (((P \longleftrightarrow Q) \longleftrightarrow R) \longleftrightarrow (P \longleftrightarrow (Q \longleftrightarrow R))) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

lemma $\langle ((P \longleftrightarrow Q) \longleftrightarrow R) \longrightarrow \neg \neg (P \longleftrightarrow (Q \longleftrightarrow R)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

13. Distributive law

lemma $\langle P \vee (Q \wedge R) \longleftrightarrow (P \vee Q) \wedge (P \vee R) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

$\neg \neg$ 14

lemma $\langle \neg \neg ((P \longleftrightarrow Q) \longleftrightarrow ((Q \vee \neg P) \wedge (\neg Q \vee P))) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

$\neg \neg$ 15

lemma $\langle \neg \neg ((P \longrightarrow Q) \longleftrightarrow (\neg P \vee Q)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

$\neg \neg$ 16

lemma $\langle \neg \neg ((P \longrightarrow Q) \vee (Q \longrightarrow P)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

$\neg \neg$ 17

lemma $\langle \neg \neg (((P \wedge (Q \longrightarrow R)) \longrightarrow S) \longleftrightarrow ((\neg P \vee Q \vee S) \wedge (\neg P \vee \neg R \vee S))) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

Dijkstra's "Golden Rule"

lemma $\langle (P \wedge Q) \longleftrightarrow P \longleftrightarrow Q \longleftrightarrow (P \vee Q) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

8 Examples with quantifiers

8.1 The converse is classical in the following implications ...

lemma $\langle (\exists x. P(x) \longrightarrow Q) \longrightarrow (\forall x. P(x)) \longrightarrow Q \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

lemma $\langle ((\forall x. P(x)) \longrightarrow Q) \longrightarrow \neg (\forall x. P(x) \wedge \neg Q) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

lemma $\langle ((\forall x. \neg P(x)) \longrightarrow Q) \longrightarrow \neg (\forall x. \neg (P(x) \vee Q)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

lemma $\langle (\forall x. P(x)) \vee Q \longrightarrow (\forall x. P(x) \vee Q) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

lemma $\langle (\exists x. P \longrightarrow Q(x)) \longrightarrow (P \longrightarrow (\exists x. Q(x))) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac context 1} \rangle$)

8.2 The following are not constructively valid!

The attempt to prove them terminates quickly!

```
lemma <((∀ x. P(x)) → Q) → (∃ x. P(x) → Q)>
  apply (tactic <IntPr.fast-tac context 1>)?
  apply (rule asm-rl) — Checks that subgoals remain: proof failed.
oops
```

```
lemma <(P → (∃ x. Q(x))) → (∃ x. P → Q(x))>
  apply (tactic <IntPr.fast-tac context 1>)?
  apply (rule asm-rl) — Checks that subgoals remain: proof failed.
oops
```

```
lemma <(∀ x. P(x) ∨ Q) → ((∀ x. P(x)) ∨ Q)>
  apply (tactic <IntPr.fast-tac context 1>)?
  apply (rule asm-rl) — Checks that subgoals remain: proof failed.
oops
```

```
lemma <(∀ x. ¬ ¬ P(x)) → ¬ ¬ (∀ x. P(x))>
  apply (tactic <IntPr.fast-tac context 1>)?
  apply (rule asm-rl) — Checks that subgoals remain: proof failed.
oops
```

Classically but not intuitionistically valid. Proved by a bug in 1986!

```
lemma <∃ x. Q(x) → (∀ x. Q(x))>
  apply (tactic <IntPr.fast-tac context 1>)?
  apply (rule asm-rl) — Checks that subgoals remain: proof failed.
oops
```

8.3 Hard examples with quantifiers

The ones that have not been proved are not known to be valid! Some will require quantifier duplication – not currently available.

¬¬18

```
lemma <¬ ¬ (∃ y. ∀ x. P(y) → P(x))>
  oops — NOT PROVED
```

¬¬19

```
lemma <¬ ¬ (∃ x. ∀ y z. (P(y) → Q(z)) → (P(x) → Q(x)))>
  oops — NOT PROVED
```

20

```
lemma
  <(∀ x y. ∃ z. ∀ w. (P(x) ∧ Q(y) → R(z) ∧ S(w)))
    → (∃ x y. P(x) ∧ Q(y)) → (∃ z. R(z))>
  by (tactic <IntPr.fast-tac context 1>)
```

21

lemma $\langle (\exists x. P \longrightarrow Q(x)) \wedge (\exists x. Q(x) \longrightarrow P) \longrightarrow \neg \neg (\exists x. P \longleftrightarrow Q(x)) \rangle$
oops — NOT PROVED; needs quantifier duplication

22

lemma $\langle (\forall x. P \longleftrightarrow Q(x)) \longrightarrow (P \longleftrightarrow (\forall x. Q(x))) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg$ 23

lemma $\langle \neg \neg ((\forall x. P \vee Q(x)) \longleftrightarrow (P \vee (\forall x. Q(x)))) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

24

lemma
 $\langle \neg (\exists x. S(x) \wedge Q(x)) \wedge (\forall x. P(x) \longrightarrow Q(x) \vee R(x)) \wedge$
 $(\neg (\exists x. P(x)) \longrightarrow (\exists x. Q(x))) \wedge (\forall x. Q(x) \vee R(x) \longrightarrow S(x))$
 $\longrightarrow \neg \neg (\exists x. P(x) \wedge R(x)) \rangle$

Not clear why *fast-tac*, *best-tac*, *ASTAR* and *ITER-DEEPEN* all take forever.

apply (*tactic* $\langle \text{IntPr.safe-tac } \textbf{context} \rangle$)
apply (*erule* *impE*)
apply (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)
apply (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)
done

25

lemma
 $\langle (\exists x. P(x)) \wedge$
 $(\forall x. L(x) \longrightarrow \neg (M(x) \wedge R(x))) \wedge$
 $(\forall x. P(x) \longrightarrow (M(x) \wedge L(x))) \wedge$
 $((\forall x. P(x) \longrightarrow Q(x)) \vee (\exists x. P(x) \wedge R(x)))$
 $\longrightarrow (\exists x. Q(x) \wedge P(x)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

$\neg\neg$ 26

lemma
 $\langle (\neg \neg (\exists x. p(x)) \longleftrightarrow \neg \neg (\exists x. q(x))) \wedge$
 $(\forall x. \forall y. p(x) \wedge q(y) \longrightarrow (r(x) \longleftrightarrow s(y)))$
 $\longrightarrow ((\forall x. p(x) \longrightarrow r(x)) \longleftrightarrow (\forall x. q(x) \longrightarrow s(x))) \rangle$
oops — NOT PROVED

27

lemma
 $\langle (\exists x. P(x) \wedge \neg Q(x)) \wedge$
 $(\forall x. P(x) \longrightarrow R(x)) \wedge$
 $(\forall x. M(x) \wedge L(x) \longrightarrow P(x)) \wedge$

$((\exists x. R(x) \wedge \neg Q(x)) \longrightarrow (\forall x. L(x) \longrightarrow \neg R(x)))$
 $\longrightarrow (\forall x. M(x) \longrightarrow \neg L(x))\rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

$\neg\neg$ 28. AMENDED

lemma

$\langle (\forall x. P(x) \longrightarrow (\forall x. Q(x))) \wedge$
 $(\neg\neg (\forall x. Q(x) \vee R(x)) \longrightarrow (\exists x. Q(x) \wedge S(x))) \wedge$
 $(\neg\neg (\exists x. S(x)) \longrightarrow (\forall x. L(x) \longrightarrow M(x)))$
 $\longrightarrow (\forall x. P(x) \wedge L(x) \longrightarrow M(x))\rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

29. Essentially the same as Principia Mathematica *11.71

lemma

$\langle (\exists x. P(x)) \wedge (\exists y. Q(y))$
 $\longrightarrow ((\forall x. P(x) \longrightarrow R(x)) \wedge (\forall y. Q(y) \longrightarrow S(y)) \longleftrightarrow$
 $(\forall x y. P(x) \wedge Q(y) \longrightarrow R(x) \wedge S(y)))\rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

$\neg\neg$ 30

lemma

$\langle (\forall x. (P(x) \vee Q(x)) \longrightarrow \neg R(x)) \wedge$
 $(\forall x. (Q(x) \longrightarrow \neg S(x)) \longrightarrow P(x) \wedge R(x))$
 $\longrightarrow (\forall x. \neg\neg S(x))\rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

31

lemma

$\langle \neg (\exists x. P(x) \wedge (Q(x) \vee R(x))) \wedge$
 $(\exists x. L(x) \wedge P(x)) \wedge$
 $(\forall x. \neg R(x) \longrightarrow M(x))$
 $\longrightarrow (\exists x. L(x) \wedge M(x))\rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

32

lemma

$\langle (\forall x. P(x) \wedge (Q(x) \vee R(x)) \longrightarrow S(x)) \wedge$
 $(\forall x. S(x) \wedge R(x) \longrightarrow L(x)) \wedge$
 $(\forall x. M(x) \longrightarrow R(x))$
 $\longrightarrow (\forall x. P(x) \wedge M(x) \longrightarrow L(x))\rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \mathbf{context} \ 1 \rangle$)

$\neg\neg$ 33

lemma

$\langle (\forall x. \neg\neg (P(a) \wedge (P(x) \longrightarrow P(b)) \longrightarrow P(c))) \longleftrightarrow$
 $(\forall x. \neg\neg ((\neg P(a) \vee P(x) \vee P(c)) \wedge (\neg P(a) \vee \neg P(b) \vee P(c))))\rangle$
apply (*tactic* $\langle \text{IntPr.best-tac } \mathbf{context} \ 1 \rangle$)
done

36

lemma

$\langle (\forall x. \exists y. J(x,y)) \wedge$
 $(\forall x. \exists y. G(x,y)) \wedge$
 $(\forall x y. J(x,y) \vee G(x,y) \longrightarrow (\forall z. J(y,z) \vee G(y,z) \longrightarrow H(x,z)))$
 $\longrightarrow (\forall x. \exists y. H(x,y)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

37

lemma

$\langle (\forall z. \exists w. \forall x. \exists y.$
 $\neg \neg (P(x,z) \longrightarrow P(y,w)) \wedge P(y,z) \wedge (P(y,w) \longrightarrow (\exists u. Q(u,w)))) \wedge$
 $(\forall x z. \neg P(x,z) \longrightarrow (\exists y. Q(y,z))) \wedge$
 $(\neg \neg (\exists x y. Q(x,y)) \longrightarrow (\forall x. R(x,x)))$
 $\longrightarrow \neg \neg (\forall x. \exists y. R(x,y)) \rangle$
oops — NOT PROVED

39

lemma $\langle \neg (\exists x. \forall y. F(y,x) \longleftrightarrow \neg F(y,y)) \rangle$

by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

40. AMENDED

lemma

$\langle (\exists y. \forall x. F(x,y) \longleftrightarrow F(x,x)) \longrightarrow$
 $\neg (\forall x. \exists y. \forall z. F(z,y) \longleftrightarrow \neg F(z,x)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

44

lemma

$\langle (\forall x. f(x) \longrightarrow$
 $(\exists y. g(y) \wedge h(x,y) \wedge (\exists y. g(y) \wedge \neg h(x,y)))) \wedge$
 $(\exists x. j(x) \wedge (\forall y. g(y) \longrightarrow h(x,y)))$
 $\longrightarrow (\exists x. j(x) \wedge \neg f(x)) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

48

lemma $\langle (a = b \vee c = d) \wedge (a = c \vee b = d) \longrightarrow a = d \vee b = c \rangle$

by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

51

lemma

$\langle (\exists z w. \forall x y. P(x,y) \longleftrightarrow (x = z \wedge y = w)) \longrightarrow$
 $(\exists z. \forall x. \exists w. (\forall y. P(x,y) \longleftrightarrow y = w) \longleftrightarrow x = z) \rangle$
by (*tactic* $\langle \text{IntPr.fast-tac } \textbf{context } 1 \rangle$)

52

Almost the same as 51.

lemma

$\langle (\exists z w. \forall x y. P(x,y) \longleftrightarrow (x = z \wedge y = w)) \longrightarrow$
 $(\exists w. \forall y. \exists z. (\forall x. P(x,y) \longleftrightarrow x = z) \longleftrightarrow y = w) \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

56

lemma $\langle (\forall x. (\exists y. P(y) \wedge x = f(y)) \longrightarrow P(x)) \longleftrightarrow (\forall x. P(x) \longrightarrow P(f(x))) \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

57

lemma

$\langle P(f(a,b), f(b,c)) \wedge P(f(b,c), f(a,c)) \wedge$
 $(\forall x y z. P(x,y) \wedge P(y,z) \longrightarrow P(x,z)) \longrightarrow P(f(a,b), f(a,c)) \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

60

lemma $\langle \forall x. P(x, f(x)) \longleftrightarrow (\exists y. (\forall z. P(z, y) \longrightarrow P(z, f(x))) \wedge P(x, y)) \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

end

9 First-Order Logic: propositional examples (intuitionistic version)

theory *Propositional-Int*

imports *IFOL*

begin

commutative laws of $\wedge$ and $\vee$

lemma $\langle P \wedge Q \longrightarrow Q \wedge P \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

lemma $\langle P \vee Q \longrightarrow Q \vee P \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

associative laws of $\wedge$ and $\vee$

lemma $\langle (P \wedge Q) \wedge R \longrightarrow P \wedge (Q \wedge R) \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

lemma $\langle (P \vee Q) \vee R \longrightarrow P \vee (Q \vee R) \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

distributive laws of $\wedge$ and $\vee$

lemma $\langle (P \wedge Q) \vee R \longrightarrow (P \vee R) \wedge (Q \vee R) \rangle$
by (*tactic* *IntPr.fast-tac context 1*)

lemma $\langle (P \vee R) \wedge (Q \vee R) \longrightarrow (P \wedge Q) \vee R \rangle$

by (*tactic IntPr.fast-tac context 1*)

lemma $\langle (P \vee Q) \wedge R \longrightarrow (P \wedge R) \vee (Q \wedge R) \rangle$
by (*tactic IntPr.fast-tac context 1*)

lemma $\langle (P \wedge R) \vee (Q \wedge R) \longrightarrow (P \vee Q) \wedge R \rangle$
by (*tactic IntPr.fast-tac context 1*)

Laws involving implication

lemma $\langle (P \longrightarrow R) \wedge (Q \longrightarrow R) \longleftrightarrow (P \vee Q \longrightarrow R) \rangle$
by (*tactic IntPr.fast-tac context 1*)

lemma $\langle (P \wedge Q \longrightarrow R) \longleftrightarrow (P \longrightarrow (Q \longrightarrow R)) \rangle$
by (*tactic IntPr.fast-tac context 1*)

lemma $\langle ((P \longrightarrow R) \longrightarrow R) \longrightarrow ((Q \longrightarrow R) \longrightarrow R) \longrightarrow (P \wedge Q \longrightarrow R) \longrightarrow R \rangle$
by (*tactic IntPr.fast-tac context 1*)

lemma $\langle \neg (P \longrightarrow R) \longrightarrow \neg (Q \longrightarrow R) \longrightarrow \neg (P \wedge Q \longrightarrow R) \rangle$
by (*tactic IntPr.fast-tac context 1*)

lemma $\langle (P \longrightarrow Q \wedge R) \longleftrightarrow (P \longrightarrow Q) \wedge (P \longrightarrow R) \rangle$
by (*tactic IntPr.fast-tac context 1*)

Propositions-as-types

lemma $\langle P \longrightarrow (Q \longrightarrow P) \rangle$
by (*tactic IntPr.fast-tac context 1*)

— The combinator S

lemma $\langle (P \longrightarrow Q \longrightarrow R) \longrightarrow (P \longrightarrow Q) \longrightarrow (P \longrightarrow R) \rangle$
by (*tactic IntPr.fast-tac context 1*)

— Converse is classical

lemma $\langle (P \longrightarrow Q) \vee (P \longrightarrow R) \longrightarrow (P \longrightarrow Q \vee R) \rangle$
by (*tactic IntPr.fast-tac context 1*)

lemma $\langle (P \longrightarrow Q) \longrightarrow (\neg Q \longrightarrow \neg P) \rangle$
by (*tactic IntPr.fast-tac context 1*)

Schwichtenberg's examples (via T. Nipkow)

lemma *stab-imp*: $\langle (((Q \longrightarrow R) \longrightarrow R) \longrightarrow Q) \longrightarrow (((P \longrightarrow Q) \longrightarrow R) \longrightarrow R) \longrightarrow P \longrightarrow Q \rangle$
by (*tactic IntPr.fast-tac context 1*)

lemma *stab-to-peirce*:

$\langle (((P \longrightarrow R) \longrightarrow R) \longrightarrow P) \longrightarrow (((Q \longrightarrow R) \longrightarrow R) \longrightarrow Q) \longrightarrow ((P \longrightarrow Q) \longrightarrow P) \longrightarrow P \rangle$
by (*tactic IntPr.fast-tac context 1*)

```

lemma peirce-imp1:
  ⟨(((Q → R) → Q) → Q)
    → (((P → Q) → R) → P → Q) → P → Q⟩
  by (tactic IntPr.fast-tac context 1)

lemma peirce-imp2: ⟨(((P → R) → P) → P) → ((P → Q → R) → P)
  → P⟩
  by (tactic IntPr.fast-tac context 1)

lemma mints: ⟨(((P → Q) → P) → P) → Q) → Q⟩
  by (tactic IntPr.fast-tac context 1)

lemma mints-solovev: ⟨(P → (Q → R) → Q) → ((P → Q) → R) → R⟩
  by (tactic IntPr.fast-tac context 1)

lemma tatsuta:
  ⟨(((P7 → P1) → P10) → P4 → P5)
    → (((P8 → P2) → P9) → P3 → P10)
    → (P1 → P8) → P6 → P7
    → (((P3 → P2) → P9) → P4)
    → (P1 → P3) → (((P6 → P1) → P2) → P9) → P5⟩
  by (tactic IntPr.fast-tac context 1)

lemma tatsuta1:
  ⟨(((P8 → P2) → P9) → P3 → P10)
    → (((P3 → P2) → P9) → P4)
    → (((P6 → P1) → P2) → P9)
    → (((P7 → P1) → P10) → P4 → P5)
    → (P1 → P3) → (P1 → P8) → P6 → P7 → P5⟩
  by (tactic IntPr.fast-tac context 1)

end

```

10 First-Order Logic: quantifier examples (intuitionistic version)

```

theory Quantifiers-Int
imports IFOL
begin

```

```

lemma ⟨(∀ x y. P(x,y)) → (∀ y x. P(x,y))⟩
  by (tactic IntPr.fast-tac context 1)

```

```

lemma ⟨(∃ x y. P(x,y)) → (∃ y x. P(x,y))⟩
  by (tactic IntPr.fast-tac context 1)

```

— Converse is false

lemma $\langle (\forall x. P(x)) \vee (\forall x. Q(x)) \longrightarrow (\forall x. P(x) \vee Q(x)) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

lemma $\langle (\forall x. P \longrightarrow Q(x)) \longleftrightarrow (P \longrightarrow (\forall x. Q(x))) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

lemma $\langle (\forall x. P(x) \longrightarrow Q) \longleftrightarrow ((\exists x. P(x)) \longrightarrow Q) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

Some harder ones

lemma $\langle (\exists x. P(x) \vee Q(x)) \longleftrightarrow (\exists x. P(x)) \vee (\exists x. Q(x)) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

— Converse is false

lemma $\langle (\exists x. P(x) \wedge Q(x)) \longrightarrow (\exists x. P(x)) \wedge (\exists x. Q(x)) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

Basic test of quantifier reasoning

lemma $\langle (\exists y. \forall x. Q(x,y)) \longrightarrow (\forall x. \exists y. Q(x,y)) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

lemma $\langle (\forall x. Q(x)) \longrightarrow (\exists x. Q(x)) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

The following should fail, as they are false!

lemma $\langle (\forall x. \exists y. Q(x,y)) \longrightarrow (\exists y. \forall x. Q(x,y)) \rangle$
apply (*tactic* *IntPr.fast-tac* **context** 1)?
oops

lemma $\langle (\exists x. Q(x)) \longrightarrow (\forall x. Q(x)) \rangle$
apply (*tactic* *IntPr.fast-tac* **context** 1)?
oops

schematic-goal $\langle P(?a) \longrightarrow (\forall x. P(x)) \rangle$
apply (*tactic* *IntPr.fast-tac* **context** 1)?
oops

schematic-goal $\langle (P(?a) \longrightarrow (\forall x. Q(x))) \longrightarrow (\forall x. P(x) \longrightarrow Q(x)) \rangle$
apply (*tactic* *IntPr.fast-tac* **context** 1)?
oops

Back to things that are provable ...

lemma $\langle (\forall x. P(x) \longrightarrow Q(x)) \wedge (\exists x. P(x)) \longrightarrow (\exists x. Q(x)) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

— An example of why exI should be delayed as long as possible

lemma $\langle (P \longrightarrow (\exists x. Q(x))) \wedge P \longrightarrow (\exists x. Q(x)) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

schematic-goal $\langle (\forall x. P(x) \longrightarrow Q(f(x))) \wedge (\forall x. Q(x) \longrightarrow R(g(x))) \wedge P(d) \longrightarrow R(?a) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

lemma $\langle (\forall x. Q(x)) \longrightarrow (\exists x. Q(x)) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

Some slow ones

lemma $\langle (\forall x y. P(x) \longrightarrow Q(y)) \longleftrightarrow ((\exists x. P(x)) \longrightarrow (\forall y. Q(y))) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

lemma $\langle (\exists x y. P(x) \wedge Q(x, y)) \longleftrightarrow (\exists x. P(x) \wedge (\exists y. Q(x, y))) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

lemma $\langle (\exists y. \forall x. P(x) \longrightarrow Q(x, y)) \longrightarrow (\forall x. P(x) \longrightarrow (\exists y. Q(x, y))) \rangle$
by (*tactic* *IntPr.fast-tac* **context** 1)

end

11 Classical Predicate Calculus Problems

theory *Classical*
imports *FOL*
begin

lemma $\langle (P \longrightarrow Q \vee R) \longrightarrow (P \longrightarrow Q) \vee (P \longrightarrow R) \rangle$
by *blast*

11.0.1 If and only if

lemma $\langle (P \longleftrightarrow Q) \longleftrightarrow (Q \longleftrightarrow P) \rangle$
by *blast*

lemma $\langle \neg (P \longleftrightarrow \neg P) \rangle$
by *blast*

11.1 Pelletier's examples

Sample problems from

- F. J. Pelletier, Seventy-Five Problems for Testing Automatic Theorem Provers, J. Automated Reasoning 2 (1986), 191-216. Errata, JAR 4 (1988), 236-236.

The hardest problems – judging by experience with several theorem provers, including matrix ones – are 34 and 43.

1

lemma $\langle (P \longrightarrow Q) \longleftrightarrow (\neg Q \longrightarrow \neg P) \rangle$
by *blast*

2

lemma $\langle \neg \neg P \longleftrightarrow P \rangle$
by *blast*

3

lemma $\langle \neg (P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \rangle$
by *blast*

4

lemma $\langle (\neg P \longrightarrow Q) \longleftrightarrow (\neg Q \longrightarrow P) \rangle$
by *blast*

5

lemma $\langle ((P \vee Q) \longrightarrow (P \vee R)) \longrightarrow (P \vee (Q \longrightarrow R)) \rangle$
by *blast*

6

lemma $\langle P \vee \neg P \rangle$
by *blast*

7

lemma $\langle P \vee \neg \neg \neg P \rangle$
by *blast*

8. Peirce's law

lemma $\langle ((P \longrightarrow Q) \longrightarrow P) \longrightarrow P \rangle$
by *blast*

9

lemma $\langle ((P \vee Q) \wedge (\neg P \vee Q) \wedge (P \vee \neg Q)) \longrightarrow \neg (\neg P \vee \neg Q) \rangle$
by *blast*

10

lemma $\langle (Q \longrightarrow R) \wedge (R \longrightarrow P \wedge Q) \wedge (P \longrightarrow Q \vee R) \longrightarrow (P \longleftrightarrow Q) \rangle$
by *blast*

11. Proved in each direction (incorrectly, says Pelletier!!)

lemma $\langle P \longleftrightarrow P \rangle$
by *blast*

12. "Dijkstra's law"

lemma $\langle (P \longleftrightarrow Q) \longleftrightarrow R \longleftrightarrow (P \longleftrightarrow (Q \longleftrightarrow R)) \rangle$
by *blast*

13. Distributive law

lemma $\langle P \vee (Q \wedge R) \longleftrightarrow (P \vee Q) \wedge (P \vee R) \rangle$
by *blast*

14

lemma $\langle (P \longleftrightarrow Q) \longleftrightarrow ((Q \vee \neg P) \wedge (\neg Q \vee P)) \rangle$
by *blast*

15

lemma $\langle (P \longrightarrow Q) \longleftrightarrow (\neg P \vee Q) \rangle$
by *blast*

16

lemma $\langle (P \longrightarrow Q) \vee (Q \longrightarrow P) \rangle$
by *blast*

17

lemma $\langle ((P \wedge (Q \longrightarrow R)) \longrightarrow S) \longleftrightarrow ((\neg P \vee Q \vee S) \wedge (\neg P \vee \neg R \vee S)) \rangle$
by *blast*

11.2 Classical Logic: examples with quantifiers

lemma $\langle (\forall x. P(x) \wedge Q(x)) \longleftrightarrow (\forall x. P(x)) \wedge (\forall x. Q(x)) \rangle$
by *blast*

lemma $\langle (\exists x. P \longrightarrow Q(x)) \longleftrightarrow (P \longrightarrow (\exists x. Q(x))) \rangle$
by *blast*

lemma $\langle (\exists x. P(x) \longrightarrow Q) \longleftrightarrow (\forall x. P(x)) \longrightarrow Q \rangle$
by *blast*

lemma $\langle (\forall x. P(x)) \vee Q \longleftrightarrow (\forall x. P(x) \vee Q) \rangle$
by *blast*

Discussed in Avron, Gentzen-Type Systems, Resolution and Tableaux, JAR 10 (265-281), 1993. Proof is trivial!

lemma $\langle \neg ((\exists x. \neg P(x)) \wedge ((\exists x. P(x)) \vee (\exists x. P(x) \wedge Q(x))) \wedge \neg (\exists x. P(x))) \rangle$
by *blast*

11.3 Problems requiring quantifier duplication

Theorem B of Peter Andrews, Theorem Proving via General Matings, JACM 28 (1981).

lemma $\langle (\exists x. \forall y. P(x) \longleftrightarrow P(y)) \longrightarrow ((\exists x. P(x)) \longleftrightarrow (\forall y. P(y))) \rangle$
by *blast*

Needs multiple instantiation of ALL.

lemma $\langle (\forall x. P(x) \longrightarrow P(f(x))) \wedge P(d) \longrightarrow P(f(f(f(d)))) \rangle$
by *blast*

Needs double instantiation of the quantifier

lemma $\langle \exists x. P(x) \longrightarrow P(a) \wedge P(b) \rangle$
by *blast*

lemma $\langle \exists z. P(z) \longrightarrow (\forall x. P(x)) \rangle$
by *blast*

lemma $\langle \exists x. (\exists y. P(y)) \longrightarrow P(x) \rangle$
by *blast*

V. Lifschitz, What Is the Inverse Method?, JAR 5 (1989), 1–23. NOT PROVED.

lemma
 $\langle \exists x x'. \forall y. \exists z z'.$
 $(\neg P(y, y) \vee P(x, x) \vee \neg S(z, x)) \wedge$
 $(S(x, y) \vee \neg S(y, z) \vee Q(z', z')) \wedge$
 $(Q(x', y) \vee \neg Q(y, z') \vee S(x', x')) \rangle$
oops

11.4 Hard examples with quantifiers

18

lemma $\langle \exists y. \forall x. P(y) \longrightarrow P(x) \rangle$
by *blast*

19

lemma $\langle \exists x. \forall y z. (P(y) \longrightarrow Q(z)) \longrightarrow (P(x) \longrightarrow Q(x)) \rangle$
by *blast*

20

lemma $\langle (\forall x y. \exists z. \forall w. (P(x) \wedge Q(y) \longrightarrow R(z) \wedge S(w)))$
 $\longrightarrow (\exists x y. P(x) \wedge Q(y)) \longrightarrow (\exists z. R(z)) \rangle$
by *blast*

21

lemma $\langle (\exists x. P \longrightarrow Q(x)) \wedge (\exists x. Q(x) \longrightarrow P) \longrightarrow (\exists x. P \longleftrightarrow Q(x)) \rangle$
by *blast*

22

lemma $\langle (\forall x. P \longleftrightarrow Q(x)) \longrightarrow (P \longleftrightarrow (\forall x. Q(x))) \rangle$

by *blast*

23

lemma $\langle (\forall x. P \vee Q(x)) \longleftrightarrow (P \vee (\forall x. Q(x))) \rangle$
by *blast*

24

lemma
 $\langle \neg (\exists x. S(x) \wedge Q(x)) \wedge (\forall x. P(x) \longrightarrow Q(x) \vee R(x)) \wedge$
 $(\neg (\exists x. P(x)) \longrightarrow (\exists x. Q(x))) \wedge (\forall x. Q(x) \vee R(x) \longrightarrow S(x))$
 $\longrightarrow (\exists x. P(x) \wedge R(x)) \rangle$
by *blast*

25

lemma
 $\langle (\exists x. P(x)) \wedge$
 $(\forall x. L(x) \longrightarrow \neg (M(x) \wedge R(x))) \wedge$
 $(\forall x. P(x) \longrightarrow (M(x) \wedge L(x))) \wedge$
 $((\forall x. P(x) \longrightarrow Q(x)) \vee (\exists x. P(x) \wedge R(x)))$
 $\longrightarrow (\exists x. Q(x) \wedge P(x)) \rangle$
by *blast*

26

lemma
 $\langle ((\exists x. p(x)) \longleftrightarrow (\exists x. q(x))) \wedge$
 $(\forall x. \forall y. p(x) \wedge q(y) \longrightarrow (r(x) \longleftrightarrow s(y)))$
 $\longrightarrow ((\forall x. p(x) \longrightarrow r(x)) \longleftrightarrow (\forall x. q(x) \longrightarrow s(x))) \rangle$
by *blast*

27

lemma
 $\langle (\exists x. P(x) \wedge \neg Q(x)) \wedge$
 $(\forall x. P(x) \longrightarrow R(x)) \wedge$
 $(\forall x. M(x) \wedge L(x) \longrightarrow P(x)) \wedge$
 $((\exists x. R(x) \wedge \neg Q(x)) \longrightarrow (\forall x. L(x) \longrightarrow \neg R(x)))$
 $\longrightarrow (\forall x. M(x) \longrightarrow \neg L(x)) \rangle$
by *blast*

28. AMENDED

lemma
 $\langle (\forall x. P(x) \longrightarrow (\forall x. Q(x))) \wedge$
 $((\forall x. Q(x) \vee R(x)) \longrightarrow (\exists x. Q(x) \wedge S(x))) \wedge$
 $((\exists x. S(x)) \longrightarrow (\forall x. L(x) \longrightarrow M(x)))$
 $\longrightarrow (\forall x. P(x) \wedge L(x) \longrightarrow M(x)) \rangle$
by *blast*

29. Essentially the same as Principia Mathematica *11.71

lemma

$\langle (\exists x. P(x)) \wedge (\exists y. Q(y))$
 $\longrightarrow ((\forall x. P(x) \longrightarrow R(x)) \wedge (\forall y. Q(y) \longrightarrow S(y)) \longleftrightarrow$
 $(\forall x y. P(x) \wedge Q(y) \longrightarrow R(x) \wedge S(y))) \rangle$
by *blast*

30

lemma

$\langle (\forall x. P(x) \vee Q(x) \longrightarrow \neg R(x)) \wedge$
 $(\forall x. (Q(x) \longrightarrow \neg S(x)) \longrightarrow P(x) \wedge R(x))$
 $\longrightarrow (\forall x. S(x)) \rangle$
by *blast*

31

lemma

$\langle \neg (\exists x. P(x) \wedge (Q(x) \vee R(x))) \wedge$
 $(\exists x. L(x) \wedge P(x)) \wedge$
 $(\forall x. \neg R(x) \longrightarrow M(x))$
 $\longrightarrow (\exists x. L(x) \wedge M(x)) \rangle$
by *blast*

32

lemma

$\langle (\forall x. P(x) \wedge (Q(x) \vee R(x)) \longrightarrow S(x)) \wedge$
 $(\forall x. S(x) \wedge R(x) \longrightarrow L(x)) \wedge$
 $(\forall x. M(x) \longrightarrow R(x))$
 $\longrightarrow (\forall x. P(x) \wedge M(x) \longrightarrow L(x)) \rangle$
by *blast*

33

lemma

$\langle (\forall x. P(a) \wedge (P(x) \longrightarrow P(b)) \longrightarrow P(c)) \longleftrightarrow$
 $(\forall x. (\neg P(a) \vee P(x) \vee P(c)) \wedge (\neg P(a) \vee \neg P(b) \vee P(c))) \rangle$
by *blast*

34. AMENDED (TWICE!!). Andrews's challenge.

lemma

$\langle ((\exists x. \forall y. p(x) \longleftrightarrow p(y)) \longleftrightarrow ((\exists x. q(x)) \longleftrightarrow (\forall y. p(y)))) \longleftrightarrow$
 $((\exists x. \forall y. q(x) \longleftrightarrow q(y)) \longleftrightarrow ((\exists x. p(x)) \longleftrightarrow (\forall y. q(y)))) \rangle$
by *blast*

35

lemma $\langle \exists x y. P(x, y) \longrightarrow (\forall u v. P(u, v)) \rangle$
by *blast*

36

lemma

$\langle (\forall x. \exists y. J(x, y)) \wedge$
 $(\forall x. \exists y. G(x, y)) \wedge$

$(\forall x y. J(x,y) \vee G(x,y) \longrightarrow (\forall z. J(y,z) \vee G(y,z) \longrightarrow H(x,z)))$
 $\longrightarrow (\forall x. \exists y. H(x,y))\rangle$
by *blast*

37

lemma

$\langle (\forall z. \exists w. \forall x. \exists y.$
 $(P(x,z) \longrightarrow P(y,w)) \wedge P(y,z) \wedge (P(y,w) \longrightarrow (\exists u. Q(u,w)))) \wedge$
 $(\forall x z. \neg P(x,z) \longrightarrow (\exists y. Q(y,z))) \wedge$
 $((\exists x y. Q(x,y)) \longrightarrow (\forall x. R(x,x)))$
 $\longrightarrow (\forall x. \exists y. R(x,y))\rangle$
by *blast*

38

lemma

$\langle (\forall x. p(a) \wedge (p(x) \longrightarrow (\exists y. p(y) \wedge r(x,y))) \longrightarrow$
 $(\exists z. \exists w. p(z) \wedge r(x,w) \wedge r(w,z))) \longleftrightarrow$
 $(\forall x. (\neg p(a) \vee p(x) \vee (\exists z. \exists w. p(z) \wedge r(x,w) \wedge r(w,z))) \wedge$
 $(\neg p(a) \vee \neg (\exists y. p(y) \wedge r(x,y)) \vee$
 $(\exists z. \exists w. p(z) \wedge r(x,w) \wedge r(w,z))))\rangle$
by *blast*

39

lemma $\langle \neg (\exists x. \forall y. F(y,x) \longleftrightarrow \neg F(y,y)) \rangle$
by *blast*

40. AMENDED

lemma

$\langle (\exists y. \forall x. F(x,y) \longleftrightarrow F(x,x)) \longrightarrow$
 $\neg (\forall x. \exists y. \forall z. F(z,y) \longleftrightarrow \neg F(z,x))\rangle$
by *blast*

41

lemma

$\langle (\forall z. \exists y. \forall x. f(x,y) \longleftrightarrow f(x,z) \wedge \neg f(x,x))$
 $\longrightarrow \neg (\exists z. \forall x. f(x,z))\rangle$
by *blast*

42

lemma $\langle \neg (\exists y. \forall x. p(x,y) \longleftrightarrow \neg (\exists z. p(x,z) \wedge p(z,x))) \rangle$
by *blast*

43

lemma

$\langle (\forall x. \forall y. q(x,y) \longleftrightarrow (\forall z. p(z,x) \longleftrightarrow p(z,y)))$
 $\longrightarrow (\forall x. \forall y. q(x,y) \longleftrightarrow q(y,x))\rangle$
by *blast*

Other proofs: Can use *auto*, which cheats by using rewriting! *Deepen-tac* alone requires 253 secs. Or *by (mini-tac 1 THEN Deepen-tac 5 1)*.

44

lemma

$\langle (\forall x. f(x) \longrightarrow (\exists y. g(y) \wedge h(x,y) \wedge (\exists y. g(y) \wedge \neg h(x,y)))) \wedge$
 $(\exists x. j(x) \wedge (\forall y. g(y) \longrightarrow h(x,y)))$
 $\longrightarrow (\exists x. j(x) \wedge \neg f(x)) \rangle$
by *blast*

45

lemma

$\langle (\forall x. f(x) \wedge (\forall y. g(y) \wedge h(x,y) \longrightarrow j(x,y))$
 $\longrightarrow (\forall y. g(y) \wedge h(x,y) \longrightarrow k(y))) \wedge$
 $\neg (\exists y. l(y) \wedge k(y)) \wedge$
 $(\exists x. f(x) \wedge (\forall y. h(x,y) \longrightarrow l(y)) \wedge (\forall y. g(y) \wedge h(x,y) \longrightarrow j(x,y)))$
 $\longrightarrow (\exists x. f(x) \wedge \neg (\exists y. g(y) \wedge h(x,y))) \rangle$
by *blast*

46

lemma

$\langle (\forall x. f(x) \wedge (\forall y. f(y) \wedge h(y,x) \longrightarrow g(y)) \longrightarrow g(x)) \wedge$
 $((\exists x. f(x) \wedge \neg g(x)) \longrightarrow$
 $(\exists x. f(x) \wedge \neg g(x) \wedge (\forall y. f(y) \wedge \neg g(y) \longrightarrow j(x,y)))) \wedge$
 $(\forall x y. f(x) \wedge f(y) \wedge h(x,y) \longrightarrow \neg j(y,x))$
 $\longrightarrow (\forall x. f(x) \longrightarrow g(x)) \rangle$
by *blast*

11.5 Problems (mainly) involving equality or functions

48

lemma $\langle (a = b \vee c = d) \wedge (a = c \vee b = d) \longrightarrow a = d \vee b = c \rangle$
by *blast*

49. NOT PROVED AUTOMATICALLY. Hard because it involves substitution for Vars; the type constraint ensures that x,y,z have the same type as a,b,u.

lemma

$\langle (\exists x y::'a. \forall z. z = x \vee z = y) \wedge P(a) \wedge P(b) \wedge a \neq b \longrightarrow (\forall u::'a. P(u)) \rangle$
apply *safe*
apply (*rule-tac* $x = \langle a \rangle$ **in** *allE*, *assumption*)
apply (*rule-tac* $x = \langle b \rangle$ **in** *allE*, *assumption*)
apply *fast* — blast's treatment of equality can't do it
done

50. (What has this to do with equality?)

lemma $\langle (\forall x. P(a,x) \vee (\forall y. P(x,y))) \longrightarrow (\exists x. \forall y. P(x,y)) \rangle$

by *blast*

51

lemma

$\langle (\exists z w. \forall x y. P(x,y) \longleftrightarrow (x = z \wedge y = w)) \longrightarrow$
 $(\exists z. \forall x. \exists w. (\forall y. P(x,y) \longleftrightarrow y=w) \longleftrightarrow x = z) \rangle$

by *blast*

52

Almost the same as 51.

lemma

$\langle (\exists z w. \forall x y. P(x,y) \longleftrightarrow (x = z \wedge y = w)) \longrightarrow$
 $(\exists w. \forall y. \exists z. (\forall x. P(x,y) \longleftrightarrow x = z) \longleftrightarrow y = w) \rangle$

by *blast*

55

Non-equational version, from Manthey and Bry, CADE-9 (Springer, 1988).
fast DISCOVERS who killed Agatha.

schematic-goal

$\langle \text{lives}(\text{agatha}) \wedge \text{lives}(\text{butler}) \wedge \text{lives}(\text{charles}) \wedge$
 $(\text{killed}(\text{agatha}, \text{agatha}) \vee \text{killed}(\text{butler}, \text{agatha}) \vee \text{killed}(\text{charles}, \text{agatha})) \wedge$
 $(\forall x y. \text{killed}(x,y) \longrightarrow \text{hates}(x,y) \wedge \neg \text{richer}(x,y)) \wedge$
 $(\forall x. \text{hates}(\text{agatha}, x) \longrightarrow \neg \text{hates}(\text{charles}, x)) \wedge$
 $(\text{hates}(\text{agatha}, \text{agatha}) \wedge \text{hates}(\text{agatha}, \text{charles})) \wedge$
 $(\forall x. \text{lives}(x) \wedge \neg \text{richer}(x, \text{agatha}) \longrightarrow \text{hates}(\text{butler}, x)) \wedge$
 $(\forall x. \text{hates}(\text{agatha}, x) \longrightarrow \text{hates}(\text{butler}, x)) \wedge$
 $(\forall x. \neg \text{hates}(x, \text{agatha}) \vee \neg \text{hates}(x, \text{butler}) \vee \neg \text{hates}(x, \text{charles})) \longrightarrow$
 $\text{killed}(\text{?who}, \text{agatha}) \rangle$

by *fast* — MUCH faster than *blast*

56

lemma $\langle (\forall x. (\exists y. P(y) \wedge x = f(y)) \longrightarrow P(x)) \longleftrightarrow (\forall x. P(x) \longrightarrow P(f(x))) \rangle$

by *blast*

57

lemma

$\langle P(f(a,b), f(b,c)) \wedge P(f(b,c), f(a,c)) \wedge$
 $(\forall x y z. P(x,y) \wedge P(y,z) \longrightarrow P(x,z)) \longrightarrow P(f(a,b), f(a,c)) \rangle$

by *blast*

58 NOT PROVED AUTOMATICALLY

lemma $\langle (\forall x y. f(x) = g(y)) \longrightarrow (\forall x y. f(f(x)) = f(g(y))) \rangle$

by (*slow elim: subst-context*)

59

lemma $\langle (\forall x. P(x) \longleftrightarrow \neg P(f(x))) \longrightarrow (\exists x. P(x) \wedge \neg P(f(x))) \rangle$

by *blast*

60

lemma $\langle \forall x. P(x, f(x)) \longleftrightarrow (\exists y. (\forall z. P(z, y) \longrightarrow P(z, f(x))) \wedge P(x, y)) \rangle$
by *blast*

62 as corrected in JAR 18 (1997), page 135

lemma

$\langle (\forall x. p(a) \wedge (p(x) \longrightarrow p(f(x))) \longrightarrow p(f(f(x)))) \longleftrightarrow$
 $(\forall x. (\neg p(a) \vee p(x) \vee p(f(f(x)))) \wedge$
 $(\neg p(a) \vee \neg p(f(x)) \vee p(f(f(x)))) \rangle$
by *blast*

From Davis, Obvious Logical Inferences, IJCAI-81, 530-531 fast indeed copes!

lemma

$\langle (\forall x. F(x) \wedge \neg G(x) \longrightarrow (\exists y. H(x, y) \wedge J(y))) \wedge$
 $(\exists x. K(x) \wedge F(x) \wedge (\forall y. H(x, y) \longrightarrow K(y))) \wedge$
 $(\forall x. K(x) \longrightarrow \neg G(x)) \longrightarrow (\exists x. K(x) \wedge J(x)) \rangle$
by *fast*

From Rudnicki, Obvious Inferences, JAR 3 (1987), 383-393. It does seem obvious!

lemma

$\langle (\forall x. F(x) \wedge \neg G(x) \longrightarrow (\exists y. H(x, y) \wedge J(y))) \wedge$
 $(\exists x. K(x) \wedge F(x) \wedge (\forall y. H(x, y) \longrightarrow K(y))) \wedge$
 $(\forall x. K(x) \longrightarrow \neg G(x)) \longrightarrow (\exists x. K(x) \longrightarrow \neg G(x)) \rangle$
by *fast*

Halting problem: Formulation of Li Dafa (AAR Newsletter 27, Oct 1994.)
author U. Egly.

lemma

$\langle ((\exists x. A(x) \wedge (\forall y. C(y) \longrightarrow (\forall z. D(x, y, z)))) \longrightarrow$
 $(\exists w. C(w) \wedge (\forall y. C(y) \longrightarrow (\forall z. D(w, y, z))))$
 $\wedge$
 $(\forall w. C(w) \wedge (\forall u. C(u) \longrightarrow (\forall v. D(w, u, v))) \longrightarrow$
 $(\forall y z.$
 $(C(y) \wedge P(y, z) \longrightarrow Q(w, y, z) \wedge OO(w, g)) \wedge$
 $(C(y) \wedge \neg P(y, z) \longrightarrow Q(w, y, z) \wedge OO(w, b))))$
 $\wedge$
 $(\forall w. C(w) \wedge$
 $(\forall y z.$
 $(C(y) \wedge P(y, z) \longrightarrow Q(w, y, z) \wedge OO(w, g)) \wedge$
 $(C(y) \wedge \neg P(y, z) \longrightarrow Q(w, y, z) \wedge OO(w, b))) \longrightarrow$
 $(\exists v. C(v) \wedge$
 $(\forall y. ((C(y) \wedge Q(w, y, y)) \wedge OO(w, g) \longrightarrow \neg P(v, y)) \wedge$
 $((C(y) \wedge Q(w, y, y)) \wedge OO(w, b) \longrightarrow P(v, y) \wedge OO(v, b))))$
 $\longrightarrow \neg (\exists x. A(x) \wedge (\forall y. C(y) \longrightarrow (\forall z. D(x, y, z)))) \rangle$
by (*blast 12*)

— Needed because the search for depths below 12 is very slow.

Halting problem II: credited to M. Bruschi by Li Dafa in JAR 18(1), p. 105.

lemma

$$\begin{aligned}
& \langle ((\exists x. A(x) \wedge (\forall y. C(y) \longrightarrow (\forall z. D(x,y,z)))) \longrightarrow \\
& \quad (\exists w. C(w) \wedge (\forall y. C(y) \longrightarrow (\forall z. D(w,y,z))))) \\
& \quad \wedge \\
& \quad (\forall w. C(w) \wedge (\forall u. C(u) \longrightarrow (\forall v. D(w,u,v))) \longrightarrow \\
& \quad \quad (\forall y z. \\
& \quad \quad \quad (C(y) \wedge P(y,z) \longrightarrow Q(w,y,z) \wedge OO(w,g)) \wedge \\
& \quad \quad \quad (C(y) \wedge \neg P(y,z) \longrightarrow Q(w,y,z) \wedge OO(w,b)))) \\
& \quad \wedge \\
& \quad ((\exists w. C(w) \wedge (\forall y. (C(y) \wedge P(y,y) \longrightarrow Q(w,y,y) \wedge OO(w,g)) \wedge \\
& \quad \quad (C(y) \wedge \neg P(y,y) \longrightarrow Q(w,y,y) \wedge OO(w,b)))) \\
& \quad \longrightarrow \\
& \quad (\exists v. C(v) \wedge (\forall y. (C(y) \wedge P(y,y) \longrightarrow P(v,y) \wedge OO(v,g)) \wedge \\
& \quad \quad (C(y) \wedge \neg P(y,y) \longrightarrow P(v,y) \wedge OO(v,b)))) \\
& \quad \longrightarrow \\
& \quad ((\exists v. C(v) \wedge (\forall y. (C(y) \wedge P(y,y) \longrightarrow P(v,y) \wedge OO(v,g)) \wedge \\
& \quad \quad (C(y) \wedge \neg P(y,y) \longrightarrow P(v,y) \wedge OO(v,b)))) \\
& \quad \longrightarrow \\
& \quad (\exists u. C(u) \wedge (\forall y. (C(y) \wedge P(y,y) \longrightarrow \neg P(u,y)) \wedge \\
& \quad \quad (C(y) \wedge \neg P(y,y) \longrightarrow P(u,y) \wedge OO(u,b)))) \\
& \quad \longrightarrow \neg (\exists x. A(x) \wedge (\forall y. C(y) \longrightarrow (\forall z. D(x,y,z)))) \rangle
\end{aligned}$$

by *blast*

Challenge found on info-hol.

lemma $\langle \forall x. \exists v w. \forall y z. P(x) \wedge Q(y) \longrightarrow (P(v) \vee R(w)) \wedge (R(z) \longrightarrow Q(v)) \rangle$

by *blast*

Attributed to Lewis Carroll by S. G. Pulman. The first or last assumption can be deleted.

lemma

$$\begin{aligned}
& \langle (\forall x. \text{honest}(x) \wedge \text{industrious}(x) \longrightarrow \text{healthy}(x)) \wedge \\
& \quad \neg (\exists x. \text{grocer}(x) \wedge \text{healthy}(x)) \wedge \\
& \quad (\forall x. \text{industrious}(x) \wedge \text{grocer}(x) \longrightarrow \text{honest}(x)) \wedge \\
& \quad (\forall x. \text{cyclist}(x) \longrightarrow \text{industrious}(x)) \wedge \\
& \quad (\forall x. \neg \text{healthy}(x) \wedge \text{cyclist}(x) \longrightarrow \neg \text{honest}(x)) \\
& \quad \longrightarrow (\forall x. \text{grocer}(x) \longrightarrow \neg \text{cyclist}(x)) \rangle
\end{aligned}$$

by *blast*

end

12 First-Order Logic: propositional examples (classical version)

```
theory Propositional-Cla
imports FOL
begin
```

commutative laws of $\wedge$ and $\vee$

```
lemma  $\langle P \wedge Q \longrightarrow Q \wedge P \rangle$ 
  by (tactic IntPr.fast-tac context 1)
```

```
lemma  $\langle P \vee Q \longrightarrow Q \vee P \rangle$ 
  by fast
```

associative laws of $\wedge$ and $\vee$

```
lemma  $\langle (P \wedge Q) \wedge R \longrightarrow P \wedge (Q \wedge R) \rangle$ 
  by fast
```

```
lemma  $\langle (P \vee Q) \vee R \longrightarrow P \vee (Q \vee R) \rangle$ 
  by fast
```

distributive laws of $\wedge$ and $\vee$

```
lemma  $\langle (P \wedge Q) \vee R \longrightarrow (P \vee R) \wedge (Q \vee R) \rangle$ 
  by fast
```

```
lemma  $\langle (P \vee R) \wedge (Q \vee R) \longrightarrow (P \wedge Q) \vee R \rangle$ 
  by fast
```

```
lemma  $\langle (P \vee Q) \wedge R \longrightarrow (P \wedge R) \vee (Q \wedge R) \rangle$ 
  by fast
```

```
lemma  $\langle (P \wedge R) \vee (Q \wedge R) \longrightarrow (P \vee Q) \wedge R \rangle$ 
  by fast
```

Laws involving implication

```
lemma  $\langle (P \longrightarrow R) \wedge (Q \longrightarrow R) \longleftrightarrow (P \vee Q \longrightarrow R) \rangle$ 
  by fast
```

```
lemma  $\langle (P \wedge Q \longrightarrow R) \longleftrightarrow (P \longrightarrow (Q \longrightarrow R)) \rangle$ 
  by fast
```

```
lemma  $\langle ((P \longrightarrow R) \longrightarrow R) \longrightarrow ((Q \longrightarrow R) \longrightarrow R) \longrightarrow (P \wedge Q \longrightarrow R) \longrightarrow R \rangle$ 
  by fast
```

```
lemma  $\langle \neg (P \longrightarrow R) \longrightarrow \neg (Q \longrightarrow R) \longrightarrow \neg (P \wedge Q \longrightarrow R) \rangle$ 
  by fast
```

```
lemma  $\langle (P \longrightarrow Q \wedge R) \longleftrightarrow (P \longrightarrow Q) \wedge (P \longrightarrow R) \rangle$ 
```

by *fast*

Propositions-as-types

lemma $\langle P \rightarrow (Q \rightarrow P) \rangle$
by *fast*

— The combinator S

lemma $\langle (P \rightarrow Q \rightarrow R) \rightarrow (P \rightarrow Q) \rightarrow (P \rightarrow R) \rangle$
by *fast*

— Converse is classical

lemma $\langle (P \rightarrow Q) \vee (P \rightarrow R) \rightarrow (P \rightarrow Q \vee R) \rangle$
by *fast*

lemma $\langle (P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P) \rangle$
by *fast*

Schwichtenberg’s examples (via T. Nipkow)

lemma *stab-imp*: $\langle (((Q \rightarrow R) \rightarrow R) \rightarrow Q) \rightarrow (((P \rightarrow Q) \rightarrow R) \rightarrow R) \rightarrow P \rightarrow Q \rangle$
by *fast*

lemma *stab-to-peirce*:
 $\langle (((P \rightarrow R) \rightarrow R) \rightarrow P) \rightarrow (((Q \rightarrow R) \rightarrow R) \rightarrow Q) \rightarrow ((P \rightarrow Q) \rightarrow P) \rightarrow P \rangle$
by *fast*

lemma *peirce-imp1*:
 $\langle (((Q \rightarrow R) \rightarrow Q) \rightarrow Q) \rightarrow ((P \rightarrow Q) \rightarrow R) \rightarrow P \rightarrow Q \rangle$
by *fast*

lemma *peirce-imp2*: $\langle (((P \rightarrow R) \rightarrow P) \rightarrow P) \rightarrow ((P \rightarrow Q \rightarrow R) \rightarrow P) \rightarrow P \rangle$
by *fast*

lemma *mints*: $\langle (((P \rightarrow Q) \rightarrow P) \rightarrow P) \rightarrow Q \rightarrow Q \rangle$
by *fast*

lemma *mints-solovev*: $\langle (P \rightarrow (Q \rightarrow R) \rightarrow Q) \rightarrow ((P \rightarrow Q) \rightarrow R) \rightarrow R \rangle$
by *fast*

lemma *tatsuta*:
 $\langle (((P7 \rightarrow P1) \rightarrow P10) \rightarrow P4 \rightarrow P5) \rightarrow (((P8 \rightarrow P2) \rightarrow P9) \rightarrow P3 \rightarrow P10) \rightarrow (P1 \rightarrow P8) \rightarrow P6 \rightarrow P7 \rightarrow (((P3 \rightarrow P2) \rightarrow P9) \rightarrow P4) \rightarrow (P1 \rightarrow P3) \rightarrow (((P6 \rightarrow P1) \rightarrow P2) \rightarrow P9) \rightarrow P5 \rangle$

```

by fast

lemma tatsuta1:
  ⟨(((P8 → P2) → P9) → P3 → P10)
  → (((P3 → P2) → P9) → P4)
  → (((P6 → P1) → P2) → P9)
  → (((P7 → P1) → P10) → P4 → P5)
  → (P1 → P3) → (P1 → P8) → P6 → P7 → P5⟩
by fast

end

```

13 First-Order Logic: quantifier examples (classical version)

```

theory Quantifiers-Cla
imports FOL
begin

```

```

lemma ⟨(∀ x y. P(x,y)) → (∀ y x. P(x,y))⟩
by fast

```

```

lemma ⟨(∃ x y. P(x,y)) → (∃ y x. P(x,y))⟩
by fast

```

Converse is false.

```

lemma ⟨(∀ x. P(x)) ∨ (∀ x. Q(x)) → (∀ x. P(x) ∨ Q(x))⟩
by fast

```

```

lemma ⟨(∀ x. P → Q(x)) ↔ (P → (∀ x. Q(x)))⟩
by fast

```

```

lemma ⟨(∀ x. P(x) → Q) ↔ ((∃ x. P(x)) → Q)⟩
by fast

```

Some harder ones.

```

lemma ⟨(∃ x. P(x) ∨ Q(x)) ↔ (∃ x. P(x)) ∨ (∃ x. Q(x))⟩
by fast

```

— Converse is false.

```

lemma ⟨(∃ x. P(x) ∧ Q(x)) → (∃ x. P(x)) ∧ (∃ x. Q(x))⟩
by fast

```

Basic test of quantifier reasoning.

```

lemma ⟨(∃ y. ∀ x. Q(x,y)) → (∀ x. ∃ y. Q(x,y))⟩
by fast

```

lemma $\langle (\forall x. Q(x)) \longrightarrow (\exists x. Q(x)) \rangle$
by *fast*

The following should fail, as they are false!

lemma $\langle (\forall x. \exists y. Q(x,y)) \longrightarrow (\exists y. \forall x. Q(x,y)) \rangle$
apply *fast?*
oops

lemma $\langle (\exists x. Q(x)) \longrightarrow (\forall x. Q(x)) \rangle$
apply *fast?*
oops

schematic-goal $\langle P(?a) \longrightarrow (\forall x. P(x)) \rangle$
apply *fast?*
oops

schematic-goal $\langle (P(?a) \longrightarrow (\forall x. Q(x))) \longrightarrow (\forall x. P(x) \longrightarrow Q(x)) \rangle$
apply *fast?*
oops

Back to things that are provable ...

lemma $\langle (\forall x. P(x) \longrightarrow Q(x)) \wedge (\exists x. P(x)) \longrightarrow (\exists x. Q(x)) \rangle$
by *fast*

An example of why *exI* should be delayed as long as possible.

lemma $\langle (P \longrightarrow (\exists x. Q(x))) \wedge P \longrightarrow (\exists x. Q(x)) \rangle$
by *fast*

schematic-goal $\langle (\forall x. P(x) \longrightarrow Q(f(x))) \wedge (\forall x. Q(x) \longrightarrow R(g(x))) \wedge P(d) \longrightarrow R(?a) \rangle$
by *fast*

lemma $\langle (\forall x. Q(x)) \longrightarrow (\exists x. Q(x)) \rangle$
by *fast*

Some slow ones

Principia Mathematica *11.53

lemma $\langle (\forall x y. P(x) \longrightarrow Q(y)) \longleftrightarrow ((\exists x. P(x)) \longrightarrow (\forall y. Q(y))) \rangle$
by *fast*

lemma $\langle (\exists x y. P(x) \wedge Q(x,y)) \longleftrightarrow (\exists x. P(x) \wedge (\exists y. Q(x,y))) \rangle$
by *fast*

lemma $\langle (\exists y. \forall x. P(x) \longrightarrow Q(x,y)) \longrightarrow (\forall x. P(x) \longrightarrow (\exists y. Q(x,y))) \rangle$
by *fast*

end

theory *Miniscope*
imports *FOL*
begin

lemmas *ccontr* = *FalseE* [*THEN classical*]

13.1 Negation Normal Form

13.1.1 de Morgan laws

lemma *demorgans1*:

$$\langle \neg (P \wedge Q) \longleftrightarrow \neg P \vee \neg Q \rangle$$

$$\langle \neg (P \vee Q) \longleftrightarrow \neg P \wedge \neg Q \rangle$$

$$\langle \neg \neg P \longleftrightarrow P \rangle$$

by *blast+*

lemma *demorgans2*:

$$\langle \bigwedge P. \neg (\forall x. P(x)) \longleftrightarrow (\exists x. \neg P(x)) \rangle$$

$$\langle \bigwedge P. \neg (\exists x. P(x)) \longleftrightarrow (\forall x. \neg P(x)) \rangle$$

by *blast+*

lemmas *demorgans* = *demorgans1 demorgans2*

lemma *nnf-simps*:

$$\langle (P \longrightarrow Q) \longleftrightarrow (\neg P \vee Q) \rangle$$

$$\langle \neg (P \longrightarrow Q) \longleftrightarrow (P \wedge \neg Q) \rangle$$

$$\langle (P \longleftrightarrow Q) \longleftrightarrow (\neg P \vee Q) \wedge (\neg Q \vee P) \rangle$$

$$\langle \neg (P \longleftrightarrow Q) \longleftrightarrow (P \vee Q) \wedge (\neg P \vee \neg Q) \rangle$$

by *blast+*

13.1.2 Pushing in the existential quantifiers

lemma *ex-simps*:

$$\langle (\exists x. P) \longleftrightarrow P \rangle$$

$$\langle \bigwedge P Q. (\exists x. P(x) \wedge Q) \longleftrightarrow (\exists x. P(x)) \wedge Q \rangle$$

$$\langle \bigwedge P Q. (\exists x. P \wedge Q(x)) \longleftrightarrow P \wedge (\exists x. Q(x)) \rangle$$

$$\langle \bigwedge P Q. (\exists x. P(x) \vee Q(x)) \longleftrightarrow (\exists x. P(x)) \vee (\exists x. Q(x)) \rangle$$

$$\langle \bigwedge P Q. (\exists x. P(x) \vee Q) \longleftrightarrow (\exists x. P(x)) \vee Q \rangle$$

$$\langle \bigwedge P Q. (\exists x. P \vee Q(x)) \longleftrightarrow P \vee (\exists x. Q(x)) \rangle$$

by *blast+*

13.1.3 Pushing in the universal quantifiers

lemma *all-simps*:

$$\langle (\forall x. P) \longleftrightarrow P \rangle$$

```

⟨ ∧ P Q. (∀ x. P(x) ∧ Q(x)) ⟷ (∀ x. P(x)) ∧ (∀ x. Q(x)) ⟩
⟨ ∧ P Q. (∀ x. P(x) ∧ Q) ⟷ (∀ x. P(x)) ∧ Q ⟩
⟨ ∧ P Q. (∀ x. P ∧ Q(x)) ⟷ P ∧ (∀ x. Q(x)) ⟩
⟨ ∧ P Q. (∀ x. P(x) ∨ Q) ⟷ (∀ x. P(x)) ∨ Q ⟩
⟨ ∧ P Q. (∀ x. P ∨ Q(x)) ⟷ P ∨ (∀ x. Q(x)) ⟩
by blast+

```

lemmas *mini-simps* = *demorgans nnf-simps ex-simps all-simps*

```

ML ⟨
  val mini-ss = simpset-of (context addsimps @ {thms mini-simps});
  fun mini-tac ctxt =
    resolve-tac ctxt @ {thms ccontr} THEN' asm-full-simp-tac (put-simpset mini-ss
    ctxt);
  ⟩

end

```

14 First-Order Logic: the 'if' example

```

theory If
imports FOL
begin

```

```

definition if :: ⟨ [o,o,o] => o ⟩
  where ⟨ if(P,Q,R) ≡ P ∧ Q ∨ ¬ P ∧ R ⟩

```

```

lemma ifI: ⟨ [P ⟹ Q; ¬ P ⟹ R] ⟹ if(P,Q,R) ⟩
  unfolding if-def by blast

```

```

lemma ifE: ⟨ [if(P,Q,R); [P; Q] ⟹ S; [¬ P; R] ⟹ S] ⟹ S ⟩
  unfolding if-def by blast

```

```

lemma if-commute: ⟨ if(P, if(Q,A,B), if(Q,C,D)) ⟷ if(Q, if(P,A,C), if(P,B,D)) ⟩
  apply (rule iffI)
  apply (erule ifE)
  apply (erule ifE)
  apply (rule ifI)
  apply (rule ifI)
  oops

```

Trying again from the beginning in order to use *blast*

```

declare ifI [intro!]
declare ifE [elim!]

```

```

lemma if-commute: ⟨ if(P, if(Q,A,B), if(Q,C,D)) ⟷ if(Q, if(P,A,C), if(P,B,D)) ⟩
  by blast

```

lemma $\langle \text{if}(P, Q, R), A, B \rangle \longleftrightarrow \text{if}(P, \text{if}(Q, A, B), \text{if}(R, A, B)) \rangle$
by *blast*

Trying again from the beginning in order to prove from the definitions

lemma $\langle \text{if}(P, Q, R), A, B \rangle \longleftrightarrow \text{if}(P, \text{if}(Q, A, B), \text{if}(R, A, B)) \rangle$
unfolding *if-def* **by** *blast*

An invalid formula. High-level rules permit a simpler diagnosis.

lemma $\langle \text{if}(P, Q, R), A, B \rangle \longleftrightarrow \text{if}(P, \text{if}(Q, A, B), \text{if}(R, B, A)) \rangle$
apply *auto*
— The next step will fail unless subgoals remain
apply (*tactic all-tac*)
oops

Trying again from the beginning in order to prove from the definitions.

lemma $\langle \text{if}(P, Q, R), A, B \rangle \longleftrightarrow \text{if}(P, \text{if}(Q, A, B), \text{if}(R, B, A)) \rangle$
unfolding *if-def*
apply *auto*
— The next step will fail unless subgoals remain
apply (*tactic all-tac*)
oops

end