Miscellaneous FOL Examples

March 13, 2025

Contents

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: ‹rec(Suc(m), a, f) = f(m, rec(m, a, f))›

lemma Suc-n-not-n: ‹Suc(k) ≠ k›
proof (induct ‹k›)
  show ‹Suc(0) ≠ 0›
  proof
    assume ‹Suc(0) = 0›
    then show ‹False› by (rule Suc-neq-0)
  qed
next
  fix n assume hyp: ‹Suc(n) ≠ n›
  show ‹Suc(Suc(n)) ≠ Suc(n)›
  proof
    assume ‹Suc(Suc(n)) = Suc(n)›
    then have ‹Suc(n) = n› by (rule Suc-inject)
    with hyp show ‹False› by contradiction
  qed
qed


definition add :: ‹nat => nat => nat›    (infixl ‹+› 60)
  where ‹m + n = rec(m, n, λx y. Suc(y))›

lemma add-0 [simp]: ‹0 + n = n›
  unfolding add-def by (rule rec-0)

lemma add-Suc [simp]: ‹Suc(m) + n = Suc(m + n)›
  unfolding add-def by (rule rec-Suc)

lemma add-assoc: ‹(k + m) + n = k + (m + n)›
  by (induct ‹k›) simp-all

lemma add-0-right: ‹m + 0 = m›
  by (induct ‹m›) simp-all

lemma add-Suc-right: ‹m + Suc(n) = Suc(m + n)›
  by (induct ‹m›) simp-all

lemma
  assumes ‹!!n. f(Suc(n)) = Suc(f(n))›
  shows ‹f(i + j) = i + f(j)›
  using assms by (induct ‹i›) 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*: ‹$P \lor P \longrightarrow P$›
apply (*rule impI*)
apply (*rule disjE*)
prefer *3* **apply** (*assumption*)
prefer *2* **apply** (*assumption*)
apply *assumption*
done

lemma ‹$(P \land Q) \lor R \longrightarrow (P \lor R)$›
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 ‹$(\forall x\ y.\ P(x,y)) \longrightarrow (\forall z\ w.\ P(w,z))$›
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 ‹$(\exists y.\ \forall x.\ J(y,x) \longleftrightarrow \neg J(x,x)) \longrightarrow \neg (\forall x.\ \exists y.\ \forall z.\ J(z,y) \longleftrightarrow \neg J(z,x))$›
apply *fast*
done

lemma ‹$\forall x.\ P(x,f(x)) \longleftrightarrow (\exists y.\ (\forall z.\ P(z,y) \longrightarrow P(z,f(x))) \land P(x,y))$›
apply *fast*
done

2.0.3 Derivation of conjunction elimination rule

lemma

assumes *major*: ‹$P \wedge Q$›
and *minor*: ‹$[\![P;\ Q]\!] \Longrightarrow R$›
shows ‹R›
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 ‹$P \Longrightarrow$ *False*›
shows ‹$\neg\ P$›
apply (*unfold not-def*)
apply (*rule impI*)
apply (*rule assms*)
apply *assumption*
done

lemma
assumes *major*: ‹$\neg\ P$›
and *minor*: ‹P›
shows ‹R›
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*: ‹$\neg\ P$›
and *minor*: ‹P›
shows ‹R›
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* ⇒ *nat*› **and**
  rec :: ‹[*nat*, *'a*, [*nat*, *'a*] ⇒ *'a*] ⇒ *'a*›
where
  induct: ‹$[\![P(0);\ \bigwedge x.\ P(x) \Longrightarrow P(Suc(x))]\!] \Longrightarrow P(n)$› **and**
  Suc-inject: ‹$Suc(m)=Suc(n) \Longrightarrow m=n$› **and**
  Suc-neq-0: ‹$Suc(m)=0 \Longrightarrow R$› **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*] ⇒ *nat*› (**infixl** ‹+› *60*)
  where ‹$m + n \equiv rec(m,\ n,\ \lambda x\ y.\ Suc(y))$›

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 = ‹k› **in** *induct*)
apply *simp*
apply *simp*
done

lemma *add-0-right*: ‹$m+0 = m$›
apply (*rule-tac* n = ‹m› **in** *induct*)
apply *simp*
apply *simp*
done

lemma *add-Suc-right*: ‹$m+Suc(n) = Suc(m+n)$›
apply (*rule-tac* n = ‹m› **in** *induct*)
apply *simp-all*
done

lemma
  assumes *prem*: ‹$\bigwedge n.\ f(Suc(n)) = Suc(f(n))$›
  shows ‹$f(i+j) = i+f(j)$›
apply (*rule-tac* n = ‹i› **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* :: ‹$'a$› (‹0›)
    and *Suc* :: ‹$'a \Rightarrow 'a$›
    and *rec* :: ‹$'a \Rightarrow 'a \Rightarrow ('a \Rightarrow 'a \Rightarrow 'a) \Rightarrow 'a$›
  assumes *induct*: ‹$P(0) \Longrightarrow (\bigwedge x.\ P(x) \Longrightarrow P(Suc(x))) \Longrightarrow P(n)$›
    and *Suc-inject*: ‹$Suc(m) = Suc(n) \Longrightarrow m = n$›
    and *Suc-neq-Zero*: ‹$Suc(m) = 0 \Longrightarrow R$›
    and *rec-Zero*: ‹$rec(0,\ a,\ f) = a$›
    and *rec-Suc*: ‹$rec(Suc(m),\ a,\ f) = f(m,\ rec(m,\ a,\ f))$›
begin

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

lemma *Suc-n-not-n*: ‹$Suc(k) \neq (k::'a)$›
  apply (*rule-tac n* = ‹*k*› **in** *induct*)
   apply (*rule notI*)
   apply (*erule Suc-neq-Zero*)
  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
  oops

lemma *add-Zero* [*simp*]: ‹$0 + n = n$›
  apply (*unfold add-def*)
  apply (*rule rec-Zero*)
  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* = ‹*k*› **in** *induct*)
   apply *simp*
  apply *simp*
  done

lemma *add-Zero-right*: ‹$m + 0 = m$›
  apply (*rule-tac n* = ‹*m*› **in** *induct*)
   apply *simp*
  apply *simp*
  done

lemma *add-Suc-right*: ‹$m + Suc(n) = Suc(m + n)$›
  apply (*rule-tac n* = ‹*m*› **in** *induct*)
   apply *simp-all*
  done

lemma

  assumes *prem*: ‹$\bigwedge n.\ f(Suc(n)) = Suc(f(n))$›
  shows ‹$f(i + j) = i + f(j)$›
  apply (*rule-tac* n = ‹i› **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 ‹$A \wedge B \longrightarrow (C \longrightarrow A \wedge C)$›
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 ‹$A \wedge B$›
    and ‹$A \Longrightarrow B \Longrightarrow C$›
  shows ‹C›
apply (*rule assms*)
apply (*rule conjunct1*)
apply (*rule assms*)
apply (*rule conjunct2*)
apply (*rule assms*)
done

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

apply (*rule-tac* [*2*] *disjI1*)
prefer *2* **apply** *assumption*
apply (*rule notI*)
apply (*rule-tac* P = ‹$B \lor \neg\, B$› **in** *notE*)
apply *assumption*
apply (*rule disjI2*)
apply *assumption*
done

lemma
  assumes ‹$\bigwedge A.\ \neg\, \neg\, A \Longrightarrow A$›
  shows ‹$B \lor \neg\, B$›
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 ‹$A \lor \neg\, A$›
    and ‹$\neg\, \neg\, A$›
  shows ‹A›
apply (*rule disjE*)
apply (*rule assms*)
apply *assumption*
apply (*rule FalseE*)
apply (*rule-tac* P = ‹$\neg\, A$› **in** *notE*)
apply (*rule assms*)
apply *assumption*
done

5.1 Examples with quantifiers

lemma
  assumes ‹$\forall\, z.\ G(z)$›
  shows ‹$\forall\, z.\ G(z) \lor H(z)$›
apply (*rule allI*)
apply (*rule disjI1*)
apply (*rule assms* [*THEN spec*])
done

lemma ‹$\forall\, x.\ \exists\, y.\ x = y$›
apply (*rule allI*)

apply (*rule exI*)
apply (*rule refl*)
done

lemma ‹$\exists\, y.\ \forall\, x.\ x = y$›
apply (*rule exI*)
apply (*rule allI*)
apply (*rule refl*)?
oops

Parallel lifting example.

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

lemma
  assumes ‹$(\exists\, z.\ F(z)) \wedge B$›
  shows ‹$\exists\, z.\ F(z) \wedge B$›
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 ‹$(\exists\, y.\ \forall\, x.\ Q(x,y)) \longrightarrow (\forall\, x.\ \exists\, y.\ Q(x,y))$›
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 $Q \to P$ is provable intuitionistically, so $\neg\neg(Q \to P)$

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

lemma ‹$\neg\ \neg\ (P \wedge Q) \longleftrightarrow \neg\ \neg\ P \wedge \neg\ \neg\ Q$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

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

Double-negation does NOT distribute over disjunction.

lemma ‹$\neg\ \neg\ (P \longrightarrow Q) \longleftrightarrow (\neg\ \neg\ P \longrightarrow \neg\ \neg\ Q)$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

lemma ‹$\neg\ \neg\ \neg\ P \longleftrightarrow \neg\ P$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

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

lemma ‹$(P \longleftrightarrow Q) \longleftrightarrow (Q \longleftrightarrow P)$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

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

lemma
  ‹$(((G \longrightarrow A) \longrightarrow J) \longrightarrow D \longrightarrow E) \longrightarrow (((H \longrightarrow B) \longrightarrow I) \longrightarrow C \longrightarrow J)$
   $\longrightarrow (A \longrightarrow H) \longrightarrow F \longrightarrow G \longrightarrow (((C \longrightarrow B) \longrightarrow I) \longrightarrow D) \longrightarrow (A \longrightarrow C)$
   $\longrightarrow (((F \longrightarrow A) \longrightarrow B) \longrightarrow I) \longrightarrow E$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

Admissibility of the excluded middle for negated formulae

lemma ‹$(P \vee \neg P \longrightarrow \neg Q) \longrightarrow \neg Q$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

The same in a more general form, no ex falso quodlibet

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

7.1 Lemmas for the propositional double-negation translation

lemma ‹$P \longrightarrow \neg\ \neg\ P$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

lemma ‹$\neg\ \neg\ (\neg\ \neg\ P \longrightarrow P)$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

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

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

lemma ‹$((P \longrightarrow Q) \longrightarrow P) \longrightarrow P$›
apply (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)*?*
apply (*rule asm-rl*) — Checks that subgoals remain: proof failed.
oops

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

7.2 de Bruijn formulae

de Bruijn formula with three predicates

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

de Bruijn formula with five predicates

lemma
  ‹$((P \longleftrightarrow Q) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \wedge$
    $((Q \longleftrightarrow R) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \wedge$
    $((R \longleftrightarrow S) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \wedge$
    $((S \longleftrightarrow T) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \wedge$
    $((T \longleftrightarrow P) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T) \longrightarrow P \wedge Q \wedge R \wedge S \wedge T$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

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
  ‹$(\forall x. \exists y. \forall z. p(x) \wedge q(y) \wedge r(z)) \longleftrightarrow$
    $(\forall z. \exists y. \forall x. p(x) \wedge q(y) \wedge r(z))$›
  by (*tactic* ‹*IntPr.best-dup-tac* ***context*** *1*›) — SLOW

Problem 3.1

lemma ‹$\neg (\exists x. \forall y. mem(y,x) \longleftrightarrow \neg mem(x,x))$›
  by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

Problem 4.1: hopeless!

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

7.3 Intuitionistic FOL: propositional problems based on Pelletier.

¬¬1

lemma $‹\neg\ \neg\ ((P \longrightarrow Q) \longleftrightarrow (\neg\ Q \longrightarrow \neg\ P))›$
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬2

lemma $‹\neg\ \neg\ (\neg\ \neg\ P \longleftrightarrow P)›$
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

3

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

¬¬4

lemma $‹\neg\ \neg\ ((\neg\ P \longrightarrow Q) \longleftrightarrow (\neg\ Q \longrightarrow P))›$
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬5

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

¬¬6

lemma $‹\neg\ \neg\ (P \vee \neg\ P)›$
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬7

lemma $‹\neg\ \neg\ (P \vee \neg\ \neg\ \neg\ P)›$
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬8. Peirce's law

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

9

lemma $‹((P \vee Q) \wedge (\neg\ P \vee Q) \wedge (P \vee \neg\ Q)) \longrightarrow \neg\ (\neg\ P \vee \neg\ Q)›$
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

10

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

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

lemma ‹$P \longleftrightarrow P$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬12. Dijkstra's law

lemma ‹$\neg\,\neg\,(((P \longleftrightarrow Q) \longleftrightarrow R) \longleftrightarrow (P \longleftrightarrow (Q \longleftrightarrow R)))$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

lemma ‹$((P \longleftrightarrow Q) \longleftrightarrow R) \longrightarrow \neg\,\neg\,(P \longleftrightarrow (Q \longleftrightarrow R))$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

13. Distributive law

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

¬¬14

lemma ‹$\neg\,\neg\,((P \longleftrightarrow Q) \longleftrightarrow ((Q \vee \neg\,P) \wedge (\neg\,Q \vee P)))$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬15

lemma ‹$\neg\,\neg\,((P \longrightarrow Q) \longleftrightarrow (\neg\,P \vee Q))$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬16

lemma ‹$\neg\,\neg\,((P \longrightarrow Q) \vee (Q \longrightarrow P))$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬17

lemma ‹$\neg\,\neg\,(((P \wedge (Q \longrightarrow R)) \longrightarrow S) \longleftrightarrow ((\neg\,P \vee Q \vee S) \wedge (\neg\,P \vee \neg\,R \vee S)))$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

Dijkstra's "Golden Rule"

lemma ‹$(P \wedge Q) \longleftrightarrow P \longleftrightarrow Q \longleftrightarrow (P \vee Q)$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

8 Examples with quantifiers

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

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

lemma ‹$((\forall\,x.\ P(x)) \longrightarrow Q) \longrightarrow \neg\,(\forall\,x.\ P(x) \wedge \neg\,Q)$›
 by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

lemma ‹$((\forall\,x.\ \neg\,P(x)) \longrightarrow Q) \longrightarrow \neg\,(\forall\,x.\ \neg\,(P(x) \vee Q))$›

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

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

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

8.2 The following are not constructively valid!

The attempt to prove them terminates quickly!

lemma ‹$((\forall x.\ P(x)) \longrightarrow Q) \longrightarrow (\exists x.\ P(x) \longrightarrow Q)$›
apply (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)*?*
apply (*rule asm-rl*) — Checks that subgoals remain: proof failed.
oops

lemma ‹$(P \longrightarrow (\exists x.\ Q(x))) \longrightarrow (\exists x.\ P \longrightarrow Q(x))$›
apply (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)*?*
apply (*rule asm-rl*) — Checks that subgoals remain: proof failed.
oops

lemma ‹$(\forall x.\ P(x) \vee Q) \longrightarrow ((\forall x.\ P(x)) \vee Q)$›
apply (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)*?*
apply (*rule asm-rl*) — Checks that subgoals remain: proof failed.
oops

lemma ‹$(\forall x.\ \neg\neg P(x)) \longrightarrow \neg\neg(\forall x.\ P(x))$›
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 ‹$\exists x.\ Q(x) \longrightarrow (\forall x.\ Q(x))$›
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.

$\neg\neg 18$

lemma ‹$\neg\neg(\exists y.\ \forall x.\ P(y) \longrightarrow P(x))$›
oops — NOT PROVED

$\neg\neg 19$

lemma ‹$\neg\neg(\exists x.\ \forall y\ z.\ (P(y) \longrightarrow Q(z)) \longrightarrow (P(x) \longrightarrow Q(x)))$›

oops — NOT PROVED

20

lemma
‹$(\forall x\ y.\ \exists z.\ \forall w.\ (P(x) \land Q(y) \longrightarrow R(z) \land S(w)))$
$\longrightarrow (\exists x\ y.\ P(x) \land Q(y)) \longrightarrow (\exists z.\ R(z))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

21

lemma ‹$(\exists x.\ P \longrightarrow Q(x)) \land (\exists x.\ Q(x) \longrightarrow P) \longrightarrow \neg\ \neg\ (\exists x.\ P \longleftrightarrow Q(x))$›
oops — NOT PROVED; needs quantifier duplication

22

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

¬¬23

lemma ‹$\neg\ \neg\ ((\forall x.\ P \lor Q(x)) \longleftrightarrow (P \lor (\forall x.\ Q(x))))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

24

lemma
‹$\neg\ (\exists x.\ S(x) \land Q(x)) \land (\forall x.\ P(x) \longrightarrow Q(x) \lor R(x)) \land$
$(\neg\ (\exists x.\ P(x)) \longrightarrow (\exists x.\ Q(x))) \land (\forall x.\ Q(x) \lor R(x) \longrightarrow S(x))$
$\longrightarrow \neg\ \neg\ (\exists x.\ P(x) \land R(x))$›

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

apply (*tactic* ‹*IntPr.safe-tac* ***context***›)
apply (*erule impE*)
apply (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)
apply (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)
done

25

lemma
‹$(\exists x.\ P(x)) \land$
$(\forall x.\ L(x) \longrightarrow \neg\ (M(x) \land R(x))) \land$
$(\forall x.\ P(x) \longrightarrow (M(x) \land L(x))) \land$
$((\forall x.\ P(x) \longrightarrow Q(x)) \lor (\exists x.\ P(x) \land R(x)))$
$\longrightarrow (\exists x.\ Q(x) \land P(x))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬26

lemma
‹$(\neg\ \neg\ (\exists x.\ p(x)) \longleftrightarrow \neg\ \neg\ (\exists x.\ q(x))) \land$
$(\forall x.\ \forall y.\ p(x) \land q(y) \longrightarrow (r(x) \longleftrightarrow s(y)))$

$\longrightarrow ((\forall x.\ p(x) \longrightarrow r(x)) \longleftrightarrow (\forall x.\ q(x) \longrightarrow s(x)))$›
oops — NOT PROVED

27

lemma
‹$(\exists x.\ P(x) \land \neg\ Q(x)) \land$
$(\forall x.\ P(x) \longrightarrow R(x)) \land$
$(\forall x.\ M(x) \land L(x) \longrightarrow P(x)) \land$
$((\exists x.\ R(x) \land \neg\ Q(x)) \longrightarrow (\forall x.\ L(x) \longrightarrow \neg\ R(x)))$
$\longrightarrow (\forall x.\ M(x) \longrightarrow \neg\ L(x))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬28. AMENDED

lemma
‹$(\forall x.\ P(x) \longrightarrow (\forall x.\ Q(x))) \land$
$(\neg\ \neg\ (\forall x.\ Q(x) \lor R(x)) \longrightarrow (\exists x.\ Q(x) \land S(x))) \land$
$(\neg\ \neg\ (\exists x.\ S(x)) \longrightarrow (\forall x.\ L(x) \longrightarrow M(x)))$
$\longrightarrow (\forall x.\ P(x) \land L(x) \longrightarrow M(x))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

29. Essentially the same as Principia Mathematica *11.71

lemma
‹$(\exists x.\ P(x)) \land (\exists y.\ Q(y))$
$\longrightarrow ((\forall x.\ P(x) \longrightarrow R(x)) \land (\forall y.\ Q(y) \longrightarrow S(y)) \longleftrightarrow$
$(\forall x\ y.\ P(x) \land Q(y) \longrightarrow R(x) \land S(y)))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

¬¬30

lemma
‹$(\forall x.\ (P(x) \lor Q(x)) \longrightarrow \neg\ R(x)) \land$
$(\forall x.\ (Q(x) \longrightarrow \neg\ S(x)) \longrightarrow P(x) \land R(x))$
$\longrightarrow (\forall x.\ \neg\ \neg\ S(x))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

31

lemma
‹$\neg\ (\exists x.\ P(x) \land (Q(x) \lor R(x))) \land$
$(\exists x.\ L(x) \land P(x)) \land$
$(\forall x.\ \neg\ R(x) \longrightarrow M(x))$
$\longrightarrow (\exists x.\ L(x) \land M(x))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

32

lemma
‹$(\forall x.\ P(x) \land (Q(x) \lor R(x)) \longrightarrow S(x)) \land$
$(\forall x.\ S(x) \land R(x) \longrightarrow L(x)) \land$
$(\forall x.\ M(x) \longrightarrow R(x))$
$\longrightarrow (\forall x.\ P(x) \land M(x) \longrightarrow L(x))$›

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

¬¬33

lemma
‹$(\forall x.\ \neg\ \neg\ (P(a) \wedge (P(x) \longrightarrow P(b)) \longrightarrow P(c))) \longleftrightarrow$
$(\forall x.\ \neg\ \neg\ ((\neg\ P(a) \vee P(x) \vee P(c)) \wedge (\neg\ P(a) \vee \neg\ P(b) \vee P(c))))$›
apply (*tactic* ‹*IntPr.best-tac* ***context*** *1*›)
done

36

lemma
‹$(\forall x.\ \exists y.\ J(x,y)) \wedge$
$(\forall x.\ \exists y.\ G(x,y)) \wedge$
$(\forall x\ y.\ J(x,y) \vee G(x,y) \longrightarrow (\forall z.\ J(y,z) \vee G(y,z) \longrightarrow H(x,z)))$
$\longrightarrow (\forall x.\ \exists y.\ H(x,y))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

37

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

39

lemma ‹$\neg\ (\exists x.\ \forall y.\ F(y,x) \longleftrightarrow \neg\ F(y,y))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

40. AMENDED

lemma
‹$(\exists y.\ \forall x.\ F(x,y) \longleftrightarrow F(x,x)) \longrightarrow$
$\neg\ (\forall x.\ \exists y.\ \forall z.\ F(z,y) \longleftrightarrow \neg\ F(z,x))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

44

lemma
‹$(\forall x.\ f(x) \longrightarrow$
$(\exists y.\ g(y) \wedge h(x,y) \wedge (\exists y.\ g(y) \wedge \neg\ h(x,y)))) \wedge$
$(\exists x.\ j(x) \wedge (\forall y.\ g(y) \longrightarrow h(x,y)))$
$\longrightarrow (\exists x.\ j(x) \wedge \neg\ f(x))$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

48

lemma ‹$(a = b \vee c = d) \wedge (a = c \vee b = d) \longrightarrow a = d \vee b = c$›
by (*tactic* ‹*IntPr.fast-tac* ***context*** *1*›)

51

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

52

Almost the same as 51.

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

56

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

57

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

60

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

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

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

lemma ‹$(P \wedge Q) \wedge R \longrightarrow P \wedge (Q \wedge R)$›

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

lemma ‹$(P \lor Q) \lor R \longrightarrow P \lor (Q \lor R)$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

distributive laws of $\land$ and $\lor$

lemma ‹$(P \land Q) \lor R \longrightarrow (P \lor R) \land (Q \lor R)$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma ‹$(P \lor R) \land (Q \lor R) \longrightarrow (P \land Q) \lor R$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma ‹$(P \lor Q) \land R \longrightarrow (P \land R) \lor (Q \land R)$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma ‹$(P \land R) \lor (Q \land R) \longrightarrow (P \lor Q) \land R$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

Laws involving implication

lemma ‹$(P \longrightarrow R) \land (Q \longrightarrow R) \longleftrightarrow (P \lor Q \longrightarrow R)$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma ‹$(P \land Q \longrightarrow R) \longleftrightarrow (P \longrightarrow (Q \longrightarrow R))$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma ‹$((P \longrightarrow R) \longrightarrow R) \longrightarrow ((Q \longrightarrow R) \longrightarrow R) \longrightarrow (P \land Q \longrightarrow R) \longrightarrow R$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma ‹$\neg (P \longrightarrow R) \longrightarrow \neg (Q \longrightarrow R) \longrightarrow \neg (P \land Q \longrightarrow R)$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma ‹$(P \longrightarrow Q \land R) \longleftrightarrow (P \longrightarrow Q) \land (P \longrightarrow R)$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

Propositions-as-types

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

— The combinator S
lemma ‹$(P \longrightarrow Q \longrightarrow R) \longrightarrow (P \longrightarrow Q) \longrightarrow (P \longrightarrow R)$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

— Converse is classical
lemma ‹$(P \longrightarrow Q) \lor (P \longrightarrow R) \longrightarrow (P \longrightarrow Q \lor R)$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

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

Schwichtenberg's examples (via T. Nipkow)

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

lemma *stab-to-peirce*:
‹$(((P \longrightarrow R) \longrightarrow R) \longrightarrow P) \longrightarrow (((Q \longrightarrow R) \longrightarrow R) \longrightarrow Q)$
$\longrightarrow ((P \longrightarrow Q) \longrightarrow P) \longrightarrow P$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma *peirce-imp1*:
‹$(((Q \longrightarrow R) \longrightarrow Q) \longrightarrow Q)$
$\longrightarrow (((P \longrightarrow Q) \longrightarrow R) \longrightarrow P \longrightarrow Q) \longrightarrow P \longrightarrow Q$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma *peirce-imp2*: ‹$(((P \longrightarrow R) \longrightarrow P) \longrightarrow P) \longrightarrow ((P \longrightarrow Q \longrightarrow R) \longrightarrow P)$
$\longrightarrow P$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma *mints*: ‹$((((P \longrightarrow Q) \longrightarrow P) \longrightarrow P) \longrightarrow Q) \longrightarrow Q$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma *mints-solovev*: ‹$(P \longrightarrow (Q \longrightarrow R) \longrightarrow Q) \longrightarrow ((P \longrightarrow Q) \longrightarrow R) \longrightarrow R$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma *tatsuta*:
‹$(((P7 \longrightarrow P1) \longrightarrow P10) \longrightarrow P4 \longrightarrow P5)$
$\longrightarrow (((P8 \longrightarrow P2) \longrightarrow P9) \longrightarrow P3 \longrightarrow P10)$
$\longrightarrow (P1 \longrightarrow P8) \longrightarrow P6 \longrightarrow P7$
$\longrightarrow (((P3 \longrightarrow P2) \longrightarrow P9) \longrightarrow P4)$
$\longrightarrow (P1 \longrightarrow P3) \longrightarrow (((P6 \longrightarrow P1) \longrightarrow P2) \longrightarrow P9) \longrightarrow P5$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma *tatsuta1*:
‹$(((P8 \longrightarrow P2) \longrightarrow P9) \longrightarrow P3 \longrightarrow P10)$
$\longrightarrow (((P3 \longrightarrow P2) \longrightarrow P9) \longrightarrow P4)$
$\longrightarrow (((P6 \longrightarrow P1) \longrightarrow P2) \longrightarrow P9)$
$\longrightarrow (((P7 \longrightarrow P1) \longrightarrow P10) \longrightarrow P4 \longrightarrow P5)$
$\longrightarrow (P1 \longrightarrow P3) \longrightarrow (P1 \longrightarrow P8) \longrightarrow P6 \longrightarrow P7 \longrightarrow P5$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

end

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

theory *Quantifiers-Int*
imports *IFOL*

begin

lemma ‹$(\forall x\ y.\ P(x,y)) \longrightarrow (\forall y\ x.\ P(x,y))$›
  by (*tactic IntPr.fast-tac* ***context*** *1*)

lemma ‹$(\exists x\ y.\ P(x,y)) \longrightarrow (\exists y\ x.\ P(x,y))$›
  by (*tactic IntPr.fast-tac* ***context*** *1*)

— Converse is false
lemma ‹$(\forall x.\ P(x)) \lor (\forall x.\ Q(x)) \longrightarrow (\forall x.\ P(x) \lor Q(x))$›
  by (*tactic IntPr.fast-tac* ***context*** *1*)

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

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

Some harder ones

lemma ‹$(\exists x.\ P(x) \lor Q(x)) \longleftrightarrow (\exists x.\ P(x)) \lor (\exists x.\ Q(x))$›
  by (*tactic IntPr.fast-tac* ***context*** *1*)

— Converse is false
lemma ‹$(\exists x.\ P(x) \land Q(x)) \longrightarrow (\exists x.\ P(x)) \land (\exists x.\ Q(x))$›
  by (*tactic IntPr.fast-tac* ***context*** *1*)

Basic test of quantifier reasoning

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

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

The following should fail, as they are false!

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

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

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

schematic-goal ‹$(P(?a) \longrightarrow (\forall x.\ Q(x))) \longrightarrow (\forall x.\ P(x) \longrightarrow Q(x))$›

apply (*tactic IntPr.fast-tac* ***context*** *1*)?
oops

Back to things that are provable ...

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

— An example of why exI should be delayed as long as possible
lemma ‹$(P \longrightarrow (\exists x.\ Q(x))) \wedge P \longrightarrow (\exists x.\ Q(x))$›
by (*tactic IntPr.fast-tac* ***context*** *1*)

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

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

Some slow ones

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

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

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

end

11 Classical Predicate Calculus Problems

theory *Classical*
imports *FOL*
begin

lemma ‹$(P \longrightarrow Q \vee R) \longrightarrow (P \longrightarrow Q) \vee (P \longrightarrow R)$›
by *blast*

11.0.1 If and only if

lemma ‹$(P \longleftrightarrow Q) \longleftrightarrow (Q \longleftrightarrow P)$›
by *blast*

lemma ‹$\neg (P \longleftrightarrow \neg P)$›
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 ‹$(P \longrightarrow Q) \longleftrightarrow (\neg\ Q \longrightarrow \neg\ P)$›
  by *blast*

2

lemma ‹$\neg\ \neg\ P \longleftrightarrow P$›
  by *blast*

3

lemma ‹$\neg\ (P \longrightarrow Q) \longrightarrow (Q \longrightarrow P)$›
  by *blast*

4

lemma ‹$(\neg\ P \longrightarrow Q) \longleftrightarrow (\neg\ Q \longrightarrow P)$›
  by *blast*

5

lemma ‹$((P \vee Q) \longrightarrow (P \vee R)) \longrightarrow (P \vee (Q \longrightarrow R))$›
  by *blast*

6

lemma ‹$P \vee \neg\ P$›
  by *blast*

7

lemma ‹$P \vee \neg\ \neg\ \neg\ P$›
  by *blast*

8. Peirce's law

lemma ‹$((P \longrightarrow Q) \longrightarrow P) \longrightarrow P$›
  by *blast*

9

lemma ‹$((P \vee Q) \wedge (\neg\ P \vee Q) \wedge (P \vee \neg\ Q)) \longrightarrow \neg\ (\neg\ P \vee \neg\ Q)$›
  by *blast*

10

lemma ‹$(Q \longrightarrow R) \wedge (R \longrightarrow P \wedge Q) \wedge (P \longrightarrow Q \vee R) \longrightarrow (P \longleftrightarrow Q)$›
  by *blast*

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

lemma ‹$P \longleftrightarrow P$›
  by *blast*

12. "Dijkstra's law"

lemma ‹$((P \longleftrightarrow Q) \longleftrightarrow R) \longleftrightarrow (P \longleftrightarrow (Q \longleftrightarrow R))$›
  by *blast*

13. Distributive law

lemma ‹$P \vee (Q \wedge R) \longleftrightarrow (P \vee Q) \wedge (P \vee R)$›
  by *blast*

14

lemma ‹$(P \longleftrightarrow Q) \longleftrightarrow ((Q \vee \neg P) \wedge (\neg Q \vee P))$›
  by *blast*

15

lemma ‹$(P \longrightarrow Q) \longleftrightarrow (\neg P \vee Q)$›
  by *blast*

16

lemma ‹$(P \longrightarrow Q) \vee (Q \longrightarrow P)$›
  by *blast*

17

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

11.2 Classical Logic: examples with quantifiers

lemma ‹$(\forall x.\ P(x) \wedge Q(x)) \longleftrightarrow (\forall x.\ P(x)) \wedge (\forall x.\ Q(x))$›
  by *blast*

lemma ‹$(\exists x.\ P \longrightarrow Q(x)) \longleftrightarrow (P \longrightarrow (\exists x.\ Q(x)))$›
  by *blast*

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

lemma ‹$(\forall x.\ P(x)) \vee Q \longleftrightarrow (\forall x.\ P(x) \vee Q)$›
  by *blast*

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

lemma ‹$\neg\ ((\exists\, x.\ \neg\ P(x)) \wedge ((\exists\, x.\ P(x)) \vee (\exists\, x.\ P(x) \wedge Q(x))) \wedge \neg\ (\exists\, x.\ P(x)))$›
by *blast*

11.3 Problems requiring quantifier duplication

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

lemma ‹$(\exists\, x.\ \forall\, y.\ P(x) \longleftrightarrow P(y)) \longrightarrow ((\exists\, x.\ P(x)) \longleftrightarrow (\forall\, y.\ P(y)))$›
by *blast*

Needs multiple instantiation of ALL.

lemma ‹$(\forall\, x.\ P(x) \longrightarrow P(f(x))) \wedge P(d) \longrightarrow P(f(f(f(d))))$›
by *blast*

Needs double instantiation of the quantifier

lemma ‹$\exists\, x.\ P(x) \longrightarrow P(a) \wedge P(b)$›
by *blast*

lemma ‹$\exists\, z.\ P(z) \longrightarrow (\forall\, x.\ P(x))$›
by *blast*

lemma ‹$\exists\, x.\ (\exists\, y.\ P(y)) \longrightarrow P(x)$›
by *blast*

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

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

11.4 Hard examples with quantifiers

18

lemma ‹$\exists\, y.\ \forall\, x.\ P(y) \longrightarrow P(x)$›
by *blast*

19

lemma ‹$\exists\, x.\ \forall\, y\ z.\ (P(y) \longrightarrow Q(z)) \longrightarrow (P(x) \longrightarrow Q(x))$›
by *blast*

20

lemma ‹$(\forall\, x\ y.\ \exists\, z.\ \forall\, w.\ (P(x) \wedge Q(y) \longrightarrow R(z) \wedge S(w)))$
$\longrightarrow (\exists\, x\ y.\ P(x) \wedge Q(y)) \longrightarrow (\exists\, z.\ R(z))$›

by *blast*

21

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

22

lemma ‹$(\forall x.\ P \longleftrightarrow Q(x)) \longrightarrow (P \longleftrightarrow (\forall x.\ Q(x)))$›
by *blast*

23

lemma ‹$(\forall x.\ P \vee Q(x)) \longleftrightarrow (P \vee (\forall x.\ Q(x)))$›
by *blast*

24

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

25

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

26

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

27

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

28. AMENDED

lemma
‹$(\forall x.\ P(x) \longrightarrow (\forall x.\ Q(x))) \land$
$((\forall x.\ Q(x) \lor R(x)) \longrightarrow (\exists x.\ Q(x) \land S(x))) \land$
$((\exists x.\ S(x)) \longrightarrow (\forall x.\ L(x) \longrightarrow M(x)))$
$\longrightarrow (\forall x.\ P(x) \land L(x) \longrightarrow M(x))$›
by *blast*

29. Essentially the same as Principia Mathematica *11.71

lemma
‹$(\exists x.\ P(x)) \land (\exists y.\ Q(y))$
$\longrightarrow ((\forall x.\ P(x) \longrightarrow R(x)) \land (\forall y.\ Q(y) \longrightarrow S(y)) \longleftrightarrow$
$(\forall x\ y.\ P(x) \land Q(y) \longrightarrow R(x) \land S(y)))$›
by *blast*

30

lemma
‹$(\forall x.\ P(x) \lor Q(x) \longrightarrow \neg\ R(x)) \land$
$(\forall x.\ (Q(x) \longrightarrow \neg\ S(x)) \longrightarrow P(x) \land R(x))$
$\longrightarrow (\forall x.\ S(x))$›
by *blast*

31

lemma
‹$\neg\ (\exists x.\ P(x) \land (Q(x) \lor R(x))) \land$
$(\exists x.\ L(x) \land P(x)) \land$
$(\forall x.\ \neg\ R(x) \longrightarrow M(x))$
$\longrightarrow (\exists x.\ L(x) \land M(x))$›
by *blast*

32

lemma
‹$(\forall x.\ P(x) \land (Q(x) \lor R(x)) \longrightarrow S(x)) \land$
$(\forall x.\ S(x) \land R(x) \longrightarrow L(x)) \land$
$(\forall x.\ M(x) \longrightarrow R(x))$
$\longrightarrow (\forall x.\ P(x) \land M(x) \longrightarrow L(x))$›
by *blast*

33

lemma
‹$(\forall x.\ P(a) \land (P(x) \longrightarrow P(b)) \longrightarrow P(c)) \longleftrightarrow$
$(\forall x.\ (\neg\ P(a) \lor P(x) \lor P(c)) \land (\neg\ P(a) \lor \neg\ P(b) \lor P(c)))$›
by *blast*

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

lemma
‹$((\exists x.\ \forall y.\ p(x) \longleftrightarrow p(y)) \longleftrightarrow ((\exists x.\ q(x)) \longleftrightarrow (\forall y.\ p(y)))) \longleftrightarrow$
$((\exists x.\ \forall y.\ q(x) \longleftrightarrow q(y)) \longleftrightarrow ((\exists x.\ p(x)) \longleftrightarrow (\forall y.\ q(y))))$›
by *blast*

35

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

36

lemma
  ‹$(\forall\, x.\ \exists\, y.\ J(x,y)) \wedge$
    $(\forall\, x.\ \exists\, y.\ G(x,y)) \wedge$
    $(\forall\, x\ y.\ J(x,y) \vee G(x,y) \longrightarrow (\forall\, z.\ J(y,z) \vee G(y,z) \longrightarrow H(x,z)))$
  $\longrightarrow (\forall\, x.\ \exists\, y.\ H(x,y))$›
  by *blast*

37

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

38

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

39

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

40. AMENDED

lemma
  ‹$(\exists\, y.\ \forall\, x.\ F(x,y) \longleftrightarrow F(x,x)) \longrightarrow$
    $\neg\ (\forall\, x.\ \exists\, y.\ \forall\, z.\ F(z,y) \longleftrightarrow \neg\ F(z,x))$›
  by *blast*

41

lemma
  ‹$(\forall\, z.\ \exists\, y.\ \forall\, x.\ f(x,y) \longleftrightarrow f(x,z) \wedge \neg\ f(x,x))$
    $\longrightarrow \neg\ (\exists\, z.\ \forall\, x.\ f(x,z))$›
  by *blast*

42

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

43

lemma
‹$(\forall x. \forall y. q(x,y) \longleftrightarrow (\forall z. p(z,x) \longleftrightarrow p(z,y)))$
$\longrightarrow (\forall x. \forall y. q(x,y) \longleftrightarrow q(y,x))$›
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
‹$(\forall x. f(x) \longrightarrow (\exists y. g(y) \wedge h(x,y) \wedge (\exists y. g(y) \wedge \neg h(x,y)))) \wedge$
$(\exists x. j(x) \wedge (\forall y. g(y) \longrightarrow h(x,y)))$
$\longrightarrow (\exists x. j(x) \wedge \neg f(x))$›
by *blast*

45

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

46

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

11.5 Problems (mainly) involving equality or functions

48

lemma ‹$(a = b \vee c = d) \wedge (a = c \vee b = d) \longrightarrow a = d \vee b = c$›
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

$\langle(\exists x\ y{::}'a.\ \forall z.\ z = x \lor z = y) \land P(a) \land P(b) \land a \neq b \longrightarrow (\forall u{::}'a.\ P(u))\rangle$
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 $\langle(\forall x.\ P(a,x) \lor (\forall y.\ P(x,y))) \longrightarrow (\exists x.\ \forall y.\ P(x,y))\rangle$
by *blast*

51

lemma
$\langle(\exists z\ w.\ \forall x\ y.\ P(x,y) \longleftrightarrow (x = z \land y = w)) \longrightarrow$
$(\exists z.\ \forall x.\ \exists w.\ (\forall y.\ P(x,y) \longleftrightarrow y{=}w) \longleftrightarrow x = z)\rangle$
by *blast*

52

Almost the same as 51.

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

55

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

schematic-goal
$\langle lives(agatha) \land lives(butler) \land lives(charles) \land$
$(killed(agatha,agatha) \lor killed(butler,agatha) \lor killed(charles,agatha)) \land$
$(\forall x\ y.\ killed(x,y) \longrightarrow hates(x,y) \land \neg\ richer(x,y)) \land$
$(\forall x.\ hates(agatha,x) \longrightarrow \neg\ hates(charles,x)) \land$
$(hates(agatha,agatha) \land hates(agatha,charles)) \land$
$(\forall x.\ lives(x) \land \neg\ richer(x,agatha) \longrightarrow hates(butler,x)) \land$
$(\forall x.\ hates(agatha,x) \longrightarrow hates(butler,x)) \land$
$(\forall x.\ \neg\ hates(x,agatha) \lor \neg\ hates(x,butler) \lor \neg\ hates(x,charles)) \longrightarrow$
$killed(?who,agatha)\rangle$
by *fast* — MUCH faster than blast

56

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

57

lemma
$\langle P(f(a,b), f(b,c)) \land P(f(b,c), f(a,c)) \land$

$(\forall x\ y\ z.\ P(x,y) \wedge P(y,z) \longrightarrow P(x,z)) \longrightarrow P(f(a,b),\ f(a,c))$›
by *blast*

58 NOT PROVED AUTOMATICALLY

lemma ‹$(\forall x\ y.\ f(x) = g(y)) \longrightarrow (\forall x\ y.\ f(f(x)) = f(g(y)))$›
by (*slow elim*: *subst-context*)

59

lemma ‹$(\forall x.\ P(x) \longleftrightarrow \neg\ P(f(x))) \longrightarrow (\exists x.\ P(x) \wedge \neg\ P(f(x)))$›
by *blast*

60

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

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

lemma
‹$(\forall x.\ p(a) \wedge (p(x) \longrightarrow p(f(x))) \longrightarrow p(f(f(x)))) \longleftrightarrow$
$(\forall x.\ (\neg\ p(a) \vee p(x) \vee p(f(f(x)))) \wedge$
$(\neg\ p(a) \vee \neg\ p(f(x)) \vee p(f(f(x)))))$›
by *blast*

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

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

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

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

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

lemma
‹$((\exists x.\ A(x) \wedge (\forall y.\ C(y) \longrightarrow (\forall z.\ D(x,y,z)))) \longrightarrow$
$(\exists w.\ C(w) \wedge (\forall y.\ C(y) \longrightarrow (\forall z.\ D(w,y,z)))))$
$\wedge$
$(\forall w.\ C(w) \wedge (\forall u.\ C(u) \longrightarrow (\forall v.\ D(w,u,v))) \longrightarrow$
$(\forall y\ z.$
$(C(y) \wedge P(y,z) \longrightarrow Q(w,y,z) \wedge OO(w,g)) \wedge$
$(C(y) \wedge \neg\ P(y,z) \longrightarrow Q(w,y,z) \wedge OO(w,b))))$

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

Challenge found on info-hol.

lemma ‹$\forall x.\ \exists v\ w.\ \forall y\ z.\ P(x) \wedge Q(y) \longrightarrow (P(v) \vee R(w)) \wedge (R(z) \longrightarrow Q(v))$›
by *blast*

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

lemma
‹$(\forall x.\ \mathit{honest}(x) \wedge \mathit{industrious}(x) \longrightarrow \mathit{healthy}(x)) \wedge$
$\neg\ (\exists x.\ \mathit{grocer}(x) \wedge \mathit{healthy}(x)) \wedge$
$(\forall x.\ \mathit{industrious}(x) \wedge \mathit{grocer}(x) \longrightarrow \mathit{honest}(x)) \wedge$
$(\forall x.\ \mathit{cyclist}(x) \longrightarrow \mathit{industrious}(x)) \wedge$
$(\forall x.\ \neg\ \mathit{healthy}(x) \wedge \mathit{cyclist}(x) \longrightarrow \neg\ \mathit{honest}(x))$
$\longrightarrow (\forall x.\ \mathit{grocer}(x) \longrightarrow \neg\ \mathit{cyclist}(x))$›

by *blast*

end

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

theory *Propositional-Cla*
imports *FOL*
begin

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

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

lemma ‹$P \vee Q \longrightarrow Q \vee P$›
by *fast*

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

lemma ‹$(P \wedge Q) \wedge R \longrightarrow P \wedge (Q \wedge R)$›
by *fast*

lemma ‹$(P \vee Q) \vee R \longrightarrow P \vee (Q \vee R)$›
by *fast*

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

lemma ‹$(P \wedge Q) \vee R \longrightarrow (P \vee R) \wedge (Q \vee R)$›
by *fast*

lemma ‹$(P \vee R) \wedge (Q \vee R) \longrightarrow (P \wedge Q) \vee R$›
by *fast*

lemma ‹$(P \vee Q) \wedge R \longrightarrow (P \wedge R) \vee (Q \wedge R)$›
by *fast*

lemma ‹$(P \wedge R) \vee (Q \wedge R) \longrightarrow (P \vee Q) \wedge R$›
by *fast*

Laws involving implication

lemma ‹$(P \longrightarrow R) \wedge (Q \longrightarrow R) \longleftrightarrow (P \vee Q \longrightarrow R)$›
by *fast*

lemma ‹$(P \wedge Q \longrightarrow R) \longleftrightarrow (P \longrightarrow (Q \longrightarrow R))$›
by *fast*

lemma ‹$((P \longrightarrow R) \longrightarrow R) \longrightarrow ((Q \longrightarrow R) \longrightarrow R) \longrightarrow (P \wedge Q \longrightarrow R) \longrightarrow R$›
  by *fast*

lemma ‹$\neg\, (P \longrightarrow R) \longrightarrow \neg\, (Q \longrightarrow R) \longrightarrow \neg\, (P \wedge Q \longrightarrow R)$›
  by *fast*

lemma ‹$(P \longrightarrow Q \wedge R) \longleftrightarrow (P \longrightarrow Q) \wedge (P \longrightarrow R)$›
  by *fast*

Propositions-as-types

lemma ‹$P \longrightarrow (Q \longrightarrow P)$›
  by *fast*

— The combinator S
lemma ‹$(P \longrightarrow Q \longrightarrow R) \longrightarrow (P \longrightarrow Q) \longrightarrow (P \longrightarrow R)$›
  by *fast*

— Converse is classical
lemma ‹$(P \longrightarrow Q) \vee (P \longrightarrow R) \longrightarrow (P \longrightarrow Q \vee R)$›
  by *fast*

lemma ‹$(P \longrightarrow Q) \longrightarrow (\neg\, Q \longrightarrow \neg\, P)$›
  by *fast*

Schwichtenberg's examples (via T. Nipkow)

lemma *stab-imp*: ‹$(((Q \longrightarrow R) \longrightarrow R) \longrightarrow Q) \longrightarrow (((P \longrightarrow Q) \longrightarrow R) \longrightarrow R)$
$\longrightarrow P \longrightarrow Q$›
  by *fast*

lemma *stab-to-peirce*:
  ‹$(((P \longrightarrow R) \longrightarrow R) \longrightarrow P) \longrightarrow (((Q \longrightarrow R) \longrightarrow R) \longrightarrow Q)$
    $\longrightarrow ((P \longrightarrow Q) \longrightarrow P) \longrightarrow P$›
  by *fast*

lemma *peirce-imp1*:
  ‹$(((Q \longrightarrow R) \longrightarrow Q) \longrightarrow Q)$
    $\longrightarrow (((P \longrightarrow Q) \longrightarrow R) \longrightarrow P \longrightarrow Q) \longrightarrow P \longrightarrow Q$›
  by *fast*

lemma *peirce-imp2*: ‹$(((P \longrightarrow R) \longrightarrow P) \longrightarrow P) \longrightarrow ((P \longrightarrow Q \longrightarrow R) \longrightarrow P)$
$\longrightarrow P$›
  by *fast*

lemma *mints*: ‹$((((P \longrightarrow Q) \longrightarrow P) \longrightarrow P) \longrightarrow Q) \longrightarrow Q$›
  by *fast*

lemma *mints-solovev*: ‹$(P \longrightarrow (Q \longrightarrow R) \longrightarrow Q) \longrightarrow ((P \longrightarrow Q) \longrightarrow R) \longrightarrow R$›

by *fast*

lemma *tatsuta*:
‹$(((P7 \longrightarrow P1) \longrightarrow P10) \longrightarrow P4 \longrightarrow P5)$
$\longrightarrow (((P8 \longrightarrow P2) \longrightarrow P9) \longrightarrow P3 \longrightarrow P10)$
$\longrightarrow (P1 \longrightarrow P8) \longrightarrow P6 \longrightarrow P7$
$\longrightarrow (((P3 \longrightarrow P2) \longrightarrow P9) \longrightarrow P4)$
$\longrightarrow (P1 \longrightarrow P3) \longrightarrow (((P6 \longrightarrow P1) \longrightarrow P2) \longrightarrow P9) \longrightarrow P5$›
by *fast*

lemma *tatsuta1*:
‹$(((P8 \longrightarrow P2) \longrightarrow P9) \longrightarrow P3 \longrightarrow P10)$
$\longrightarrow (((P3 \longrightarrow P2) \longrightarrow P9) \longrightarrow P4)$
$\longrightarrow (((P6 \longrightarrow P1) \longrightarrow P2) \longrightarrow P9)$
$\longrightarrow (((P7 \longrightarrow P1) \longrightarrow P10) \longrightarrow P4 \longrightarrow P5)$
$\longrightarrow (P1 \longrightarrow P3) \longrightarrow (P1 \longrightarrow P8) \longrightarrow P6 \longrightarrow P7 \longrightarrow P5$›
by *fast*

end

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

theory *Quantifiers-Cla*
imports *FOL*
begin

lemma ‹$(\forall x\ y.\ P(x,y)) \longrightarrow (\forall y\ x.\ P(x,y))$›
by *fast*

lemma ‹$(\exists x\ y.\ P(x,y)) \longrightarrow (\exists y\ x.\ P(x,y))$›
by *fast*

Converse is false.

lemma ‹$(\forall x.\ P(x)) \lor (\forall x.\ Q(x)) \longrightarrow (\forall x.\ P(x) \lor Q(x))$›
by *fast*

lemma ‹$(\forall x.\ P \longrightarrow Q(x)) \longleftrightarrow (P \longrightarrow (\forall x.\ Q(x)))$›
by *fast*

lemma ‹$(\forall x.\ P(x) \longrightarrow Q) \longleftrightarrow ((\exists x.\ P(x)) \longrightarrow Q)$›
by *fast*

Some harder ones.

lemma ‹$(\exists x.\ P(x) \lor Q(x)) \longleftrightarrow (\exists x.\ P(x)) \lor (\exists x.\ Q(x))$›
by *fast*

— Converse is false.
lemma ‹$(\exists x.\ P(x) \wedge Q(x)) \longrightarrow (\exists x.\ P(x)) \wedge (\exists x.\ Q(x))$›
  by *fast*

Basic test of quantifier reasoning.

lemma ‹$(\exists y.\ \forall x.\ Q(x,y)) \longrightarrow (\forall x.\ \exists y.\ Q(x,y))$›
  by *fast*

lemma ‹$(\forall x.\ Q(x)) \longrightarrow (\exists x.\ Q(x))$›
  by *fast*

The following should fail, as they are false!

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

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

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

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

Back to things that are provable . . .

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

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

lemma ‹$(P \longrightarrow (\exists x.\ Q(x))) \wedge P \longrightarrow (\exists x.\ Q(x))$›
  by *fast*

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

lemma ‹$(\forall x.\ Q(x)) \longrightarrow (\exists x.\ Q(x))$›
  by *fast*

Some slow ones

Principia Mathematica *11.53

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

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

lemma ‹$(\exists\, y.\ \forall\, x.\ P(x) \longrightarrow Q(x,y)) \longrightarrow (\forall\, x.\ P(x) \longrightarrow (\exists\, y.\ Q(x,y)))$›
  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*:
  ‹$\neg\ (P \wedge Q) \longleftrightarrow \neg\ P \vee \neg\ Q$›
  ‹$\neg\ (P \vee Q) \longleftrightarrow \neg\ P \wedge \neg\ Q$›
  ‹$\neg\ \neg\ P \longleftrightarrow P$›
  by *blast*+

lemma *demorgans2*:
  ‹$\bigwedge P.\ \neg\ (\forall\, x.\ P(x)) \longleftrightarrow (\exists\, x.\ \neg\ P(x))$›
  ‹$\bigwedge P.\ \neg\ (\exists\, x.\ P(x)) \longleftrightarrow (\forall\, x.\ \neg\ P(x))$›
  by *blast*+

lemmas *demorgans* = *demorgans1 demorgans2*

lemma *nnf-simps*:
  ‹$(P \longrightarrow Q) \longleftrightarrow (\neg\ P \vee Q)$›
  ‹$\neg\ (P \longrightarrow Q) \longleftrightarrow (P \wedge \neg\ Q)$›
  ‹$(P \longleftrightarrow Q) \longleftrightarrow (\neg\ P \vee Q) \wedge (\neg\ Q \vee P)$›
  ‹$\neg\ (P \longleftrightarrow Q) \longleftrightarrow (P \vee Q) \wedge (\neg\ P \vee \neg\ Q)$›
  by *blast*+

13.1.2 Pushing in the existential quantifiers

lemma *ex-simps*:
  ‹$(\exists\, x.\ P) \longleftrightarrow P$›
  ‹$\bigwedge P\ Q.\ (\exists\, x.\ P(x) \wedge Q) \longleftrightarrow (\exists\, x.\ P(x)) \wedge Q$›
  ‹$\bigwedge P\ Q.\ (\exists\, x.\ P \wedge Q(x)) \longleftrightarrow P \wedge (\exists\, x.\ Q(x))$›

‹$\bigwedge P\ Q.\ (\exists\, x.\ P(x) \lor Q(x)) \longleftrightarrow (\exists\, x.\ P(x)) \lor (\exists\, x.\ Q(x))$›
‹$\bigwedge P\ Q.\ (\exists\, x.\ P(x) \lor Q) \longleftrightarrow (\exists\, x.\ P(x)) \lor Q$›
‹$\bigwedge P\ Q.\ (\exists\, x.\ P \lor Q(x)) \longleftrightarrow P \lor (\exists\, x.\ Q(x))$›
by *blast+*

13.1.3 Pushing in the universal quantifiers

lemma *all-simps*:
‹$(\forall\, x.\ P) \longleftrightarrow P$›
‹$\bigwedge P\ Q.\ (\forall\, x.\ P(x) \land Q(x)) \longleftrightarrow (\forall\, x.\ P(x)) \land (\forall\, x.\ Q(x))$›
‹$\bigwedge P\ Q.\ (\forall\, x.\ P(x) \land Q) \longleftrightarrow (\forall\, x.\ P(x)) \land Q$›
‹$\bigwedge P\ Q.\ (\forall\, x.\ P \land Q(x)) \longleftrightarrow P \land (\forall\, x.\ Q(x))$›
‹$\bigwedge P\ Q.\ (\forall\, x.\ P(x) \lor Q) \longleftrightarrow (\forall\, x.\ P(x)) \lor Q$›
‹$\bigwedge P\ Q.\ (\forall\, x.\ P \lor Q(x)) \longleftrightarrow P \lor (\forall\, x.\ Q(x))$›
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 ‹$if(P,Q,R) \equiv P \land Q \lor \neg P \land R$›

lemma *ifI*: ‹$[\![P \Longrightarrow Q;\ \neg P \Longrightarrow R]\!] \Longrightarrow if(P,Q,R)$›
unfolding *if-def* **by** *blast*

lemma *ifE*: ‹$[\![if(P,Q,R);\ [\![P;\ Q]\!] \Longrightarrow S;\ [\![\neg P;\ R]\!] \Longrightarrow S]\!] \Longrightarrow S$›
unfolding *if-def* **by** *blast*

lemma *if-commute*: ‹$if(P,\ if(Q,A,B),\ if(Q,C,D)) \longleftrightarrow if(Q,\ if(P,A,C),\ if(P,B,D))$›
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*: ‹$if(P, if(Q,A,B), if(Q,C,D)) \longleftrightarrow if(Q, if(P,A,C), if(P,B,D))$›
  by *blast*

lemma ‹$if(if(P,Q,R), A, B) \longleftrightarrow if(P, if(Q,A,B), if(R,A,B))$›
  by *blast*

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

lemma ‹$if(if(P,Q,R), A, B) \longleftrightarrow if(P, if(Q,A,B), if(R,A,B))$›
  unfolding *if-def* **by** *blast*

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

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

end