Pure_I

Jørgen Villadsen

May 2021

Contents

```
theory Pure-I imports Pure begin

typedecl bool

judgment Trueprop :: ‹bool ⇒ prop› (‹-›)

axiomatization Imp (infixr ‹⟶› 3)
  where Imp-I [intro]: ‹(p ⟹ q) ⟹ p ⟶ q›
    and Imp-E [elim]: ‹p ⟶ q ⟹ p ⟹ q›

axiomatization Dis (infixr ‹∨› 4)
  where Dis-E [elim]: ‹p ∨ q ⟹ (p ⟹ r) ⟹ (q ⟹ r) ⟹ r›
    and Dis-I1 [intro]: ‹p ⟹ p ∨ q›
    and Dis-I2 [intro]: ‹q ⟹ p ∨ q›

axiomatization Con (infixr ‹∧› 5)
  where Con-I [intro]: ‹p ⟹ q ⟹ p ∧ q›
    and Con-E1 [elim]: ‹p ∧ q ⟹ p›
    and Con-E2 [elim]: ‹p ∧ q ⟹ q›

axiomatization Falsity (‹⊥›)
  where Falsity-E [elim]: ‹⊥ ⟹ p›

definition Truth (‹⊤›) where ‹⊤ ≡ ⊥ ⟶ ⊥›

theorem Truth-I [intro]: ‹⊤›
  unfolding Truth-def ..

definition Neg (‹¬ -› [6] 6) where ‹¬ p ≡ p ⟶ ⊥›

theorem Neg-I [intro]: ‹(p ⟹ ⊥) ⟹ ¬ p›
  unfolding Neg-def ..

theorem Neg-E [elim]: ‹¬ p ⟹ p ⟹ q›
  unfolding Neg-def
proof –
  assume ‹p ⟶ ⊥› and ‹p›
  then have ‹⊥› ..
  then show ‹q› ..
qed

definition Iff (infixr ‹⟷› 3) where ‹p ⟷ q ≡ (p ⟶ q) ∧ (q ⟶ p)›

theorem Iff-I [intro]: ‹(p ⟹ q) ⟹ (q ⟹ p) ⟹ p ⟷ q›
  unfolding Iff-def
proof –
  assume ‹p ⟹ q› and ‹q ⟹ p›
```

```
  from ‹p ⟹ q› have ‹p ⟶ q› ..
  from ‹q ⟹ p› have ‹q ⟶ p› ..
  from ‹p ⟶ q› and ‹q ⟶ p› show ‹(p ⟶ q) ∧ (q ⟶ p)› ..
qed

theorem Iff-E1 [elim]: ‹p ⟷ q ⟹ p ⟹ q›
  unfolding Iff-def
proof –
  assume ‹(p ⟶ q) ∧ (q ⟶ p)›
  then have ‹p ⟶ q› ..
  then show ‹p ⟹ q› ..
qed

theorem Iff-E2 [elim]: ‹p ⟷ q ⟹ q ⟹ p›
  unfolding Iff-def
proof –
  assume ‹(p ⟶ q) ∧ (q ⟶ p)›
  then have ‹q ⟶ p› ..
  then show ‹q ⟹ p› ..
qed

proposition ‹p ⟶ ¬ ¬ p›
proof
  assume ‹p›
  show ‹¬ ¬ p›
  proof
    assume ‹¬ p›
    from ‹¬ p› and ‹p› show ‹⊥› ..
  qed
qed

proposition ‹(p ⟶ q) ∧ ¬ q ⟶ ¬ p›
proof
  assume ‹(p ⟶ q) ∧ ¬ q›
  show ‹¬ p›
  proof
    assume ‹p›
    from ‹(p ⟶ q) ∧ ¬ q› have ‹p ⟶ q› ..
    from ‹p ⟶ q› and ‹p› have ‹q› ..
    from ‹(p ⟶ q) ∧ ¬ q› have ‹¬ q› ..
    from ‹¬ q› and ‹q› show ‹⊥› ..
  qed
qed

proposition ‹(p ⟷ q) ⟷ q ⟷ p›
proof
  assume ‹p ⟷ q›
  show ‹q ⟷ p›
  proof
    from ‹p ⟷ q› show ‹q ⟹ p› ..
  next
    from ‹p ⟷ q› show ‹p ⟹ q› ..
  qed
next
  assume ‹q ⟷ p›
  show ‹p ⟷ q›
  proof
    from ‹q ⟷ p› show ‹p ⟹ q› ..
  next
    from ‹q ⟷ p› show ‹q ⟹ p› ..
```

qed
qed

end