
Ejercicio 1 [1 punto] Formalizar las siguientes frases usando los símbolos $L(x)$ para "x es una línea", $P(x)$ para "x es un punto", $R(x,y)$ para "x e y son paralelas", $O(x,y)$ para "x e y son ortogonales", $E(x,y)$ para "x pertenece a y".

- Dos líneas ortogonales tiene un punto común (es decir, un punto que pertenece a ambas líneas).
- Las rectas paralelas no tienen puntos comunes.
- Por cada punto exterior a una línea pasa una paralela a dicha línea.

Solución:

- $\forall x \forall y (L(x) \land L(y) \land O(x,y) \to \exists z (P(z) \land E(z,x) \land E(z,y)))$
- $\forall x \forall y (L(x) \land L(y) \land P(x,y) \to \neg \exists z (P(z) \land E(z,x) \land E(z,y)))$
- $\forall x \forall y (P(x) \land L(y) \land \neg E(x,y) \to \exists z (L(z) \land R(z,y) \land E(x,z)))$

Ejercicio 2 [1 punto] Sean F y G dos fórmulas tales que $\models F \to G$. Demostrar o refutar las siguientes proposiciones:

1. Si F es satisfacible, entonces G es satisfacible.
2. Si G es satisfacible, entonces F es satisfacible.

Solución:

Solución del apartado 1: Es cierto ya que si F es satisfacible, existe una interpretación I que es un modelo de F; es decir $I(F) = 1$. Por la hipótesis, $I(F \to G) = 1$. Por tanto, $I(G) = 1$, I es un modelo de G y G es satisfacible.

Solución del apartado 2: Es falso, un contraejemplo es $F = p \land \neg p$ y $G = q$.

Ejercicio 3 [1 punto] Demostrar que $\exists z R(z,z)$ no es consecuencia lógica de $\forall x \exists y R(x,y)$.

Solución:

Sea $U = \{1,2\}$ e $I(R) = \{(1,2),(2,1)\}$. Entonces, I es modelo de $\forall x \exists y R(x,y)$ pero lo es de $\exists z R(z,z)$.

Ejercicio 4 [2 puntos] Demostrar por deducción natural en Isabelle que la fórmula

$$p \to (\neg q \lor r)$$

es consecuencia lógica del conjunto

$$\{p \land q \to r \lor s, q \to \neg s\}.$$

```
lemma ejercicio_4a:
  assumes "p ∧ q ⟶ r ∨ s"
          "q ⟶ ¬s"
  shows "p ⟶ (¬q ∨ r)"
proof
  assume "p"
  have "¬q ∨ q" by (rule excluded_middle)
  then show "¬q ∨ r"
  proof
    assume "¬q"
    then show "¬q ∨ r" by (rule disjI1)
  next
    assume "q"
    with ‘p‘ have "p ∧ q" by (rule conjI)
    with assms(1) have "r ∨ s" by (rule mp)
    then show "¬q ∨ r"
    proof
      assume "r"
      then show "¬q ∨ r" by (rule disjI2)
    next
      assume "s"
      have "¬s" using assms(2) ‘q‘ by (rule mp)
      then show "¬q ∨ r" using ‘s‘ by (rule notE)
    qed
  qed
qed
```

```
lemma ejercicio_4b:
  "⟦p ∧ q ⟶ r ∨ s;
    q ⟶ ¬s ⟧
   ⟹ p ⟶ (¬q ∨ r)"
  apply (rule impI)
    (* ⟦p ∧ q ⟶ r ∨ s; q ⟶ ¬ s; p⟧ ⟹ ¬ q ∨ r ) *)
  apply (cut_tac P=q in excluded_middle)
    (* ⟦p ∧ q ⟶ r ∨ s; q ⟶ ¬ s; p; ¬ q ∨ q⟧ ⟹ ¬ q ∨ r ) *)
  apply (erule disjE)
    (* p ∧ q ⟶ r ∨ s; q ⟶ ¬ s; p; ¬ q⟧ ⟹ ¬ q ∨ r *)
    (* ⟦p ∧ q ⟶ r ∨ s; q ⟶ ¬ s; p; q⟧ ⟹ ¬ q ∨ r *)
  apply (rule disjI1, assumption)
    (* ⟦p ∧ q ⟶ r ∨ s; q ⟶ ¬ s; p; q⟧ ⟹ ¬ q ∨ r *)
  apply (drule mp, assumption)
    (* ⟦p ∧ q ⟶ r ∨ s; p; q; ¬ s⟧ ⟹ ¬ q ∨ r *)
  apply (drule mp)
    (* ⟦p; q; ¬ s⟧ ⟹ p ∧ q *)
    (* ⟦p; q; ¬ s; r ∨ s⟧ ⟹ ¬ q ∨ r *)
   apply (rule conjI, assumption+)
    (* ⟦p; q; ¬ s; r ∨ s⟧ ⟹ ¬ q ∨ r  *)
   apply (erule disjE)
    (* ⟦p; q; ¬ s; r⟧ ⟹ ¬ q ∨ r *)
    (* ⟦p; q; ¬ s; s⟧ ⟹ ¬ q ∨ r *)
   apply (rule disjI2, assumption)
    (* ⟦p; q; ¬ s; s⟧ ⟹ ¬ q ∨ r *)
   apply (erule notE, assumption)
    (* *)
  done
```

Ejercicio 5 [2.5 puntos] Demostrar de forma estructurada, usando Isar:

```
lemma ejercicio_5:
  assumes "∀x. ∀y. ∀z. P(x,y) ∧ P(y,z) → R(x,z)"
          "∀x. ∃y. P(x,y)"
  shows   "∀x. ∃y. R(x,y)"
```

```
lemma ejercicio_5a:
  assumes "∀x. ∀y. ∀z. P(x,y) ∧ P(y,z) ⟶ R(x,z)"
          "∀x. ∃y. P(x,y)"
  shows "∀x. ∃y. R(x,y)"
proof
  fix a
  show "∃y. R(a,y)"
  proof -
    have "∃y. P(a,y)" using assms(2) by (rule allE)
    then obtain b where "P(a,b)" by (rule exE)
    have "∃y. P(b,y)" using assms(2) by (rule allE)
    then obtain c where "P(b,c)" by (rule exE)
    with ‘P(a,b)‘ have "P(a,b) ∧ P(b,c)" by (rule conjI)
    have  "∀y. ∀z. P(a,y) ∧ P(y,z) ⟶ R(a,z)" using assms(1)
      by (rule allE)
    then have "∀z. P(a,b) ∧ P(b,z) ⟶ R(a,z)" by (rule allE)
    then have "P(a,b) ∧ P(b,c) ⟶ R(a,c)" by (rule allE)
    then have "R(a,c)" using ‘P(a,b) ∧ P(b,c)‘ by (rule mp)
    then show  "∃y. R(a,y)" by (rule exI)
  qed
qed
```

```
lemma ejercicio_5b:
  "⟦∀x. ∀y. ∀z. P(x,y) ∧ P(y,z) ⟶ R(x,z);
   ∀x. ∃y. P(x,y)⟧
  ⟹ ∀x. ∃y. R(x,y)"
  apply (rule allI)
  (* ⋀x. ⟦∀x y z. P (x, y) ∧ P (y, z) ⟶ R (x, z); ∀x. ∃y. P (x, y)⟧ ⟹ ∃y. R (x, y) *)
  apply (erule_tac x = x in allE)
  (* ⋀x. ⟦∀x. ∃y. P (x, y); ∀y z. P (x, y) ∧ P (y, z) ⟶ R (x, z)⟧ ⟹ ∃y. R (x, y) *)
  apply (frule_tac x = x in spec)
  (*  ⋀x. ⟦∀x. ∃y. P (x, y); ∀y z. P (x, y) ∧ P (y, z) ⟶ R (x, z); ∃y. P (x, y)⟧ ⟹ ∃y. R (x,
  apply (erule exE)
  (* ⋀x y. ⟦∀x. ∃y. P (x, y); ∀y z. P (x, y) ∧ P (y, z) ⟶ R (x, z); P (x, y)⟧ ⟹ ∃y. R (x, y)
  apply (erule_tac x = y in allE)
  (* ⋀x y. ⟦∀y z. P (x, y) ∧ P (y, z) ⟶ R (x, z); P (x, y); ∃ya. P (y, ya)⟧ ⟹ ∃y. R (x, y) *)
  apply (erule exE)
  (* ⋀x y ya. ⟦∀y z. P (x,y) ∧ P (y,z) ⟶ R (x,z); P (x, y); P (y, ya)⟧ ⟹ ∃y. R (x, y) *)
  apply (erule_tac x = y in allE)
  (* ⋀x y ya. ⟦P (x, y); P (y, ya); ∀z. P (x,y) ∧ P (y,z) ⟶ R (x,z)⟧ ⟹ ∃y. R (x, y) *)
  apply (erule_tac x = ya in allE)
  (* ⋀x y ya. ⟦P (x, y); P (y, ya); P (x, y) ∧ P (y, ya) ⟶ R (x, ya)⟧ ⟹ ∃y. R (x, y) *)
  apply (rule_tac x = ya in exI)
  (* ⋀x y ya. ⟦P (x, y); P (y, ya); P (x, y) ∧ P (y, ya) ⟶ R (x, ya)⟧ ⟹ R (x, ya) *)
  apply (erule mp)
  (* ⋀x y ya. ⟦P (x, y); P (y, ya)⟧ ⟹ P (x, y) ∧ P (y, ya) *)
   apply (rule conjI, assumption+)
  (* *)
  done
```

Ejercicio 6 [2.5 puntos] Demostrar usando tácticas

```
lemma ejercicio_6:
  "[∀x. Q(x) → ¬R(x);
   ∀x. P (x) →  Q(x) ∨ S(x);
   ∃x. P (x) ∧ R(x) ]
   ⊢ ∃x. P(x) ∧ S(x)"
```

```
lemma ejercicio_6b:
  assumes "∀x. Q(x) ⟶ ¬R(x)"
          "∀x. P (x) ⟶  Q(x) ∨ S(x)"
          "∃x. P (x) ∧ R(x)"
  shows "∃x. P(x) ∧ S(x)"
proof-
  from assms(3) obtain a where "P(a) ∧ R(a)" by (rule exE)
  then have "P(a)" by (rule conjunct1)
  have "P (a) ⟶  Q(a) ∨ S(a)" using assms(2) by (rule allE)
  then have "Q(a) ∨ S(a)" using ‘P(a)‘ by (rule mp)
  then have "P(a) ∧ S(a)"
  proof
    assume "Q(a)"
    have "R(a)" using ‘P(a) ∧ R(a)‘ by (rule conjunct2)
    have "Q(a) ⟶ ¬R(a)" using assms(1) by (rule allE)
    then have "¬R(a)" using ‘Q(a)‘ by (rule mp)
    then show  "P(a) ∧ S(a)" using ‘R(a)‘ by (rule notE)
  next
    assume "S(a)"
    with ‘P(a)‘ show  "P(a) ∧ S(a)" by (rule conjI)
  qed
  then show "∃x. P(x) ∧ S(x)" by (rule exI)
qed
```

```
lemma ejercicio_6b:
  "⟦∀x. Q(x) ⟶ ¬R(x);
    ∀x. P (x) ⟶  Q(x) ∨ S(x);
    ∃x. P (x) ∧ R(x) ⟧ ⟹
   ∃x. P(x) ∧ S(x)"
  apply (erule exE)
    (* ⋀x. ⟦∀x. Q x ⟶ ¬ R x; ∀x. P x ⟶ Q x ∨ S x; P x ∧ R x⟧ ⟹ ∃x. P x ∧ S x *)
  apply (erule_tac x = x in allE)+
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x ⟶ Q x ∨ S x⟧ ⟹ ∃x. P x ∧ S x *)
  apply (rule_tac x = x in exI)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x ⟶ Q x ∨ S x⟧ ⟹ P x ∧ S x *)
  apply (rule conjI)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x ⟶ Q x ∨ S x⟧ ⟹ P x *)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x ⟶ Q x ∨ S x⟧ ⟹ S x *)
   apply (rule conjunct1, assumption)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x ⟶ Q x ∨ S x⟧ ⟹ S x *)
  apply (frule conjunct1)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x ⟶ Q x ∨ S x; P x⟧ ⟹ S x *)
  apply (drule mp, assumption)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x; Q x ∨ S x⟧ ⟹ S x *)
  apply (erule disjE)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x; Q x⟧ ⟹ S x *)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x; S x⟧ ⟹ S x *)
   prefer 2
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x; S x⟧ ⟹ S x *)
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x; Q x⟧ ⟹ S x *)
   apply assumption
    (* ⋀x. ⟦P x ∧ R x; Q x ⟶ ¬ R x; P x; Q x⟧ ⟹ S x *)
   apply (drule conjunct2)
    (* ⋀x. ⟦Q x ⟶ ¬ R x; P x; Q x; R x⟧ ⟹ S x*)
  apply (drule mp, assumption)
    (* ⋀x. ⟦P x; Q x; R x; ¬ R x⟧ ⟹ S x*)
  apply (erule notE, assumption)
    (* *)
  done
```