Preguntas de final
Lógica y computabilidad

Juan Vanecek

Febrero 2016

1. Computabilidad

1. Probar que la clase de funciones computables es una clase PRC.

Solution:

Idea: una clase de función es PRC si están las iniciales y el resto se forma por composición o recursión. Lo que hay que demostrar es que las computables cumplen esto.

Demostración: Las funciones iniciales son computables, porque

- $n(x) = 0$ se computa con el programa vacío.
- $s(x) = x + 1$ se computa con el programa $\{Y \leftarrow X + 1\}$.
- $u_i(x_1, \ldots, x_n) = x_i$ se computa con el programa $\{Y \leftarrow X_i\}$.

Falta ver que la clase de funciones computables está cerrada por (a) composición; y (b) recursión primitiva.

(a) Si h se obtiene a partir de las funciones (parciales) computables $f, g_1, \ldots, g_k$ por composición, entonces h es computable. Y esto es cierto porque el siguiente programa computa h:

$$
\begin{array}{l}
Z_1 \leftarrow g_1(X_1, \ldots, X_n) \\
\vdots \\
Z_k \leftarrow g_k(X_1, \ldots, X_n) \\
Y \leftarrow f(Z_1, \ldots, Z_k)
\end{array}
$$

Y si $f, g_1, \ldots, g_k$ son computables, entonces h es computable.

(b) Si h se obtiene a partir de g por recursión primitiva y g es computable entonces h es computable.

En efecto, el siguiente programa computa h:

$$
\begin{array}{ll}
 & Y \leftarrow k \\
[A] & \text{IF } X = 0 \text{ GOTO } E \\
 & Y \leftarrow g(Z, Y) \\
 & Z \leftarrow Z + 1 \\
 & X \leftarrow X - 1 \\
 & \text{GOTO } A
\end{array}
$$

Y por lo tanto, si g es computable, h también.

2. Probar que una función es primitiva recursiva si pertenece a toda clase PRC.

Solution:

Demostración: Supongamos que existe f p.r. y una clase C PRC.

Por definición de p.r., f se puee obtener a partir de funciones inciales y por un número finito de aplicaciones de composición y recursión primitiva. Es decir, que existe una lista de funciones $f_1, f_2, \ldots, f_n$ tal que:

- f_i o bien es una función inicial (y por lo tanto está en C) o bien se obtiene por composición o recursión primitiva a partir de funciones f_j, con $j < i$ (y por lo tanto está en C).

- $f_n = f$

Por consiguiente (o más bien por inducción), todas las funciones de la lista están en C. En particular $f_n = f \in C$.

3. Sean A y B dos conjuntos c.e. Demostrar que $A \cup B$ y $A \cap B$ son c.e. ¿Es $A \backslash B$ c.e.?

Solution:

Definición: Un conjunto A es c.e. cuando existe una función parcial computable $g : \mathbb{N} \to \mathbb{N}$ tal que:

$$A = \{x : g(x) \downarrow\} = \text{dom}(g)$$

(En otras palabras, podemos decidir algorítmicamente si un elemento sí pertenece a A, aunque para elementos que no pertenecen a A el algoritmo se indefine)

Demostración: $A \cap B$

Sean p y q los números de los programas tales que

$$A = \{x : \Phi_p(x)\} \quad B = \{x : \Phi_q(x)\}$$

Si podemos encontrar un programa que se defina con las mismas entradas que definen a A y B, entonces estamos.

Sea R el siguiente programa:

$Y \leftarrow \Phi_p(x)$
$Y \leftarrow \Phi_q(x)$

Notemos que $\Psi_R(x) \downarrow$ si y solo si $\Phi_p(x) \downarrow$ y $\Phi_q(x) \downarrow$.

En efecto, $A \cap B$ es c.e. porque

$$A \cap B = \{x : \Psi_R(x) \downarrow\}$$

Demostración: $A \cup B$

Tengo que encontrar un programa que me permita decidir si un elemento pertenece a A o a B.

Para ello vamos a usar la función $STP^{(n)}(x_1, \ldots, x_n, e, t)$ que es $TRUE$ si y solo si el programa con número e termina en t o menos pasos con las entradas $x_1, \ldots, x_n$.

Definimos R como:

$[C]$ IF $STP^{(1)}(X, p, T)$ GOTO E
IF $STP^{(1)}(X, q, T)$ GOTO E
$T \leftarrow T + 1$
GOTO C

Notemos que este programa chequea *al mismo tiempo* si un elemento pertenece al dominio de Φ_p o de Φ_q, ejecutando paso a paso los programas p y q.

En efecto $\Psi_R(x) \downarrow$ si y solo si $\Phi_p(x) \downarrow$ o $\Phi_q(x) \downarrow$:

$$A \cup B = \{x : \Psi_R(x) \downarrow\}$$

Demostración: $A \backslash B$

$A \backslash B$ NO es c.e.. Sea A el conjunto de todos los programas, y B sólo de aquellos que paran cuando se les da como entrada su número de programa. $A \backslash B$ es entonces el conjunto de programas que no se detienen cuando se les da su número de programa como entrada.

Supongamos que $A \backslash B$ fuera c.e. Pero por definición, un conjunto es c.e. si podemos decidir algorítmicamente si un elemento pertenece o no. Es decir que $\text{HALT}(X, X)$ sería computable. Absurdo!

4. Seas A y B conjuntos de una clase PRC C. Entonces $A \cup B$, $A \cap B$, y $\bar{A}$ están en C.

5. Definir conjunto de índices. Enunciar y demostrar el Teorema de Rice.

Solution:

Definición: un conjunto de naturales A es un conjunto de índices si existe una clase de funciones $\mathbb{N} \to \mathbb{N}$ parciales computables C tal que $A = \{x : \Phi_x \in C\}$.

Teorema de Rice: Si A es un conjunto de índices tal que $\emptyset \neq A \neq \mathbb{N}$, A no es computable.

Demostración: Supongamos la clase de funciones C tal que $A = \{x : \Phi_x \in C\}$ computable. Sea $f \in C$ y $g \neq C$ funciones parciales computables. Sea $h : \mathbb{N}^2 \to \mathbb{N}$ la siguiente función parcial computable:

$$h(t, x) = \begin{cases} g(x) & \text{si } t \in A \\ f(x) & \text{si no} \end{cases}$$

Por el teorema de la Recursión, existe e tal que $\Phi_e(x) = h(e, x)$. Luego,

- Si $e \notin A \Rightarrow h(e, x) = f(x) \Rightarrow \Phi_e = h = f \Rightarrow$ como $f \in C, \Phi_e \in C \Rightarrow e \in A$. Absurdo!
- Si $e \in A \Rightarrow h(e, x) = g(x) \Rightarrow \Phi_e = h = g \Rightarrow$ como $g \notin C, \Phi_e \notin C \Rightarrow e \notin A$. Absurdo!

6. Enunciar y demostrar el Teorema de la Recursión.

Solution:

Teorema de la Recursión: Si $g : \mathbb{N}^{n+1} \to \mathbb{N}$ es parcial computable, existe un e tal que

$$\Phi_e(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Demostración: Sea S_n^1 la función del teorema del Parámetro, tal que:

$$\Phi_y^{(n+1)}(x_1, \ldots, x_n, u) = \Phi_{S_n^1(u,y)}^{(n)}(x_1, \ldots, x_n)$$

La función $(x_1, \ldots, x_n, v) \mapsto g(S_n^1(v, v), x_1, \ldots, x_n)$ es parcial computable, de modo que existe d tal que:

$$\begin{aligned} g(S_n^1(v, v), x_1, \ldots, x_n) &= \Phi_d^{(n+1)}(x_1, \ldots, x_n, v) \\ &= \Phi_{S_n^1(v,d)}^{(n)}(x_1, \ldots, x_n) \end{aligned}$$

Como d está fijo, y v es variable, elegimos $v = d$ y $e = S_n^1(d, d)$. Luego

$$\begin{aligned} g(S_n^1(v, v), x_1, \ldots, x_n) &= \Phi_{S_n^1(v,v)}^{(n)}(x_1, \ldots, x_n) \\ g(e, x_1, \ldots, x_n) &= \Phi_e(x_1, \ldots, x_n) \end{aligned}$$

7. Enunciar y demostrar el Teorema de Punto Fijo.

Solution:

Teorema del Punto Fijo: Si $f : \mathbb{N} \to \mathbb{N}$ es computable, existe e tal que $\Phi_{f(e)} = \Phi_e$.

Demostración: Considerar la función $g : \mathbb{N}^2 \to \mathbb{N}$, $g(z, x) = \Phi_{f(z)}(x)$.

Aplicando el Teorema de la Recursión, existe un e tal que para todo x,

$$\Phi_e(x) = g(e, x) = \Phi_{f(e)}(x)$$

8. Enunciar Halt y demostrar que no es computable. (Con cualquiera de las demostraciones vistas.)

Solution:

El problema Halt consiste en tratar de determinar si un programa P con número de programa y, y con entrada x termina.

Y ello se define con la función:

$$HALT(x, y) = \begin{cases} 1 & \text{si } \Phi_y^{(1)}(x) \downarrow \\ 0 & \text{si } \Phi_y^{(1)}(x) \uparrow \end{cases}$$

Supongamos que HALT es computable. Construimos el siguiente programa Q:

$[A]$ IF HALT$(X, X) = 1$ GOTO A

Supongamos que $\#Q = e$. Entonces

$$\Phi_e(x) = \begin{cases} \uparrow & \text{si } HALT(x, x) = 1 \\ 0 & \text{si no} \end{cases}$$

Entonces

$$\text{HALT}(x, e) = 1 \quad \Leftrightarrow \quad \Phi_e(x) \downarrow \quad \Leftrightarrow \quad \text{HALT}(x, x) \neq 1$$

Si bien e está fijo, x es variable. Llegamos a un absurdo con $x = e$.

9. Exhibir una función computable que no sea p.r. y demostrarlo.

Solution:

Sea la función $\tilde{\Phi}_e^{(n)}(x_1, \ldots, x_n)$ computable que simula a la e-ésima función p.r. con entrada $x_1, \ldots, x_n$.

Sea $g : \mathbb{N} \to \mathbb{N}$, definida como $g(x) = \tilde{\Phi}_x^{(1)}(x)$. Demostremos que g es computable pero no p.r.

Demostración: Claramente g es computable porque $\tilde{\Phi}$ lo es.

Supongamos que es p.r. $\Rightarrow f(x) = g(x) + 1$ es p.r. por composición $\Rightarrow \exists e$ tal que $\tilde{\Phi}_e = f$ por ser f p.r. $\Rightarrow \tilde{\Phi}_e(x) = \tilde{\Phi}_x(x) + 1$.

e está fijo pero x es variable. Así que instanciando $x = e$, llegamos a que $\tilde{\Phi}_e(e) = \tilde{\Phi}_e(e) + 1$. Absurdo!

10. (Mat.) Probar que la función $\tau : \mathrm{N} \to \mathrm{N}$ que asigna a cada número natural positivo n el número de divisores positivos n y $\tau(0) = 0$, es una función recursiva primitiva.

Solution:

Una función es primitiva recursiva si se obtiene a través de las funciones iniciales por composición y/o recursión en finitos pasos.

Sea $f : \mathbb{N}^2 \to \mathbb{N}$ tal que $f(a, b)$ devuelve la cantidad de divisores positivos desde 0 hasta b. Con a y b naturales.

Queremos que $\tau(n) = f(n, n) \forall n \in \mathbb{N}$ definiendo a f de la siguiente forma:

$$\begin{aligned} f(n, 0) &= n(n) \\ f(n, m+1) &= g(n, m, f(n, m)) \end{aligned}$$

Con

$$g(a, b, c) = \begin{cases} suc(u_3^3(a, b, c)) & \text{si } P \\ u_3^3(a, b, c) & \text{si } \neg P \end{cases}$$

Donde

$$P = u_1^3(a, b, c) \mid suc(u_2^3(a, b, c))$$

Veamos que:

A) f cumple con el esquema de recursión primitivo.

B) $n(n)$ es la función inicial nula aplicada a n.

C) El predicado P usa la función proyección y la función "divide a" ($|$) que es primitiva recursiva.

D) La función g es una división de casos disjuntos y usa las funciones iniciales de proyección y sucesor.

Por todo lo anterior, la función $f(n, m)$ es primitiva recursiva (con n y m naturales).

Falta ver que $\tau(n) = f(n, n)$. Probamos por inducción en el segundo parámetro.

- Caso base:

  $\tau(0) = 0$

  $f(0, 0) = n(0) = 0$

- Paso inductivo:

  Se cumple la hipótesis inductiva $f(n, m)$ devuelve los divisiores de n desde 0 hasta m. Ahora queremos ver para $m + 1$: $f(n, m + 1) = g(n, m, f(n, m))$. Abrimos los dos casos según el dominio de g:

  1. Caso $n \mid (m + 1)$: $g(n, m, f(n, m)) = f(n, m) + 1$.
     Entonces por H.I. al dividir $m+1$ incremento en 1 a lo ya calculado en el paso recursivo anterior y éste calculo correctamente hasta m. Queda $f(n, m + 1) = f(n, m) + 1$.
  2. Caso $\neg(n \mid (m + 1))$: $g(n, m, f(n, m)) = f(n, m)$.
     Entonces por H.I. al no dividir $m + 1$ no sumo nada a lo ya calculado en el paso recursivo anterior y éste calcula correctamente hasta m. Queda $f(n, m+1) = f(n, m)$.

  Por lo tanto, $\tau(n) = f(n, n) \forall n \in \mathbb{N}$.

11. (Mat.) Dar un ejemplo de una función no computable de una variable tal que la imagen tenga tres elementos.

Solution: Sea f:

$$f(x) = \begin{cases} Halt(x,x) & \text{si } x \neq 0 \\ 2 & \text{si } x = 0 \end{cases}$$

La $Img(f) = \{0, 1, 2\}$. Tiene exactamente 3 elementos.

Supongamos que f es computable, entonces $\exists P$ programa que computa f.

Si $f(x) = 2$, entonces $x = 0$.

Si $f(x) = 0(Halt(x, x))$ o $f(x) = 1(\neg Halt(x, x))$, si y sólo si $\varphi_P(x) \downarrow$.

Sea $e = \#P$. Tomemos el caso particular: $x = e$. $H(e, e)$ determina si el programa e con entrada e termina o no.

Como vimos en las teóricas, f está resolviendo *halting problem*. Absurdo! pues *Halt* no es computable. Vino de suponer que $f(x)$ es computable.

Entonces $f(x)$ no es computable.

12. Probar que TOT no es c.e. ni co-c.e.

Solution:

Def. TOT= $\{e : \Phi_e(x)$ es total$\}$, y Φ_e es total sii $(\forall x)\Phi_e(x) \downarrow$.

Def. Un conjunto A es computablemente enumerable si y sólo si existe una función g parcialmente computable, $g : \mathbb{N} \to \mathbb{N}$ tal que $A = \{x : g(x) \downarrow\} = \text{Dom}(g)$.

Def. Un conjunto A es co-c.e. sii $\overline{A}$ es c.e.

Demostración: TOT no c.e.

Supongamos que es c.e. $\Rightarrow \exists g$ parcial computable tal que TOT=Dom(g).

Existe además e, tal que $\Phi_e(x) = \Phi_{g(x)}(x) + 1$. Notar que esto se define para todo x. Luego Φ_e es total $\Rightarrow e \in$TOT$\Rightarrow \exists u : g(u) = e \Rightarrow \Phi_{g(u)}(x) = \Phi_{g(u)}(x) + 1$.

Absurdo si $x = u$, y esto vino de suponer que TOT es c.e.

Demostración: TOT no co-c.e.

Supongamos que $\overline{\text{TOT}}$ c.e., existe entonces d tal que $\overline{\text{TOT}}$ =Dom(Φ_d).

Definimos el siguiente programa P:

$[C]$ IF $\text{STP}^{(1)}(X, d, T) = 1$ GOTO E
$T \leftarrow T + 1$
GOTO C

Tenemos

$$\Psi_P(x, d) = g(x, d) = \begin{cases} \uparrow & \text{si } \Phi_d \text{ es total} \\ 0 & \text{si no} \end{cases}$$

Por el Teorema de la Recursión: sea e tal que $\Phi_e(x) = g(x, e)$.

- Φ_e es total $\Rightarrow g(e, d) \uparrow$ para todo $y \Rightarrow \Phi_e(x) \uparrow$ para todo $x \Rightarrow \Phi_e$ no es total.
- Φ_e no es total $\Rightarrow g(e, d) = 0$ para todo $y \Rightarrow \Phi_e(x) = 0$ para todo $x \Rightarrow \Phi_e$ es total.

13. Probar que todo conjunto r.e. infinito contiene un conjunto recursivo infinito.

Solution:

Def. Un conjunto A es recursivamente enumerable si y sólo si existe una función f computable, $f : \mathbb{N} \to \mathbb{N}$ tal que $A = \{x : f(x) \downarrow\} = \text{Dom}(f)$.

Demostración: Primero probamos que si $B = \{f(n) : n \in \mathbb{N}\}$, con f una función estrictamente creciente, parcialmente computable y total, entonces B es recursivo. Para verlo, basta ver el siguiente programa:

$[A]$ IF $f(Z) = X$ GOTO B
IF $f(Z) > X$ GOTO E
$Z \leftarrow Z + 1$
GOTO A
$[B]$ $Y \leftarrow 1$
GOTO E

computa Ψ_B. Este programa termina siempre, así que B es recursivo.

Ahora, sea A r.e. e infinito. Tenemos una función p.r. g que nos va dando los elementos de A.

Definimos $f(0) = g(0)$ y suponiendo definida $f(k)$ para todo $k < n$, definimos $f(n) = g(\text{mín}_z(g(z) > f(n-1)))$.

Esta minimización es propia, pues A es infinito. Entonces, siempre podemos encontrar un elemento A mayor que todos los que tengamos. Esto nos da una f parcialmente computable, total, y estrictamente creciente. Si tomamos $B = \{f(n) : n \in \mathbb{N}\}$, tenemos que B es recursivo y que $B \subseteq A$.

14. Probar que si A es computable entonces es c.e. ¿Vale la vuelta? Justificar.

Solution: Sea P_A un programa para [la función característica de] A.

Consideremos el siguiente programa P:

$[C]$ IF $P_A(X) = 0$ GOTO C

Tenemos que:

$$\Psi_A(x) = \begin{cases} 0 & \text{si } x \in A \\ \uparrow & \text{si no} \end{cases}$$

Y por lo tanto $A = \{x : \Psi_P(X) \downarrow\}$.

La vuelta no es cierta. Lo demostramos con un contraejemplo.

Definimos $W_n = \{x : \Phi_n(x)\}$ = el dominio del n-ésimo programa, y a $K = \{n : n \in W_n\}$. Observemos que $n \in W_n$ sii $\Phi_n(n) \downarrow$ sii HALT(n, n). K es (a) c.e. pero (b) no computable.

(a) K es c.e. porque $K = \{x : g(n) \downarrow\}$

(b) Supongamos que K fuera computable. Entonces $\overline{K}$ también lo sería. Luego existe un e tal que $\overline{K} = W_e$. Por lo tanto

$$e \in K \Leftrightarrow e \in W_e \Leftrightarrow e \in \overline{K}$$

15. Probar que un conjunto $A \subseteq \mathbb{N}$ es computable sii A y $\overline{A}$ son c.e.

Solution:

$\Rightarrow$) Si A es computable entonces $\overline{A}$ es computable.

$\Leftarrow$) Supongamos que A y $\overline{A}$ son c.e.

$$A = \{x : \Phi_p(x)\downarrow\} \quad \overline{A} = \{x : \Phi_q(x)\downarrow\}$$

Consideremos el programa P:

$[C]$ IF $\text{STP}^{(1)}(X, p, T) = 1$ GOTO F
IF $\text{STP}^{(1)}(X, q, T) = 1$ GOTO E
$T \leftarrow T + 1$
GOTO C
$[F]$ $Y \leftarrow 1$

Para cada x, vale que $x \in A$ o bien $a \in \overline{A}$. Entonces Ψ_P computa A.

2. Logica proposicional

1. Definir consistente, maximal consistente, y demostrar que φ es teorema de Γ si y sólo si φ pertenece a Γ.

Solution:

Def. Un conjunto $\Gamma \subseteq$FORM es consistente si no existe $\varphi \in$FORM tal que $\Gamma \vdash \varphi$ y $\Gamma \nvdash \varphi$.

Def. Un conjunto $\Gamma \subseteq$FORM es maximal consistente en SP si es consistente y para toda fórmula φ, $\varphi \in \Gamma$ o existe ψ tal que $\Gamma \cup \{\varphi\} \vdash \psi$ y $\Gamma \cup \{\varphi\} \nvdash \psi$.

2. Definir conjunto maximal consistente y probar que si Γ es m.c. entonces:

   a) $\varphi \in \Gamma$ o (exclusivo) $\neg\varphi \in \Gamma$

   b) $\varphi \in \Gamma$ sii $\Gamma \vdash \varphi$

Solution:

a) No puede ser que φ y $\neg\varphi$ estén en Γ porque sería inconsistente.

Supongamos que ninguna está. Como Γ es maximal y por proposición ($\Gamma \cup \{\neg\varphi\}$ es inconsistente sii $\Gamma \vdash \varphi$):

- $\Gamma \cup \{\varphi\}$ es inconsistente $\Rightarrow \Gamma \vdash \neg\varphi$
- $\Gamma \cup \{\neg\varphi\}$ es inconsistente $\Rightarrow \Gamma \vdash \varphi$

Luego Γ es inconsistente.

3. Enunciar y demostrar el Lema de Lindenbaum para la lógica proposicional.

Solution:

Lema de Lindenbaum: Si $\Gamma \subseteq$ FORM es consistente, existe Γ' maximal consistente tal que $\Gamma \subseteq \Gamma'$

Demostración:

Hay que encontrar el Γ' tal que es m.c. y contiene a Γ.

Enumeremos todas las fórmulas $\varphi_1, \varphi_2, \ldots$ y definimos:

$$\Gamma_i = \begin{cases} \Gamma & \text{si } i = 0 \\ \Gamma_{i-1} \cup \{\varphi_i\} & \text{si } \Gamma_{i-1} \cup \{\varphi_i\} \text{ es consistente} \\ \Gamma_{i-1} & \text{si no} \end{cases}$$

Luego definimos $\Gamma' = \bigcup_{i \geq 0} \Gamma_i$. Y terminamos. Γ' es (a) consistente, (b) maximal y (c) contiene a Γ como queríamos.

(a) Si no fuera consistente, existiría ψ tal que: $\Gamma' \vdash \psi$ y $\Gamma' \nvdash \psi$. En ambas derivaciones aparece únicamente $\{\gamma_1, \ldots, \gamma_k\} \subseteq \Gamma'$. Sea j el mínimo índice tal que $\{\gamma_1, \ldots, \gamma_k\} \subseteq \Gamma_j$. Entonces Γ_j es inconsistente, y esto es absurdo por que Γ' se construye de Γ_i consistentes.

(b) Es maximal, porque supongamos un $\varphi \notin \Gamma'$. Debe existir un n tal que $\varphi_n = \varphi$. Y como $\varphi_n \notin \Gamma_n$ (porque sino estaría en Γ'), entonces $\Gamma_{n-1} \cup \{\varphi_n\}$ es inconsistente. Luego $\Gamma' \cup \{\varphi_n\}$ es inconsistente.

4. Enunciar y demostrar el teorema de la Deducción.

Solution:

Teorema de la Deducción: Si $\Gamma \cup \{\varphi\} \vdash \psi$ entonces $\Gamma \vdash \varphi \to \psi$.

Demostración: Por inducción en la demostración de $\Gamma \cup \{\varphi\} \vdash \psi$.

Supongamos que $\varphi_1, \ldots, \varphi_n (= \psi)$ es una derivación de ψ a partir de $\Gamma \cup \{\varphi\}$.

Caso base ($n = 1$): la derivación es una sóla fórmula ($\varphi_1 = \psi$). Queremos ver que $\Gamma \vdash \varphi \to \psi$. Hay 3 posibilidades:

a) ψ es un axioma de SP. Usando el mecanismo deductivo:
   1. ψ (por ser axioma)
   2. $\psi \to (\varphi \to \psi)$ (por SP1)
   3. $\psi \to \varphi$ (por MP 1,2)

b) $\psi \in \Gamma$
   1. ψ (por $\psi \in \Gamma$)
   2. $\psi \to (\varphi \to \psi)$ (por SP1)
   3. $\psi \to \varphi$ (por MP 1,2)

c) $\psi = \varphi$. Vale porque $\varphi \to \varphi$ (visto en clase)

Paso inductivo: HI: *"para toda derivación ψ' a partir de $\Gamma \cup \{\varphi\}$ de longitud menor a n vale $\Gamma \vdash \varphi \to \psi'$"*. Queremos ver que $\Gamma \vdash \varphi \to \psi$. Hay 4 posibilidades:

a) ψ es un axioma de SP: igual que caso base.

b) $\psi \in \Gamma$: igual que en caso base.

c) $\psi = \varphi$: igual que en caso base.

d) ψ se infiere por MP de φ_i y φ_j ($i, j < n$):
   1. Sin pérdida de generalidad: $\varphi_j = \varphi_i \to \psi$.
   2. $\Gamma \cup \{\varphi\} \vdash \varphi_i$ y la derivación tiene longitud menor a n. Por HI: $\Gamma \vdash \varphi \to \varphi_i$
   3. $\Gamma \cup \{\varphi\} \vdash \varphi_j$ y la derivación tiene longitud menor a n. Por HI: $\Gamma \vdash \varphi \to \varphi_j$ (i.e. $\Gamma \vdash \varphi \to (\varphi_j \to \psi)$.
   4. (Por SP2) $\vdash (\varphi \to (\varphi_i \to \psi)) \to ((\varphi \to \varphi_i) \to (\varphi \to \psi))$.
   5. (Por MP 2 veces) $\Gamma \vdash \varphi \to \psi$.

5. Usando (el teorema de) correctitud de la lógica proposicional probar que si (un conjunto de fórmulas) Γ es satisfactible entonces Γ es consistente.

Solution:

Supongamos que Γ es satisfacible, y sin embargo es inconsistente.

Por lo tanto, por satisfacibilidad existe v tal que $v \vDash \Gamma$.

Por inconsistencia, existe φ tal que $\Gamma \vdash \varphi$ y $\Gamma \nvdash \varphi$.

Por correctitud de SP, $\Gamma \vdash \varphi \Rightarrow \Gamma \vDash \varphi$ y $\Gamma \nvdash \varphi \Rightarrow \Gamma \nvDash \varphi$.

Luego $\Gamma \vDash \varphi$ y $\Gamma \nvDash \varphi$, y por último $v \vDash \varphi$ y $v \nvDash \varphi$. Absurdo!

6. Demostrar que $\Gamma \cup \{\neg\varphi\}$ es inconsistente si y sólo si φ es consecuencia sintáctica de Γ.

Solution:

($\Leftarrow$) Por propiedad: si $\Gamma \vdash \varphi \Rightarrow \Gamma \cup \{\neg\varphi\} \vdash \varphi$.

Por Teorema de la Deducción: $\Gamma \cup \{\neg\varphi\} \vdash \neg\varphi$.

Por lo tanto, $\Gamma \cup \{\neg\varphi\}$ es inconsistente.

($\Rightarrow$) Por hipótesis: existe ψ tal que $\Gamma \cup \{\neg\varphi\} \vdash \psi$ y $\Gamma \cup \{\neg\varphi\} \vdash \neg\psi$.

Por Teorema de la Deducción: $\Gamma \vdash \neg\varphi \to \psi$ y $\Gamma \vdash \neg\varphi \to \neg\psi$.

Se puede ver que: $\vdash (\neg\varphi \to \psi) \to ((\neg\varphi \to \neg\psi) \to \varphi)$.

Por MP 2 veces: $\Gamma \vdash \varphi$.

7. Demostrar la correctitud de SP: Si φ es teorema de la teoría Γ, es válido en toda interpretación de Γ ($\Gamma \vdash \varphi \Rightarrow \Gamma \vDash \varphi$).

Solution:

Queremos ver que $\Gamma \vdash \varphi \Rightarrow \Gamma \vDash \varphi$. Supongamos que vale el antecedente. Es decir, existe una derivación $\varphi_1, \ldots, \varphi_n$ tal que $\varphi_n = \varphi$ y (a) φ_i es un axioma o (b) $\varphi_i \in \Gamma$ o (c) φ_i es una consecuencia inmediata de $\varphi_k, \varphi_l, k, l < i$.

Demostramos que $\Gamma \vDash \varphi$ por inducción en n (la longitud de la derivación):

$P(n) =$ *"si* $\varphi_1, \ldots, \varphi_n = \varphi$ *es una derivación de* φ *a partir de* Γ *entonces* $v \vDash \Gamma \Rightarrow v \vDash \varphi$*"*

Caso base ($n = 1$): Supongamos v tal que $v \vDash \Gamma$. Queremos ver que $v \vDash \Gamma \Rightarrow v \vDash \varphi$. Hay 2 posibilidades: o bien φ es axioma de SP, o pertenece a Γ. En ambos casos $v \vDash \varphi$.

Paso inductivo: Supongamos v tal que $v \vDash \Gamma$. Supongamos que vale $P(m)$ para todo $m \leq n$. Qvq vale $P(n+1)$. Supongamos $\varphi_1, \ldots, \varphi_n, \varphi_{n+1} = \varphi$ es una derivación de φ a partir de Γ. Hay 3 posibilidades:

- φ es axioma de SP: igual que en caso base.
- $\varphi \in \Gamma$: igual que en caso base.
- φ es consecuencia inmediata de φ_i y $\varphi_j = \varphi_i \to \varphi$ $(i, j, \leq n)$. Por HI ($P(i)$ y $P(j)$), sabemos que $v \vDash \varphi_i$ y $v \vDash \varphi_i \to \varphi$. Entonces necesariamente $v \vDash \varphi$.

8. (Mat.) Sea *Var* el conjunto de variables proposicionales de la lógica proposicional. Probar que si $f : Var \to \{0, 1\}$ es una función entonces existe una valuación $v : F \to \{0, 1\}$ que extiende a f, donde F es el conjunto de las fórmulas de la lógica proposicional.

Solution:

Hay que probar a) la existencia y b) la unicidad de que existe una única valuación v que extiende a la función f.

a) La existencia se prueba por inducción en la complejidad de la fórmula definiendo en cada caso cómo se evalúa.

Caso base: sea a tal que $comp(a) = 0$, entonces a es una variable proposicional, y por lo tanto $f(a)$ esá definida. Vale entonces $v(a) = f(a)$.

Paso inductivo: supongamos válido $comp(a) = n$, siendo n la complejidad de a. Veamos si $comp(a) = n + 1$.

- Si $a = \neg b$, entonces $comp(b) = n$. Por H.I., $v(b)$ está definido. Queda que $v(a) = 1 - v(b)$.
- Si $a = b * c$, con $* \in \{\wedge, \vee, \to\}$. Entonces $comp(b)$ y $comp(b)$ son menores a $n + 1$. Por H.I. $v(b)$ y $v(c)$ están definidos. Por lo tanto:

$$v(a) = \text{mín}(v(b), v(c)) \quad \text{si } a = b \wedge c$$
$$v(a) = \text{máx}(v(b), v(c)) \quad \text{si } a = b \vee c$$
$$v(a) = \text{máx}(1 - v(b), v(c)) \quad \text{si } a = b \to c$$

La función v queda definido para toda fórmula de cualquier complejidad.

b) La unicidad se prueba suponiendo que existiese otra función de valuación w que extiende a f.

Consideremos el conjunto $I = \{a \in FORM | v(a) = w(a)\}$.

Como w también extiende a f, I contiene a todas las variables proposicionales. Como v y w son ambas valuaciones, I es cerrado por los conectivos por los que $FORM \subseteq I$. Es decir, $v(P) = w(P)$ para toda fórmula P.

Usando el teorema de que si un subconjunto S de A es cerrado por los conectivos y S contiene a todas las variables proposicionales entonces S contiene a todas las fórmulas.

(Unicidad basado en el apunte de lógica de Roberto Cignoli y Guillermo Martínez.)

3. Lógica de primer orden

1. Demostrar que si Γ (teoría de primer orden) tiene modelos arbitrariamente grandes, tiene un modelo infinito.

> **Solution:** Definimos (en el lenguaje con sólo la igualdad):
>
> $$\varphi_i = \text{“hay al menos } i \text{ elementos”} \quad \forall i \geq 2$$
>
> Por hipótesis, todo subconjunto finito $\Gamma \cup \{\varphi_i | i \geq 2\}$ tiene modelo.
>
> Por Compacidad, $\Gamma \cup \{\varphi_i | i \geq 2\}$ tiene algún modelo $\mathcal{M}$.
>
> Por lo tanto, $\mathcal{M}$ tiene que ser infinito.

2. Dado L un lenguaje de primer orden con igualdad. Decidir si las siguientes afirmaciones son verdaderas o falsas.
   a) Existe un conjunto Γ tal que $\Gamma \models$ A sii A tiene universo infinito. [Otra version: Existe Γ tal que A es modelo de Γ sii A es un modelo infinito.]
   b) Existe un conjunto Γ tal que $\Gamma \models$ A sii A tiene universo finito. [Otra version: Existe Γ tal que A es modelo de Γ sii A es un modelo finito.]
   c) El conjunto Γ del ítem 1 necesariamente es infinito.

3. Dar un conjunto de fórmulas Γ tal que Γ es válida si y sólo si el modelo que la satisface es infinito. ¿Existe una fórmula φ tal que φ es válida si y sólo si el modelo que la satisface es finitio?¿Por qué?

4. Sea $\mathcal{L} = \{0, S, <, +, \cdot\}$ con igualdad y sea $\mathcal{N} = \langle \mathcal{N}; 0, S, <, +, \cdot \rangle$ una $\mathcal{L}$-estructura de primer orden con la interpretación usual. Mostrar que existe un modelo de $\mathcal{N}$ en donde valen todas las verdades de $\mathcal{N}$ pero en donde existe un elemento inalcanzable (desde el 0, usando la función sucesor S).

> **Solution:**
> Sea:
>
> $$\begin{aligned} \text{el lenguaje } \mathcal{L} &= \{0, S, <, +, \cdot\} \text{ con igualdad} \\ \text{la estructura } \mathcal{N} &= \langle \mathbb{N}; 0, S, <, +, \cdot \rangle \text{ con la interpretación usual} \\ \text{Teo}(\mathcal{L}) &= \{\varphi \in \text{FORM}(\mathcal{L}) : \varphi \text{ es sentencia y } \mathcal{L} \vDash \varphi\} \end{aligned}$$
>
> Expandimos el lenguaje con una nueva constante c y definimos:
>
> $$\Gamma = \{0 < c, S(0) < c, S(S(0)) < c, S(S(S(0))) < c, \dots\}$$
>
> Para $\Gamma \cup \text{Teo}(\mathcal{N})$, cada subconjunto finitio tiene modelo; por compacidad tiene modelo; y por Lowenheim-Skolem tiene un modelo numerable:
>
> $$\mathcal{M} = \langle M; 0^{\mathcal{M}}, S^{\mathcal{M}}, <^{\mathcal{M}}, +^{\mathcal{M}}, \cdot^{\mathcal{M}}, c^{\mathcal{M}} \rangle$$
>
> Sea $\mathcal{M}'$ la restricción de $\mathcal{M}$ al lenguaje original $\mathcal{L}$. Veamos que $\mathcal{N} \vDash \varphi$ sii $\mathcal{M}' \vDash \varphi$ para toda sentencia $\varphi \in \text{FORM}(\mathcal{L})$:
>
> - $\mathcal{N} \vDash \varphi \Rightarrow \varphi \in \text{Teo}(\mathcal{N}) \Rightarrow \mathcal{M} \vDash \varphi \Rightarrow \mathcal{M}' \vDash \varphi$.
> - $\mathcal{N} \nvDash \varphi \Rightarrow \neg\varphi \in \text{Teo}(\mathcal{N}) \Rightarrow \mathcal{M} \vDash \neg\varphi \Rightarrow \mathcal{M}' \nvDash \varphi$.
>
> $\mathcal{N}$ y $\mathcal{M}$ no son isomorfos: $c^{\mathcal{M}}$ es inalcanzable en $\mathcal{M}'$.

5. Probar que existen modelos no estándar de la aritmética en los que hay un elemento inalcanzable.

> **Solution:** Vale la respuesta de la pregunta 4.

6. Mostrar un modelo no estándar de los naturales.

> **Solution:** Vale la respuesta de la pregunta 4.

7. Verdadero o Falso (Justificar):

$$\exists M \vDash \varphi \Leftrightarrow M \text{ es infinito.}$$

8. Enumerar (y explicar muy brevemente) los pasos de la demostración de completitud en Primer Orden y mostrar el modelo canónico utilizado en la demostración.

> **Solution:**
>
> *Teorema de Completitud:* Si $\Gamma \vDash \varphi$ entonces $\Gamma \vdash \varphi$.
>
> *Demostración:* Sea $\mathcal{L}$ un lenguaje fijo. Sea $\Gamma \subseteq \text{FORM}(\mathcal{L})$ consistente. Queremos construir un modelo canónico $\mathcal{B}$ y una valuación v de $\mathcal{B}$ tal que:
>
> $$\mathcal{B} \vDash \varphi[v] \quad \forall \varphi \in \Gamma$$
>
> Para ello:
>
> 1. Se expande $\mathcal{L}$ a $\mathcal{L}'$ con nuevas constantes $\mathcal{C}$ que no aparecen en $\mathcal{L}$. Γ sigue siendo consistente bajo $\mathcal{L}'$.
> 2. Se agrega un conjunto $\Theta \subseteq \text{FORM}(\mathcal{L}')$ de testigos a Γ. $\Gamma \cup \Theta$ sigue siendo consistente.
> 3. Se aplica Lindenbaum para $\Gamma \cup \Theta$ y se obtiene $\Delta \supseteq \Gamma \cup \Theta$ maximal consistente.
> 4. Se construye el modelo canónico $\mathcal{A}$ y valuación v (para el lenguaje $\mathcal{L}'$) tal que $\mathcal{A} \vDash \varphi[v]$ sii $\varphi \in \Delta$, para toda $\varphi \in \text{FORM}(\mathcal{L}')$.
> 5. Se define $\mathcal{B}$ como la restricción de $\mathcal{A}$ a $\mathcal{L}$, concluyendo que Γ es satisfacible.

9. Sea $\mathcal{L} = \{c, f\}$ un lenguaje de primer orden con igualdad donde c es un símbolo de constante y f un símbolo de función unaria.
   1. Definir $\varphi, \psi \in \text{FORM}(\mathcal{L})$ tal que:
      - φ sea verdadera sii c no pertenece al rango de f.
      - ψ sea verdadera sii f es inyectiva.
   2. Para $\theta = (\varphi \wedge \psi)$, probar que si $A \vDash \theta$ entonces A es infinito.
   3. Definir $\theta' \in \text{FORM}(\mathcal{L})$ tal que si A es infinito entonces $A \vDash \theta'$.
   4. ¿Existe $\theta'' \in \text{FORM}(\mathcal{L})$ tal que A es infinito sii $A \vDash \theta''$? Justificar.

10. (Mat.)

    a) Definir el concepto de interpretación de un lenguaje de primer orden.

    b) Sea $\mathcal{L}$ un lenguaje de igualdad y un símbolo de función binario f^2. Encontrar un enunciado en este lenguaje que exprese que una operación binaria es conmutativa y asociativa.

Solution:

a) La interpretación de un lenguaje de primer orden es una extensión del lenguaje que mapea cada símbolo constante, función k-aria y predicado k-ario a algún elemento del universo de interpretación.

Sea un lenguaje $L =< C, F, P >$ para una interpretación se define:

- Un universo de interpretación, conjunto no nulo U_I. Ejemplo: Naturales.
- Para cada símbolo constante $c \in C$, mapea con un elemento $c_I \in C_I$.
- Para cada símbolo de función k-aria$\in F$, mapea con una función f_I de k variables sobre el universo U_I, $f_I : U_I^k \to U_I$.
- Para cada símbolo de predicado k-ario$\in P$, mapea a una relación k-aria P_I sobre el universo U_I.

b) Conmutativo: $a = \forall x \forall y (f^2(x, y) = f^2(y, x))$

Asociativo: $b = \forall x \forall y \forall z (f^2(x, f^2(y, z)) = f^2(f^2(x, y), z))$.

La solución es $a \wedge b$.