Lógica y Computabilidad - Guías de Ejercicio 2020

Práctica 1

Funciones primitivas recursivas y clases PRC

Ejercicio 1.1.
*Mostrar que, dado un k fijo, la función constante $f(x) = k$ puede definirse usando las funciones iniciales y composición (*sin *usar recursión primitiva).*

Solución 1.1.
Sea $k \in \mathbb{N}$, entonces como $s(n(x)) = 1$ para todo $x \in \mathbb{N}$ se puede observar de manera inductiva que

$$\underbrace{s \dots s}_{k \text{ veces}} \circ n(x) = s^{(k)}(n(x)) = k$$

para todo $x \in \mathbb{N}$ donde s es la función inicial sucesor y n es la función inicial nula.

Por lo tanto, la función constante $f(x) = k$ puede definirse usando composición de funciones iniciales.

Ejercicio 1.2.
Probar que las siguientes funciones son primitivas recursivas, mostrando que pueden obtenerse a partir de las funciones iniciales usando composición y/o recursión primitiva:

$$f_1(x,y) = x + y \qquad f_2(x,y) = x \cdot y \qquad f_3(x,y) = x^y \qquad f_4(x,y) = \underbrace{x^{x^{\cdot^{\cdot^{\cdot^{x}}}}}}_{y \text{ veces}}$$

$$g_1(x) = x \dot{-} 1 \qquad g_2(x,y) = x \dot{-} y \qquad g_3(x,y) = \max\{x,y\} \qquad g_4(x,y) = \min\{x,y\}$$

Observaciones: *Se asume $f_4(x,0) = 1$ y $x \dot{-} y = \begin{cases} x - y & \text{si } y \leq x \\ 0 & \text{si } y > x \end{cases}$.*

Solución 1.2.

(i) La función $f_1(x,y) = x + y$, es p.r. puesto que puede escribirse como

$$f_1(x,0) = u_1^1(x,y) = x$$
$$f_1(x,y+1) = g(f_1(x,y), x, y)$$

donde $g(x,y,z) = s(u_1^3(x,y,z))$.

(ii) La función $f_2(x,y) = x \cdot y$, es p.r. puesto que puede escribirse como

$$f_2(x,0) = 0$$
$$f_2(x,y+1) = g(f_2(x,y), x, y)$$

donde $g(x,y,z) = f_1(u_1^3(x,y,z), u_2^3(x,y,z))$.

(iii) La función $f_3(x,y) = x^y$, es p.r. puesto que puede escribirse como

$$f_3(x,0) = f(x,y)$$
$$f_3(x,y+1) = g(f_3(x,y), x, y)$$

donde $f(x,y) = s(n(u_1^1(x,y))) \equiv 1$ y $g = f_2(u_1^3(x,y,z), u_2^3(x,y,z))$.

(iv) La función $f_4(x,y) = \underbrace{x^{x^{\cdot^{\cdot^{\cdot^{x}}}}}}_{y \text{ veces}}$, *es p.r. puesto que puede escribirse como*

$$\begin{aligned} f_4(x,0) &= f(x,y) \\ f_4(x,y+1) &= g(f_4(x,y),x,y) \end{aligned}$$

donde $f(x,y) = s(n(u_1^1(x,y))) \equiv 1$ *y* $g = f_3(u_2^3(x,y,z), u_1^3(x,y,z))$.

(v) La función $g_1(x) = x \dot{-} 1$, *es p.r. puesto que puede escribirse como*

$$\begin{aligned} g_1(x,0) &= 0 \\ g_1(x,t+1) &= g(g_1(x,t),x,t) \end{aligned}$$

donde $g(x,y,z) = u_1^3(x,y,z)$.

(vi) La función $g_2(x,y) = x \dot{-} y$, *es p.r. puesto que puede escribirse como*

$$\begin{aligned} g_2(x,0) &= u_1^2(x,y) \\ g_2(x,y+1) &= g(g_2(x,y),x,y) \end{aligned}$$

donde $g(x,y,z) = g_1(x,y)$

(vii) La función $g_3(x,y) = \text{máx}\{x,y\}$, *es p.r. puesto que puede escribirse como*

$$\begin{aligned} g_3(x,0) &= u_1^2(x,y) \\ g_3(x,y+1) &= g(g_3(x,y),x,y) \end{aligned}$$

donde $g(x,y,z) = f_1(1, g_3(y, g_1(x))) = (x \dot{-} y) + y$.

(viii) La función $g_4(x,y) = \text{mín}\{x,y\}$, *es p.r. puesto que puede escribirse como* $\text{mín}\{x,y\} = g_2(f_1(x,y), \text{máx}\{x,y\} = x - (x \dot{-} y)$.

Ejercicio 1.3.
Sea $\mathcal{C}$ *la clase más chica que contiene las funciones iniciales y está cerrada por composición. Sea* $f : \mathbb{N}^n \to \mathbb{N} \in \mathcal{C}$. *Demostrar que existen* $i \in \mathbb{N}$ *y* $g; \mathbb{N} \to \mathbb{N}$ *tal que* $f = g \circ u_i^n$.

Solución 1.3.

(C.B.) Para el caso base el enunciado es claramente cierto puesto que

- $s = s \circ u_1^1$.
- $n = n \circ u_1^1$.
- *y para cualesquiera* $i, n \in \mathbb{N}$, *con* $1 \leq i \leq n$ *se tiene que* $u_i^n = u_i^1 \circ u_n^n$.

(H.I.) Supongamos que vale para $f, g_1, \ldots, g_k$ *y veamos que vale para* h *obtenida por composición.*

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$$

Sean i_f *y* g_f *tal que* $g_f \circ u_{i_f}^k = f$. *Sean* i_g *y* g_g *tal que* $g_g \circ u_{i_g}^n = g_{i_f}$. *Tomando* $g = g_f \circ g_g$ *e* $i = i_g$ *se tiene que*

$$\begin{aligned} f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)) &= g_f\left(u_{i_f}^k\left(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)\right)\right) \\ &= g_f\left(g_{i_f}(x_1, \ldots, x_n)\right) \\ &= g_f\left(g_g\left(u_{i_g}^n(x_1, \ldots, x_n\right)\right) \end{aligned}$$

Ejercicio 1.4.
Sea $\mathcal{C}_i$ *la clase de funciones iniciales, es decir, aquella que contiene a:*

$$n(x) = 0 \qquad s(x) = x + 1 \qquad u_i^n(x_1, \ldots, x_n) = x_i \text{ para cada } n \in \mathbb{N} \text{ e } i \in \{1, \ldots, n\}$$

y sea $\mathcal{C}_c$ *la (mínima) clase que extiende a* $\mathcal{C}_i$ *y se encuentra cerrada por composición, i.e., si* $f, g_1, \ldots g_m$ *están en* $\mathcal{C}_c$, *entonces* $h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$ *también lo está.*

a. Demostrar que para toda $f : \mathbb{N}^n \to \mathbb{N}$, f *está en* $\mathcal{C}_c$ *sii existe* $k \geq 0$ *tal que, o bien sucede* $f(x_1, \ldots, x_n) = k$, *o bien para algún* i *fijo, se tiene* $f(x_1, \ldots, x_n) = x_i + k$.

b. Mostrar que existe una función primitiva recursiva que no está en $\mathcal{C}_c$.

Solución 1.4.

a. $(\Rightarrow)$: *Veamos que vale para las funciones iniciales,*

- $n(x) = 0$, *entonces existe* $k \geq 0$ *tal que* $n(x) = k \equiv 0$.
- $s(x) = x + 1$, *entonces existe* $k \geq 0$ $(k = 1)$ *e* i *fijo* $(i = 1)$ *tal que* $s(x_1) = x_1 + k$.
- $u_1^n(x_1, \ldots, x_n) = x_i$, *entonces existe* $k \geq 0$ $(k = 0)$ *e* i *fijo tal que* $u_1^n(x_1, \ldots, x_n) = x_i + k = x_i$.

Supongamos, ahora, que $f, g_1, \ldots g_m$ *cumplen la condición, entonces veamos que* $h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n))$ *también la cumple.*

(i) Si $f(x_1, \ldots, x_n) = k$ *para algún* $k \geq 0$, *entonces*

$$h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) = k.$$

(ii) Si $f(x_1, \ldots, x_n) = x_{i_0} + k$ *para algún* $k \geq 0$ *e* i_0 *fijo, entonces*

- *Si* $g_{i_0}(x_1, \ldots, x_n) = k'$ *para algún* $k' \geq 0$, *entonces*

$$\begin{aligned} h(x_1, \ldots, x_n) &= f(g_1(x_1, \ldots, x_n), \ldots, g_{i_0}(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \\ &= g_{i_0}(x_1, \ldots, x_n) + k \\ &= k' + k \geq 0. \end{aligned}$$

Es decir, existe $k'' \geq 0$ *tal que* $h(x_1, \ldots, x_n) = k''$.

- *Si* $g_{i_0}(x_1, \ldots, x_n) = x_{i_1} + k'$ *para algún* $k \geq 0$, *entonces*

$$\begin{aligned} h(x_1, \ldots, x_n) &= f(g_1(x_1, \ldots, x_n), \ldots, g_{i_0}(x_1, \ldots, x_n), \ldots, g_m(x_1, \ldots, x_n)) \\ &= g_{i_0}(x_1, \ldots, x_n) + k \\ &= x_{i_1} + k' + k. \end{aligned}$$

Es decir, existe $k'' \geq 0$ *e* i *fijo* (i_1) *tal que* $h(x_1, \ldots, x_n) = x_{i_1} + k''$.

$(\Leftarrow)$: *Si existe* $k \geq 0$ *tal que* $f(x_1, \ldots, x_n) = k$, *entonces*

$$f(x_1, \ldots, x_n) = \underbrace{s \ldots s}_{k \text{ veces}} \circ n(x_1, \ldots, x_n) \in \mathcal{C}_c$$

pues las funciones sucesor y nula están en $\mathcal{C}_c$ *y* $\mathcal{C}_c$ *es cerrado por composición.*
Si existe $k \geq 0$ *e* i *fijo tal que* $f(x_1, \ldots, x_n) = x_i + k$, *entonces*

$$f(x_1, \ldots, x_n) = \underbrace{s \ldots s}_{k \text{ veces}} \circ u_i^n(x_1, \ldots, x_n) \in \mathcal{C}_c$$

pues las funciones sucesor y proyección están en $\mathcal{C}_c$ *y* $\mathcal{C}_c$ *es cerrado por composición*

b.

$$f(x, y) = \begin{cases} x - y & x \geq y \\ 0 & \text{si no} \end{cases}$$

Es una función pr pero no está en $\mathcal{C}_c$ *dado que no puede escribirse como* $f(x, y) = k$ *o* $f(x, y) = x_i + k$ *para algún* $k \geq 0$ *e* i *fijo.*

Ejercicio 1.5. *Llamamos* predicado *a cualquier función* $p : \mathbb{N}^n \to \{0, 1\}$, *escribimos* $p(a_1, \ldots, a_n)$ *en lugar de* $p(a_1, \ldots, a_n) = 1$ *y decimos, informalmente, en ese caso, que "*$p(a_1, \ldots, a_n)$ *es verdadero".*
Mostrar que los predicados $\leq, \geq, =, \neq, <$ *y* $>: \mathbb{N}^2 \to \{0, 1\}$ *están en cualquier clase PRC.*

Solución 1.5.

(i) Sea $igual(x, y) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$.

Entonces $igual(x, y) = 1$ *si y sólo si* $x \dot{-} y - y \dot{-} x = 0$.

Es decir, $igual(x, y) = g_2(x, y) - g_2(y, x)$ *donde* g_2 *es pr.*

Por lo tanto, el predicado "$=$*" se escribe como composición de funciones pr.*

(ii) *Sea* $noigual(x,y) = \begin{cases} 1 & x \neq y \\ 0 & x = y \end{cases}$.

Entonces $noigual(x,y) = \neg igual(x,y)$.

Por lo tanto, el predicado "$\neq$*" se escribe como composición de funciones pr.*

(iii) *Sea* $menor\_igual(x,y) = \begin{cases} 1 & x \leq y \\ 0 & x > y \end{cases}$.

Entonces $menor\_igual(x,y) = 1$ *si y sólo si* $x \leq y$ *si y sólo si* $y \dot{-} x \neq 0$ *si y sólo si* $igual(y \dot{-} x, 0) = 0$.

Es decir, $menor\_igual(x,y) = 1 \dot{-} igual(y \dot{-} x, 0)$.

Por lo tanto, el predicado "$\leq$*" se escribe como composición de funciones pr.*

(iii) *Sea* $menor(x,y) = \begin{cases} 1 & x < y \\ 0 & x \geq y \end{cases}$.

Entonces $menor(x,y) = 1$ *si y sólo si* $x < y$ *si y sólo si* $y \dot{-} x \neq 0$ *si y sólo si* $igual(y \dot{-} x, 0) = 0$.

Es decir, $menor(x,y) = 1 \dot{-} igual(y \dot{-} x, 0)$.

Por lo tanto, el predicado "$<$*" se escribe como composición de funciones pr.*

(iii) *Sea* $mayor\_igual(x,y) = \begin{cases} 1 & x \geq y \\ 0 & x < y \end{cases}$.

Entonces $mayor\_igual(x,y) = 1$ *si y sólo si* $x \geq y$ *si y sólo si* $menor\_igual(y,x) = 1$.

Es decir, $mayor\_igual(x,y) = menor\_igual(y,x)$.

Por lo tanto, el predicado "$\geq$*" se escribe como composición de funciones pr.*

(iii) *Sea* $mayor(x,y) = \begin{cases} 1 & x > y \\ 0 & x \leq y \end{cases}$.

Entonces $mayor\_igual(x,y) = 1$ *si y sólo si* $x > y$ *si y sólo si* $menor(y,x) = 1$.

Es decir, $mayor(x,y) = menor(y,x)$.

Por lo tanto, el predicado "$<$*" se escribe como composición de funciones pr.*

Ejercicio 1.6.
Sea $\mathcal{C}$ *una clase PRC, sean* $f_1, \ldots, f_k, g : \mathbb{N}^n \to \mathbb{N}$ *funciones en* $\mathcal{C}$ *y sean también* $p_1, \ldots, p_k : \mathbb{N}^n \to \{0,1\}$ *predicados disjuntos en* $\mathcal{C}$ *(es decir, no sucede* $p_i(a_1, \ldots, a_n) = p_j(a_1, \ldots, a_n) = 1$ *con* $i \neq j$ *para ningún* $(a_1, \ldots, a_n) \in \mathbb{N}^n$*). Mostrar que también está en* $\mathcal{C}$ *cualquier función* h *que cumpla:*

$$h(x_1, \ldots, x_n) = \begin{cases} f_1(x_1, \ldots, x_n) & si\ p_1(x_1, \ldots, x_n) \\ \vdots & \\ f_k(x_1, \ldots, x_n) & si\ p_k(x_1, \ldots, x_n) \\ g(x_1, \ldots, x_n) & si\ no \end{cases}$$

Observar que h *queda completamente determinada por este esquema.*

Solución 1.6.
Basta con observar que como los predicados son disjunto sen $\mathcal{C}$*, la función* h *puede ser escrita como*

$$h(x_1, \ldots, x_n) = \sum_{j=1}^{k} f_i(x_1, \ldots, x_n) p_i(x_1, \ldots, x_n) + g(x_1, \ldots, x_n) \left(1 - \sum_{j=1}^{k} p_i(x_1, \ldots, x_n)\right).$$

Ejercicio 1.7.

a. Demostrar que el predicado $par(x) = \begin{cases} 1 & si\ x\ es\ par \\ 0 & si\ no \end{cases}$ *está en toda clase PRC.*

b. Demostrar que la función $f(x) = \lfloor x/2 \rfloor$ *está en toda clase PRC.*

c. Sea $\mathcal{C}$ una clase PRC, y sean $f : \mathbb{N}^n \to \mathbb{N}$ y $g_1, g_2 : \mathbb{N}^{n+2} \to \mathbb{N}$ funciones en $\mathcal{C}$. Mostrar que también está en $\mathcal{C}$ cualquier h que cumpla:

$$h(x_1, \ldots, x_n, t) = \begin{cases} f(x_1, \ldots, x_n) & si\ t = 0 \\ g_1(x_1, \ldots, x_n, k, h(x_1, \ldots, x_n, t-1)) & si\ t = 2 \cdot k + 1 \\ g_2(x_1, \ldots, x_n, k, h(x_1, \ldots, x_n, t-1)) & si\ t = 2 \cdot k + 2 \end{cases}$$

Observar que h queda completamente determinada por este esquema.

Solución 1.7.

a. Sea h una función pr definida por recurrencia,

$$h(0) = 1$$
$$h(t+1) = g(h(t), t)$$
$$donde\ g(x,t) = 1 - u_1^2(x,t).$$

Usando inducción en t, se probará que $h(t) = par(t)$ y, de esa manera, el predicado par es pr

- *$t = 0$:*
  si $t = 0$, entonces $h(0) = 1 = par(0)$.
- *Paso Inductivo:*
  Sabiendo que $h(t) = par(t)$, se quiere ver que $h(t+1) = par(t+1)$, luego

$$h(t+1) = g(h(t), t) = 1 - h(t) = \begin{cases} 1 & t\ no\ es\ par \\ 0 & t\ es\ par \end{cases}$$
$$= \begin{cases} par(t+1) & t\ no\ es\ par \\ par(t+1) & t\ es\ par \end{cases}$$

Por lo tanto, $h(t) = par(t)$ para todo $t \in \mathbb{N}$.

b. Sea h una función pr definida por recurrencia,

$$h(0) = 0$$
$$h(t+1) = g(h(t), t)$$
$$donde\ g(x,t) = x + par(t+1).$$

Usando inducción en t, se probará que $h(t) = f(t) = \lfloor t/2 \rfloor$ y, de esa manera, la función $f(x)$ es pr

- *$t = 0$:*
  si $t = 0$, entonces $h(0) = 0 = \lfloor 0/2 \rfloor$.
- *Paso Inductivo:*
  Sabiendo que $h(t) = \lfloor t/2 \rfloor$, se quiere ver que $h(t+1) = \lfloor (t+1)/2 \rfloor$, luego

$$h(t+1) = g(h(t), t) = h(t) + par(t+1) = \begin{cases} \lfloor t/2 \rfloor & t\ es\ par \\ \lfloor t/2 \rfloor + 1 & t\ no\ es\ par \end{cases}$$
$$= \begin{cases} \lfloor (t+1)/2 \rfloor & t\ es\ par \\ \lfloor t/2 \rfloor + \frac{2}{2} & t\ no\ es\ par \end{cases} = \begin{cases} \lfloor (t+1)/2 \rfloor & t\ es\ par \\ \lfloor (t+1)/2 \rfloor & t\ no\ es\ par \end{cases}$$

Por lo tanto, $h(t) = \lfloor (t+1)/2 \rfloor$ para todo $t \in \mathbb{N}$.

c. h está en $\mathcal{C}$ puesto que está dividida en casos finitos y disjuntos donde las guardas son predicados pr (comparar por igualdad es pr) y los resultados son funciones pr.

Ejercicio 1.8.
Sea $\mathcal{C}$ una clase PRC y sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado en $\mathcal{C}$. Mostrar que también están en $\mathcal{C}$ las siguientes

funciones:

$$\text{cantidad}_p(x_1,\dots,x_n,y,z) = |\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\}|$$

$$\text{todos}_p(x_1,\dots,x_n,y,z) = \begin{cases} 1 & si\ (\forall t : y \le t \le z)p(x_1,\dots,x_n,t) \\ 0 & si\ no \end{cases}$$

$$\text{alguno}_p(x_1,\dots,x_n,y,z) = \begin{cases} 1 & si\ (\exists t : y \le t \le z)p(x_1,\dots,x_n,t) \\ 0 & si\ no \end{cases}$$

$$\text{minimo}_p(x_1,\dots,x_n,y,z) = \begin{cases} \text{mín}\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\} & si\ z \ge y \\ 0 & si\ no \end{cases}$$

$$\text{maximo}_p(x_1,\dots,x_n,y,z) = \begin{cases} \text{máx}\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\} & si\ z \ge y \\ 0 & si\ no \end{cases}$$

$$\text{unico}_p(x_1,\dots,x_n,y,z) = \begin{cases} u & si\ \{u\} = \{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\} \\ z+1 & si\ no \end{cases}$$

Observación: pueden usarse los operadores acotados $(\text{mín}, \Sigma, \forall, \exists)$ *vistos en la teórica.*

Solución 1.8.

(i) $\text{cantidad}_p(x_1,\dots,x_n,y,z) = \sum_{t=0}^{z} p(x_1,\dots,x_n,t)\mathbb{1}_{\{y\le t\}}$.

Otra manera es observar que

$$\text{cantidad}_p(x_1,\dots,x_n,y,z) = \sum_{t=0}^{z} p(x_1,\dots,x_n,t) - \sum_{t=0}^{y-1} p(x_1,\dots,x_n,t)$$

(ii) En este caso, usando el ejercicio 5, basta con ver que $(\forall t : y \le t \le z)p(x_1,\dots,x_n,t)$ *está en* $\mathcal{C}$.

$$(\forall t : y \le y \le z)p(x_1,\dots,x_n,t) = (\forall t)_{\le z}((t \ge y) \to p(x_1,\dots,x_n,t))$$

(iii) En este caso, usando el ejercicio 5, basta con ver que $(\exists t : y \le t \le z)p(x_1,\dots,x_n,t)$ *está en* $\mathcal{C}$*, pero ese predicado está en* $\mathcal{C}$ *si y sólo si su negación también está. Por lo tanto,*

$$\begin{aligned}\neg[(\exists t : y \le t \le z)p(x_1,\dots,x_n,t)] &= \neg[(\exists t : y \le t \le z)\neg p(x_1,\dots,x_n,t) \\ &= (\forall t : y \le t \le z)\neg p(x_1,\dots,x_n,t)\end{aligned}$$

(iv) En este caso, sabemos que la comparación menor o igual $y \le z$ *está en* $\mathcal{C}$*, por lo tanto, por el ejercicio 1.6, basta con ver que* $\text{mín}\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\}$ *está en* $\mathcal{C}$.

$$\text{mín}\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\} = \min_{t\le z}\left(\mathbb{1}_{\{y\le t\}} \cdot p(x_1,\dots,x_n,t)\right)$$

(v) En este caso, sabemos que la comparación menor o igual $y \le z$ *está en* $\mathcal{C}$*, por lo tanto, por el ejercicio 1.6, basta con ver que* $\text{máx}\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\}$ *está en* $\mathcal{C}$.

$$\text{máx}\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\} = y - \sum_{u=0}^{y}\prod_{t=0}^{u}\alpha(p(x_1,\dots,x_n,y-t))$$

(vi) En este caso, sabemos que la proyección (u) y la suma $(z+1)$ *están en* $\mathcal{C}$*, por lo tanto, por el ejercicio 1.6, basta con ver que* $\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\}$ *está en* $\mathcal{C}$.

$$\{t \mid y \le t \le z \wedge p(x_1,\dots,x_n,t)\}) = \mathbb{1}_{\{t\ge y\}}\mathbb{1}_{\{t\le z\}}p(x_1,\dots,x_n,t)$$

Ejercicio 1.9.

Mostrar que las siguientes funciones están en toda clase PRC:

$$\text{cociente}(x,y) = \lfloor x/y \rfloor$$

$$\text{resto}(x,y) = x \bmod y$$

$$\text{divide}(x,y) = \begin{cases} 1 & si\ x\ divide\ a\ y \\ 0 & si\ no \end{cases}$$

$$\text{primo}(x) = \begin{cases} 1 & si\ x\ es\ un\ número\ primo \\ 0 & si\ no \end{cases}$$

$$\text{raiz}(x,y) = \begin{cases} \lfloor \sqrt[x]{y} \rfloor & si\ x \ne 0 \\ 0 & si\ x = 0 \end{cases}$$

$$\text{nprimo}(n) = k \text{ sii } k \text{ es primo y hay sólo } n \text{ primos positivos menores que } k$$

Observación: Se asume que $cociente(x,0) = 0$ *y* $resto(x,0) = x$.

Solución 1.9.

(i) $\text{cociente}(x,y) = \min_{t\leq x}\{(t+1)\cdot y > x\}$.

(ii) $\text{resto}(x,y) = x \dot{-} (y \cdot \lfloor x/y \rfloor)$.

(iii) $\text{divide}(x,y) \equiv (\exists z)_{\leq y}(y = z \cdot x)$.

(iv) $\text{primo}(x) \equiv (x>1) \wedge (\forall t)_{\leq x}((t|x) \to (t=1 \vee t=x))$.

(v) $\text{raiz}(x,y) = \sum_{i=1}^{y}(i^x > y) = \min_{t\leq y}\{(t+1)^x > y\}$.

(vi) $\text{nprimo}(n) = \min_{t\leq K(n)}(primo(t) \wedge t > p_n)$, *donde* $K(n) = p_n! + 1$.

Ejercicio 1.10.
Mostrar que, dado un predicado $p(x_1,\ldots,x_n,t)$ pr, la siguiente función es p.r.

$$max\_acot(x_1,\ldots,x_n,b) = \begin{cases} b+1 & si\ (\forall y)_{y\leq b}\neg p(x_1,\ldots,x_n,y) \\ \text{máx}_{t\leq b}\, p(x_1,\ldots,x_n,t) & si\ no \end{cases}$$

Solución 1.10.
Sabemos si un predicado $q(x_1,\ldots,x_n,t)$ es pr entonces la minimización acotada

$$min\_acot(x_1,\ldots,x_n,b) = \begin{cases} b+1 & si\ (\forall y)_{y\leq b}\neg q(x_1,\ldots,x_n,y) \\ \text{mín}_{t\leq b}\, q(x_1,\ldots,x_n,t) & si\ no \end{cases}$$

es pr. Luego si $q(x_1,\ldots,x_n,t) = p(x_1,\ldots,x_n,t) \wedge (\forall t')_{t<t'\leq b}\neg p(x_1,\ldots,x_n,t')$

$$max\_acot = \min_{t\leq b} q(x_1,\ldots,x_n,t)$$

En este caso, $q(x_1,\ldots,x_n,t)$ dice exactamente que t cumple $p(x_1,\ldots,x_n,t)$ y, además, no hay otro más grande que lo cumpla (hasta la cota). Es decir que es el máximo que cumple $p(x_1,\ldots,x_n,t)$. Y como la cuantificación acotada, los operadores lógicos, las comparaciones entre números y nuestra cota son pr tenemos $q(x_1,\ldots,x_n,t)$ pr y, por lo tanto, nuestra función maxacot es pr.

Ejercicio 1.11.
Considerar la codificación de pares de naturales dada por $\langle x,y\rangle = 2^x(2y+1) \dot{-} 1$.

Mostrar que las funciones observadoras $\ell, r : \mathbb{N} \to \mathbb{N}$ tales que $\ell(\langle x,y\rangle) = x$ y $r(\langle x,y\rangle) = y$ están en toda clase PRC.

Solución 1.11.
Basta con ver que ambas son recursivas primitas. Luego,

$$\ell(\langle x,y\rangle) = \text{máx}\{t\in\mathbb{N} : 2^y | \langle x,y\rangle + 1\}$$

$$r(\langle x,y\rangle) = \frac{1}{2}\left(\frac{\langle x,y\rangle + 1}{2^{\ell(\langle x,y\rangle)}} - 1\right)$$

Ejercicio 1.12.

1. Mostrar que fib, la función de Fibonacci, está en toda clase PRC, donde:

$$\begin{aligned} fib(0) &= 0 \\ fib(1) &= 1 \\ fib(n+2) &= fib(n+1) + fib(n) \end{aligned}$$

2. Demostrar que la siguiente función $f : \mathbb{N} \to \mathbb{N}$ es primitiva recursiva:

$$\begin{aligned} f(0) &= 1 \\ f(1) &= 2 \\ f(2) &= 4 \\ f(n+3) &= f(n) + f(n+1)^2 + f(n+2)^3 \end{aligned}$$

Solución 1.12.

1. *$F(n) = [fib(n), fib(n+1)]$ de manera que, entonces, $fib(n) = F(n)[1]$. Luego, para ver que fib es pr basta con ver que F lo es.*

$$F(0) = k = [0,1], \qquad \text{notar que lo importante es que } F(0) \text{ debe ser una constante}$$
$$F(n+1) = g(F(n), n) \underbrace{=}_{\substack{buscamos\ que \\ valga\ ésto}} [f(n+1), f(n+2)]$$

Entonces, definiendo $F(n+1) = [F(n)[2], F(n)[2] + F(n)[1]]$. Es decir, $g(x,y) = [x[2], x[2] + y[1]]$.

- *Otra manera de que fib es pr:*

Sea $f(n) = \langle fib(n), fibn+1)\rangle$. De esta manera,

$$\begin{aligned} f(0) &= \langle fib(0), fib(1)\rangle = \langle 0,1\rangle = 2 \\ f(n+1) &= \langle fib(n+1), fib(n+2)\rangle = \langle fib(n+1), fib(n) + fib(n+1)\rangle \\ &= \langle r(f(n)), l(f(n)) + r(f(n+1))\rangle = g(f(n), n) \end{aligned}$$

Por lo tanto el caso recursivo depende únicamente del valor de la función en el punto anterior, haciendo que f use recursión primitiva.

Observar que demostramos que f hace uso de recursión primitiva a partir de analizar casos particulares usando su de

nición. Si hubiésemos dado la de

nición por casos de manera directa (es decir, sin derivarla de su de

nición) habría que demostrar por inducción que la misma calcula lo que decimos que calcula.

Resta ver cómo obtener fib a partir de f, pero esto es sencillo: simplemente proyectamos la primer componente del par.

$$fib(n) = \ell(f(n))$$

Como f y la proyección ℓ de pares están en toda clase PRC, entonces fib también lo está.

2. *En el esquema de recursión primitiva visto en clase, para calcular cada paso de la recursión, sólo puede usarse el paso anterior. Sin embargo, f utiliza tres pasos anteriores. Por lo tanto, para resolver este problema, sea $F : \mathbb{N} \to \mathbb{N}$ auxiliar tal que:*

- *guarde los valores de f que se necesitan en cada paso.*
- *sea fácil justificar que es p.r. (pues sólo usaremos su paso anterior).*
- *podamos extraer el valor de f que se necesita de manera p.r.*

es decir, sea $F : \mathbb{N} \to \mathbb{N}$ definida por:

$$F(n) = [f(n), f(n+1), f(n+2)]$$

tal que se ajusta al e squema de recursión primitiva:

$$\begin{aligned} F(0) &= [f(0), f(1), f(2)] = [1,2,4], \ \textit{(es una constante y por lo tanto es p.r.)} \\ F(n+1) &= [f(n), f(n+1), f(n+2)], \ \textit{(definición de } F\textit{)} \\ &= [f(n+1), f(n+2), f(n) + f(n+1)^2 + f(n+2)^3], \ \textit{(definición de } f\textit{)} \\ &= [F(n)[2], F(n)[3], F(n)[1] + (F(n)[2])^2 + (F(n)[3])^3], \ \textit{(definición de } F\textit{)} \end{aligned}$$

A partir de esta última línea, es sencillo justificar que F es p. r., dado que $F(0)$ resultó una función p.r. (es constante) y que $F(n+1)$ resultó una función p.r. (es composición de todas funciones p.r: tomar observadores de listas, sumar, elevar a potencias, construir listas) que sólo depende del valor anterior $F(n)$, la función F se ajusta al esquema de recursión primitiva visto en clase y, por lo tanto, es p.r.

Por último, observemos que $f(n) = F(n)[1]$ y, por lo tanto, es primitiva recursiva ya que es composición de funciones que lo son (F lo es y los observadores de listas lo son).

Ejercicio 1.13.
Demostrar que toda clase PRC se encuentra cerrada por recursión mutua. *Es decir, dada $\mathcal{C}$, una clase PRC y dadas f_1, f_2, g_1 y g_2 funciones en $\mathcal{C}$, mostrar que también están en $\mathcal{C}$ las funciones h_1 y h_2 que cumplen:*

$$h_1(x_1, \ldots, x_n, t) = \begin{cases} f_1(x_1, \ldots, x_n) & \textit{si } t = 0 \\ g_1(h_1(x_1, \ldots, x_n, t-1), h_2(x_1, \ldots, x_n, t-1), x_1, \ldots, x_n, t) & \textit{si no} \end{cases}$$
$$h_2(x_1, \ldots, x_n, t) = \begin{cases} f_2(x_1, \ldots, x_n) & \textit{si } t = 0 \\ g_2(h_2(x_1, \ldots, x_n, t-1), h_1(x_1, \ldots, x_n, t-1), x_1, \ldots, x_n, t) & \textit{si no} \end{cases}$$

Observar que h_1 y h_2 quedan completamente determinadas por el esquema de recursión mutua.

Solución 1.13.

Ejercicio 1.14.
Sea $\mathcal{C}_{i+p}$ la clase de funciones que extiende a la clase de funciones iniciales $\mathcal{C}_i$ con la función codificadora de pares $\langle\cdot,\cdot\rangle : \mathbb{N}^2 \to \mathbb{N}$ y las proyectoras $l, r : \mathbb{N} \to \mathbb{N}$ y sea $\mathcal{C}_{Ack}$ la (mínima) clase que incluye a $\mathcal{C}_{i+p}$ y se encuentra cerrada por composición y por iteración de funciones unarias*, i.e., si $f : \mathbb{N} \to \mathbb{N}$ está en $\mathcal{C}_{Ack}$, entonces también está $h(n,x) = f^{(n)}(x)$ (recordar que $f^{(n)} = \underbrace{f \circ f \circ \cdots \circ f}_{n}$).*

a. *Demostrar que $\mathcal{C}_{Ack} \subset PRC$.*

b. *Observar que en $\mathcal{C}_{Ack}$ se tienen las funciones codificadoras de n-tuplas y sus observadoras.*

c. *Demostrar que si $f\mathbb{N}^n \to \mathbb{N}$ y $g : \mathbb{N}^{n+2} \to \mathbb{N}$ pertenecen a la clase $\mathcal{C}_{Ack}$ y h se obtiene mediante el esquema de recursión primitiva a partir de f y g, entonces la función $s : \mathbb{N}^{n+1} \to \mathbb{N}$ definida por $s(\overline{x}, y) = \langle \overline{x}, y, h(\overline{x}, y)\rangle$ también pertenece a la clase $\mathcal{C}_{Ack}$.*

d. *Concluir que $PRC \subset \mathcal{C}_{Ack}$ y, por lo tanto, coinciden.*

Solución 1.14.

a.
- *$\mathcal{C}_i \subset PR$ pues todas las funciones iniciales son pr.*
- *$\mathcal{C}_{i+p} \subset PR$ pues las funciones iniciales, las codificaciones de pares (ejercicio anterior) y los obsevadores (ejercicio anterior) son pr.*
- *$\mathcal{C}_{i+p} \cup \{composición\} \subset PR$ pues las funciones de $\mathcal{C}_{i+p}$ son pr y composición de pr es pr.*
- *Sea f pr tal que $h(n,x) = \underbrace{f \circ f \circ \cdots \circ f}_{n \text{ veces}}(x)$ pero como h resulta ser composición de pr, resulta ser pr.*

Luego, $\mathcal{C}_{Ack} \subset PRC$

b.

c. *Para ver que $s(\overline{x}, y) \in \mathcal{C}_{Ack}$ basta con ver que $h(x_1, \ldots, x_n, y) \in \mathcal{C}_{Ack}$ puesto que, de esta manera, s resulta ser una codificación de $n+2$ tuplas que están en $\mathcal{C}_{Ack}$.*

Por inducción en t,

- *Si $t = 0$, entonces $h(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n)$ con $f \in \mathcal{C}_{Ack}$.*
- *Suponiendo que $h(x_1, \ldots, x_n, t) \in \mathcal{C}_{Ack}$, entonces*

$$h(x_1, \ldots, x_n, t+1) = g(h(x_1, \ldots, x_n, t), x_1, \ldots, x_n, t) \text{ con } g, h \in \mathcal{C}_{Ack}$$

Luego, $h \in \mathcal{C}_{Ack}$.

d. *Como las funcione siniciales están en $\mathcal{C}_{Ack}$ y $\mathcal{C}_{Ack}$ está cerrada por composición y recursión primitiva, entonces toda función pr está en $\mathcal{C}_{Ack}$.*

Luego, $PRC \subset \mathcal{C}_{Ack}$

Ejercicio 1.15.
Considerar la codificación de secuencias finitas de números naturales dada por

$$[a_0, \ldots, a_{n-1}] = \prod_{i=0}^{n-1} \text{nprimo}(i)^{a_i}, \text{ donde nprimo}$$

es la función definida en el Ejercicio 1.9.

a. *Mostrar que la codificación dada forma una biyección entre el conjunto de secuencias finitas que no terminan en cero y los números naturales mayores que cero.*

b. *Determinar qué valor codifica la secuencia vacía y mostrar que las siguientes funciones están en toda clase PRC:*

- *$|\cdot| : \mathbb{N} \to \mathbb{N}$ tal que $|[a_0, \ldots, a_{n-1}]| = n$* *(longitud)*
- *$\pi_\cdot(\cdot) : \mathbb{N}^2 \to \mathbb{N}$ tal que $\pi_i([a_0, \ldots, a_{n-1}]) = \begin{cases} a_i & si\ i < n \\ 0 & si\ no \end{cases}$* *(proyección)*
- *$[\cdot] : \mathbb{N} \to \mathbb{N}$ tal que $[x]$ es la lista con único elemento x* *(creación)*
- *$\cdot \circ \cdot : \mathbb{N}^2 \to \mathbb{N}$ tal que $[a_1, \ldots, a_n] \circ [b_1, \ldots, b_m] = [a_1, \ldots a_n, b_1, \ldots b_m]$* *(concatenación)*

- $\text{sub} : \mathbb{N}^3 \to \mathbb{N}$ *tal que* $\text{sub}([a_1, \ldots, a_n], i, j) = [a_i, \ldots, a_j]$ *(sublista)*

c. *Proponer una codificación de secuencias* $\rho : \text{Listas} \to \mathbb{N}$ *que forme una biyección entre los números naturales (incluyendo el cero) y el conjunto de* todas *las secuencias finitas de naturales tal que las funciones del punto b estén en toda clase PRC.*

Solución 1.15.

a. *Sea* $A = \{$*conjunto de secuencias finitas que no terminan en 0*$\}$

$$A \xrightarrow{f} \mathbb{N}_0$$
$$[a_1, \ldots, a_n] \longmapsto \prod_{i=0}^{n-1} \text{nprimo}(i)^{a_i}$$

- *Inyectiva:*
  Sean $\ell_1, \ell_2 \in A$ *y suponemos que* $f(\ell_1) = f(\ell_2)$*, entonces*

$$f(\ell_1) = \prod_{i=0}^{n-1} \text{nprimo}(i)^{a_i^{(1)}} = \prod_{i=0}^{n-1} \text{nprimo}(i)^{a_i^1} = \prod_{i=0}^{n-1} \text{nprimo}(i)^{a_i^2} = f(\ell_2)$$

  Como la factorizaci ón de números primos es única, entonces $a_i^1 = a_i^2$ *para todo* $1 \leq i \leq n$ *y, por lo tanto,* $\ell_1 = \ell_2$*.*

- *Sobreyectiva:*
  Sea $x \in \mathbb{N}$*, entonces* $x = \prod_{i=0}^{n-1} \text{nprimo}(i)^{a_i}$ *(su factorización en primo), entonces si* $\ell = [a_1, \ldots, a_n]$ *se tiene que* $f(\ell) = x$*.*

Ejercicio 1.16.
Mostrar que $f : [\mathbb{N}] \to [\mathbb{N}]$ *es primitiva recursiva, donde* f *toma una lista y devuelve otra lista de la misma longitud, tal que los valores de las posiciones impares son iguales a la original y las pares son iguales al cuadrado del valor que había en la misma posicióon en la lista original.*

Ejemplo: dada la lista $x = [1, 2, 3, 4, 5, 6]$*,* $f(x) = [1, 4, 3, 16, 5, 36]$*.*

Solución 1.16.
Recordar que la codificación de listas es biyectiva y que tanto sus constructores como sus observadores son p.r.

Sabiendo entonces que una lista ls se caracteriza por ser la productoria de los primeros $|ls|$ *primos, y que el exponente que lleva el primo i-ésimo es el valor de* $ls[i]$ *entonces, lo que se nos está pidiendo en el enunciado es que los exponentes de las posiciones pares sean elevados al cuadrado, mientras que el resto no sean tocados.*

Una manera es trabajar directamente con la codificación y si i es par entonces elevo al cuadrado y sino no. Para ello puedo utilizar el predicado $\alpha(i \bmod 2)$*. Notar que* $2^{\alpha(i \bmod 2)}$ *será igual a 2 cuando i sea par y será 1 en caso contrario ya que* $2^0 = 1$*.*

Por lo tanto, utilizando lo anterior, podemos definir la función de la siguiente manera:

$$f(x) = \prod_{i=1}^{|x|} primo_i^{x[i]^{2^{\alpha(i \bmod 2)}}}$$

Ejercicio 1.17.
Una lista x *de pares de naturales se llama camino de longitud* n *que comienza en* a_0 *si*

$$\ell = [\langle a_0, a_1\rangle, \langle a_1, a_2\rangle, \ldots, \langle a_{n-2}, a_{n-1}\rangle, \langle a_{n-1}, a_n\rangle].$$

Sea $f : \mathbb{N}^3 \to \mathbb{N}$ *una función tal que* $f(x, a_0, n)$ *devuelve el camino más largo de longitud menor o igual a , que comienza en* a_0 *y tal que todos los elementos del camino (es decir, los pares de naturales) son elementos de la lista* x*. Si no existe tal camino, la función debe devolver cero; si existe más de uno, debe devolver cualquiera de ellos.*

Demostrar que f *es primitiva recursiva (por simplicidad, se considera que el par* $\langle 0, 0\rangle \notin x$*).*

Solución 1.17.
Una primera idea para resolver este ejercicio es utilizar recursión para "generar" una lista que cumpla con lo pedido. Sin embargo, resolver el ejercicio de esta manera no es, a simple vista, sencillo. Notar que resulta muchísimo más fácil verificar que una solución a este problema es válida, que armar una solución válida. Si se tuviera un conjunto con todas las listas de pares de naturales, se podría revisar y "elegir" el que se necesita. Y si se pudiera hacer de manera primitiva recursiva, se estaría probando lo pedido. Sin embargo, este conjunto es infinito, y no parece razonable pensar que revisar un conjunto infinito sea primitivo recursivo.

Supongamos por un momento que verificar que una lista de pares es un camino es primitiva recursiva y se intentará acotar ese conjunto. Es decir, armar un conjunto finito de listas de pares candidatas a cumplir con lo pedido. En ese caso, nos alcanzaría con revisar uno por uno los elementos de ese conjunto, hasta encontrar uno que cumpla lo pedido. Si esta "revisión" fuera primitiva recursiva, como la veri

cación también lo es, se estaría probando lo pedido.

Dado que las secuencias se representan con números naturales, y que suponemos que el conjunto de búsqueda es acotado, existe una secuencia cuya representación en naturales tiene el valor máximo. Llamemos por el momento a esa secuencia max_seq. También supongamos que tenemos el predicado $P(x, \ell, a_0, n)$ que devuelve 1 cuando x es una posible solución a $f(\ell, a_0, n)$. Aprovechando esta idea, se puede utilizar minimización acotada para encontrar la mínima secuencia que cumple lo pedido:

$$f(\ell, a_0, n) = \min_{x \leq max\_seq} P(x, \ell, a_0, n)$$

Si bien el resultado de este algoritmo es siempre un camino de longitud menor o igual que n, comenzado en a_0, podria suceder que no fuera el más largo posible. Por eso, se debe agregar a la definición

$$f(\ell, a_0, n) = \min_{x \leq max\_seq} P(x, \ell, a_0, n) \wedge \left(\forall_{y \leq max\_seq} P(x, \ell, a_0, n) \rightarrow |y| \leq |x|\right)$$

Es decir, se busca la mínima secuencia que sea un camino, pero además, cualquier otra secuencia que cumpla con ser camino, debe ser de longitud menor o igual a la devuelta. De esta manera, se asegura que la secuencia obtenida es una de las de máxima longitud. Luego, falta ver cómo definir $P(x, \ell, a_0, n)$ y decidir qué valor toma max_seq. Para que una lista sea un camino que cumple lo pedido en el enunciado debe suceder:

- *La longitud de x es menor o igual que n.*
- *Todos los elementos de x deben ser elementos de ℓ.*
- *El primer elemento del primer par debe ser a_0.*
- *El segundo elemento de un par debe ser igual al primer elemento del siguiente par.*

Luego, podemos escribir $P(x, \ell, a_0, n)$ como:

$$P(x, \ell, a_0, n) = |x| \leq n \wedge \forall_{0 \leq i \leq |x|} x[i] \in \ell \wedge \ell(x[1]) = a_0 \wedge \forall_{0 \leq i \leq |x|} r(x[i]) = \ell(x[i+1])$$

Por último, falta ver cómo se escribe max_seq de forma p.r. Para ello se necesita encontrar una cota superior a la lista que puede obtenerse como respuesta. Si bien los elementos de la lista son pares, éstos son representados con naturales. Por lo tanto, hay alguno cuya representación es la máxima. Si se arna una lista que repita n veces a este par, seguro que la representación en naturales de la solución de f va a ser menor o igual a este valor. Luego, usando la representación de Godel:

$$max\_seq = \prod_{i=1}^{n} primo_i^{\text{máx}(\ell)}$$

Ejercicio 1.18.
Sea $f : \mathbb{N} \rightarrow \mathbb{N}$ una función primitiva recursiva y $P : \mathbb{N} \rightarrow \{0, 1\}$ un predicado primitivo recursivo. Probar que las siguientes funciones son primitivas recursivas:

1. *$map_f : \mathbb{N} \rightarrow \mathbb{N}$, que interpreta la entrada como una lista (sin ceros al final) y devuelve la lista con el resultado de aplicar f a cada uno de sus elementos. Por ejemplo, si $doble(x) = 2x$, entonces*

$$map_{doble}([1, 2, 3, 5]) = [2, 4, 6, 10].$$

2. *$filter_P : \mathbb{N} \rightarrow \mathbb{N}$, que interpreta la entrada como lista sin ceros al final y devuelve la lista con los elementos de la misma que cumplen con P (en el mismo orden que estaban en la lista original). $filter(0)$ se de*

   ne como 1. Por ejemplo, si $esPar(x)$ representa 'x es par' entonces

$$filter_{esPar}([1, 8, 3, 6, 6]) = [8, 6, 6]$$
$$filter_{esPar}([1, 3, 3, 7, 9]) = []$$

Solución 1.18.

1. *La idea será desarmar la lista, aplicar f y construir la lista otra vez.*

   Luego, basta con notar que

$$map_f(\ell) = \prod_{i=1}^{|\ell|} primo_i^{f\ell[i]} = [f(\ell[1]), f(\ell[2]), \ldots, f(\ell[|\ell|])]$$

   Luego, como tomar longitud de una lista, calcular el i-ésimo primo, f, los observadores de listas, las potencias y tomar productoria son funciones p.r., map_f resulta p.r.

2. *En este caso el problema de que no se sabe en qué posiciones de la lista resultado aparecen los elementos (que aparecen) de la lista de entrada (porque no se sabe cuáles hay que poner y cuáles no). Entonces: se buscan y se construye la lista compuesta por el primer elemento que aparece, el segundo elemento que aparece, etc*

   Para ésto, de

   nimos una función $indice_P(l, t)$ que nos dice en quée posición de la lista ℓ aparece el t-éesimo elemento que cumple P.

$$indice_P([1,8,3,6,6],1) = 2$$
$$indice_P([1,8,3,6,6],2) = 4$$
$$indice_P([1,8,3,6,6],3) = 5$$

   De esta manera, abusando de la notación,

$$filter_P(\ell) = [indice_P(\ell,1), indice_P(\ell,2), \ldots, indice_P(\ell, cuantos_P(\ell))].$$

   Aclaración: $cuantos_P(\ell))$ cuenta cuántos elementos de ℓ cumplen P.

   Formalmente, se pensará $indice_P$ de manera recursiva para escribirla usando un esquema de recursión primitiva.

   Sea $indice'_P$ que devuelve en 0 el primer elemento que cumple P.

   (C.B) $indice'_P(\ell, 0)$, es decir el primer elemento de la lista que cumple P. Luego,

$$indice'_P(\ell,0) = \min_{1\le t\le|\ell|}{}^{(} \ell([t]))$$

   (C.R.) Se conoce que el índice del t-ésimo elemento y se quiere averiguar el índice del $t+1$-ésimo. Luego, el índice en cuestión es el primero de los que cumplen P y es mayor que $indice'_P(\ell, t)$.

$$indice'_P(\ell,t+1) = \min_{1\le t\le|\ell|}{}^{(} \ell([t])) \wedge t > indice'_P(\ell,t)$$

   Escrita de esta manera y usando que la minimización acotada, la longitud de una lista, los observadores de listas, la comparación por mayor y el predicado P son p.r., es inmediato que $indice'_P$ es p.r. . Por lo tanto, $indice_P(\ell, t) = indice'_P(\ell, t \dot{-} 1)$ también lo es.

   Finalmente, escribimos formalmente

$$filter_P(\ell) = \prod_{i=1}^{cuanto_P(\ell)} primo_i^{\ell[indice(\ell,i)]}$$

   que resulta p.r.

Ejercicio 1.19.
El problema subset sum (suma de subconjunto) se define de la siguiente manera: dado un conjunto S de números enteros y un entero k, decidir si existe un subconjunto de S tal que la suma de sus elementos sea exactamente k.

Probar que es primitivo recursivo el predicado $subsetSum(l, k) : \mathbb{N} \to \{0, 1\}$ que decide si, dada ℓ una lista de naturales y un natural k, existe una sublista de ℓ cuyos elementos sumen exactamente k.

Solución 1.19.
La idea es lograr escribir algo como

$$\exists_{\ell' \subset \ell} suma(\ell') = k$$

donde ℓ' es una sublista de ℓ.

Se sabe que

- *La condición que debe cumplir la sublista, es p.r. (es una sumatoria de elementos de una lista y una comparación por igual).*

- *Tenemos un existencial p.r. para predicados p.r. pero es acotado sobre un rango.*

Luego, faltaría convertir la condición $\ell' \subset \ell$ a un rango con una cota (que además sea p.r.).

Una manera de hacer ésto es proponer una cota muy grosera (pero que sea cota) y hacer más estricta la condición del existencial. Notar que se está probando existencia, no complejidad; por lo que no importa que las cotas sean absurdamente grandes mientras sean cotas y p.r..

Luego, sea ℓ una lista de naturales y ℓ' una subtlista de ℓ (con los elementos en el mismo orden en el que aparecen en ℓ), entonces $\#(\ell') \leq \#(\ell)$ y, por lo tanto, los núumeros que representan las sublistas que interesan están entre 0 y ℓ. El problema es que además hay otros números que no interesan porque no representan sublistas de ℓ.

Para solucionar ésto, sea el predicado (primitivo recursivo) $esSublista(\ell', \ell) : \mathbb{N}^2 \to \{0, 1\}$ que decide si ℓ' es sublista de ℓ (en cualquier orden).

Luego, se tiene el rango de búusqueda para acotar el existencial y una condición p.r. lo su
cientemente fuerte para el nuevo rango. Podemos escribir:

$$subsetSum(l, k) = \exists_{0 \leq \ell' \leq \ell}\ esSublista(\ell', \ell) \wedge suma(\ell') = k$$

de donde es evidente que subsetSum es p.r. pues es composición de todas operaciones p.r.

Ejercicio 1.20.
Demostrar que toda clase PRC se encuentra cerrada por recursión sobre la cola de una lista. Es decir, dada $\mathcal{C}$, una clase PRC y dadas f y g funciones en $\mathcal{C}$ y una lista de $\mathbb{N}$, mostrar que también está en $\mathcal{C}$ la función h que cumple:

$$h(x_1, \ldots, x_n, []) = f(x_1, \ldots, x_n)$$
$$h(x_1, \ldots, x_n, [b, a_1, \ldots, a_m]) = g(h(x_1, \ldots, x_n, [a_1, \ldots, a_m]), x_1, \ldots, x_n, b)$$

Solución 1.20.
Recordemos la definición de recursión primitiva vista en la teórica.

$$h(x_1, \ldots, x_n, 0) = f(x_1, \ldots, x_n)$$
$$h(x_1, \ldots, x_n, t+1) = g(h(x_1, \ldots, x_n, t), x_1, \ldots, x_n, t)$$

Tambiéen recordemos que las listas están representadas por naturales, utilizando la representación de Godel vista en la teórica, y que sus observadores $|\ell|$ y $x[i]$ también están en toda clase PRC.

Una forma de mostrar que este esquema de recursión es válido en toda clase PRC es mostrar que respeta el esquema de recursión primitiva. Es decir, mostrar una traducción de manera tal que toda función escrita con recursión sobre la cola de una lista pueda ser escrita con recursión primitiva.

Para lograr eso, se debe observar las diferencias entre ambas definiciones. Luego, podemos de
nir una función h' que se comporte de manera similar a h, pero que utilice un esquema de recursión similar al esquema de recursión primitiva.

A simple vista puede verse que los esquemas parecen sustancialmente diferentes. Cuando utilizamos recursión primitiva, la recursión se realiza sobre el elemento inmediato anterior, mientras que en la recursión sobre la cola de una lista, la recursión se realiza sobre una lista diferente, pero la representación en $\mathbb{N}$ de ambas listas difiere en un valor que no necesariamente es 1.

Por lo tanto, se debe buscar una manera de que la recursión se realice en intervalos de uno. De esa manera, podríamos acercarnos a la definición de recursión primitiva. Notar entonces que en cada paso recursivo sobre la cola de la lista el tamaño de la misma disminuye en uno. Luego, se podría intentar realizar la recursión primitiva sobre la longitud de la lista a analizar.

Por lo tanto, deberíaa ser

$$h(x_1, \ldots, x_n, \ell) = h'(x_1, \ldots, x_n, \ell, |\ell|)$$

Si h' estuviera en PRC, h también, pues es composición de funciones primitivas recursivas.

Luego, se define una función h' que cumple lo pedido y además sea primitiva recursiva. El caso base debería ser la lista vacía según la definición de h, y 0 según la definión de recursión primitiva. Esto nos da la pauta de que el valor del último parámetro representa la longitud de la lista que estamos analizando.

También notamos que el esquema de recursión primitiva no nos permite modi
car ningún parámetro que no sea el último. Por lo tanto, a lo largo de las llamadas recursivas, la lista ℓ deberá mantenerse idéntica, pero debemos "simular" la modificación de la lista utilizando la variable sobre la que hacemos recursión. Como esta variable representa la longitud de la lista, esto se puede hacer tomando en cada caso los últimos t elementos de la lista utilizada.

Veamos entonces una primera aproximación a h'

$$h'(x_1, \ldots, x_n, \ell, 0) = f'(x_1, \ldots, x_n, \ell)$$
$$h'(x_1, \ldots, x_n, \ell, t+1) = g'(h'(x_1, \ldots, x_n, \ell, t), x_1, \ldots, x_n, \ell, t)$$

Si consideramos a h' como una función de $n+2$ parámetros, podemos notar que respeta el esquema de recursión primitiva, utilizando $x_1, \ldots, x_n, \ell$ como los parámetros que no se modi

can. Sólo falta probar que f' y g' son primitivas recursivas. Si esto fuera así, h' también sería primitiva recursiva. Veamos por último cómo se definen f' y g'.

Para respetar la definición de h debería ser $f'(x_1, \ldots, x_n, \ell) = f(x_1, \ldots, x_n)$, pero f es primitiva recursiva, por lo que f' también lo es. Por otra parte, si $g'(r, x_1, \ldots, x_n, \ell, t) = g(r, x_1, \ldots, x_n, \ell[|\ell| - t])$ se tiene que g' es primitiva recursiva por ser composición de de primitivas recursivas (g es primitiva recursiva).

Con ésto se probó que h' así de

nida es primitiva recursiva. Resta ver la relación entre h y h' para ver que efectivamente se cumple

$$h(x_1, \ldots, x_n, \ell) = h'(x_1, \ldots, x_n, \ell, |\ell|).$$

Para ello probaremos que:

$$h'(x_1, \ldots, x_n, \ell, t) = h(x_1, \ldots, x_n, ultimos(\ell, t)$$

donde la función $ultimos(\ell, t)$ devuelve los últimos t elementos de la lista ℓ con $t \leq \ell$.

Para probar esta propiedad haremos inducción en t.

- *Si $t = 0$,*

$$h'(x_1, \ldots, x_n, \ell, 0) = f'(x_1, \ldots, x_n, \ell) = f(x_1, \ldots, x_n) = h(x_1, \ldots, x_n, [])$$

- *Si la propiedad vale para t, veamos que vale para $t+1$,*

$$\begin{aligned} h'(x_1, \ldots, x_n, [c_1, \ldots, c_k, b, a_1, \ldots, a_t], t+1) &= g'(h'(x_1, \ldots, x_n)x_1, \ldots, x_n, \ell, t) \\ &= g'(h'(x_1, \ldots, x_n)x_1, \ldots, x_n, \ell[|\ell| - t]) \\ &\underbrace{=}_{HI} g(h(x_1, \ldots, x_n, ultimos(\ell, t)), x_1, \ldots, x_n, \ell[k+1+t-t]) \\ &= g(h(x_1, \ldots, x_n, [a_1, \ldots, a_t]), x_1, \ldots, x_n, b) \\ &= h(x_1, \ldots, x_n, [b, a_1, \ldots, a_t]) \end{aligned}$$

Sabiendo que la propiedad efectivamente vale, instanciamos t en la longitud de la lista para ver que

$$h(x_1, \ldots, x_n, \ell) = h'(x_1, \ldots, x_n, \ell, |\ell|)$$

es correcta:

$$h'(x_1, \ldots, x_n, \ell, |\ell|) = h(x_1, \ldots, x_n ultimos(\ell, |\ell|)) = h(x_1, \ldots, x_n, \ell)$$

que es lo que se quería probar.

Por lo tanto, acabamos de mostrar que si tenemos una función h que respeta el esquema de recursión "sobre la cola de una lista", también puede ser escrita utilizando el esquema de recursión primitiva. Luego, cualquier función h que respete este esquema de recursión es primitiva recursiva.

- *Otra forma de resolver el ejercicio:*

Intentando replicar la manera de resolver el problema de la función de Fibonacci, si

$$h'(\overline{x}, \ell, t) = h(\overline{x}, ultimos(\ell, t))$$

Luego, sólo resta ver que h' está en la clase PRC dada:

$$\begin{aligned} h'(\overline{x}, \ell, 0) &= f(\overline{x}) \\ h'(\overline{x}, \ell, t+1) &= h(\overline{x}, ultimos(\ell, t+1)) = g(h(\overline{x}, ultimos(\ell, t)), \overline{x}, \ell[|\ell| - t]) = g(h'(\overline{x}, \ell, t), \overline{x}, \ell[|\ell| - t]) \end{aligned}$$

Pero h' así definida respeta el esquema de recursión primitiva y es composición de funciones que sabemos están en $\mathcal{C}$. Por lo tanto, h' también lo está.

Ejercicio 1.21.
Utilicemos [] para referirnos a la codificación de secuencias dada en el punto 13.c.

a. Demostrar que toda clase PRC se encuentra cerrada por recursión global *(*course-of-values recursion*). Es decir, dada $\mathcal{C}$, una clase PRC, y dada una función $f : \mathbb{N}^{n+1} \to \mathbb{N}$ en $\mathcal{C}$, mostrar que la función definida como*

$$\begin{aligned} h(x_1, \ldots, x_n, 0) &= f([], x_1, \ldots, x_n) \\ h(x_1, \ldots, x_n, t+1) &= f([h(x_1, \ldots, x_n, 0), \ldots, h(x_1, \ldots, x_n, t)], x_1, \ldots, x_n) \end{aligned}$$

también está en $\mathcal{C}$.

Observar que h queda completamente determinada por el esquema de recursión global.

b. Demostrar, a partir del ítem anterior, que dada $\mathcal{C}$, una clase PRC, y funciones $g_1 : \mathbb{N}^n \to \mathbb{N}$, $g_2 : \mathbb{N}^{n+1} \to \mathbb{N}$ en $\mathcal{C}$, la función definida como

$$h(x_1, \ldots, x_n, 0) = g_1(x_1, \ldots, x_n)$$
$$h(x_1, \ldots, x_n, t+1) = g_2([h(x_1, \ldots, x_n, 0), \ldots, h(x_1, \ldots, x_n, t)], x_1, \ldots, x_n, t)$$

también está en $\mathcal{C}$.

Observar que h queda completamente determinada por el esquema de recursión global.

Solución 1.21.

Ejercicio 1.22.
Demostrar que toda clase PRC se encuentra cerrada por recursión doble. Es decir, dadas $f : \mathbb{N}^3 \to \mathbb{N}$ y $g : \mathbb{N}^4 \to \mathbb{N}$ pertenecientes a $\mathcal{C}$, una clase PRC, demostrar que también está en $\mathcal{C}$ la función $h : \mathbb{N}^3 \to \mathbb{N}$ que cumple:

$$h(x, 0, z) = f(x, 0, z)$$
$$h(x, y, 0) = f(x, y, 0)$$
$$h(x, y+1, z+1) = g(x, y, z, h(x, y, z))$$

Observar que h queda completamente determinada por el esquema de recursión doble.

Solución 1.22.

Práctica 2

Funciones $\mathcal{S}$-computables

Ejercicio 2.1.

a. Definir macros *para las siguientes pseudo-instrucciones (con su interpretación natural) e indicar en cada caso qué variables y qué etiquetas se asumen "frescas":*

- $V_i \leftarrow k$.
- $V_i \leftarrow V_j + k$.
- $IF\ V_i = 0\ GOTO\ L$.
- $GOTO\ L$.

b. Definir dos pseudo-programas distintos en el lenguaje $\mathcal{S}$ (usando las macros convenientes del punto anterior) que compute la función de dos variables $f_1(x_1, x_2) = x_1 + x_2$. Para alguno de los dos, expandir las macros utilizadas prestando atención a la instanciación de variables y etiquetas frescas.

c. Sea p el programa en $\mathcal{S}$ que resulta de expandir todas las macros en el código del punto anterior. Determinar cuál es las función computada en cada caso:

- $\Psi_p^{(1)} : \mathbb{N} \to \mathbb{N}$
- $\Psi_p^{(2)} : \mathbb{N}^2 \to \mathbb{N}$
- $\Psi_p^{(3)} : \mathbb{N}^3 \to \mathbb{N}$

Solución 2.1.

a. *(i)* $V_i \leftarrow k$,

$$
\begin{array}{ll}
[L] & V_i \leftarrow V_i - 1 \\
 & IF\ V_i \neq 0\ GOTO\ L \\
 & \left.\begin{array}{l} V_i \leftarrow V_i + 1 \\ V_i \leftarrow V_i + 1 \\ \quad\vdots \\ V_i \leftarrow V_i + 1 \end{array}\right\} k\ veces
\end{array}
$$

(ii) $V_i \leftarrow V_j + k$,

$$
\begin{array}{ll}
 & V_i \leftarrow 0 \\
[A] & IF\ V_j \neq 0\ GOTO\ B \\
 & GOTO\ C \\
[B] & V_j \leftarrow V_j - 1 \\
 & V_i \leftarrow V_i + 1 \\
 & Z \leftarrow Z + 1 \\
 & GOTO\ A \\
[C] & IF\ Z \neq 0\ GOTO\ D \\
 & GOTO\ E \\
[D] & Z \leftarrow Z - 1 \\
 & V_j \leftarrow V_j + 1 \\
 & GOTO\ C
\end{array}
$$

De esta manera, $V_i \leftarrow V_j$. Luego,

$$\boxed{\begin{array}{l} V_i \leftarrow V_j \\ \left.\begin{array}{l} V_i \leftarrow V_i + 1 \\ V_i \leftarrow V_i + 1 \\ \quad\vdots \\ V_i \leftarrow V_i + 1 \end{array}\right\} k \text{ veces} \end{array}}$$

(iii) *IF $V_i = 0$ GOTO L,*

$$\boxed{\begin{array}{l} IF\ V_i \neq 0\ GOTO\ B \\ GOTO\ L \end{array}}$$

(iv) *GOTO L,*

$$\boxed{\begin{array}{l} Z \leftarrow Z + 1 \\ IF\ Z \neq 0\ GOTO\ L \end{array}}$$

b.

(i)

$$\boxed{\begin{array}{ll} & Y \leftarrow X_1 \\ & Z \leftarrow X_2 \\ [A] & Y \leftarrow Y + 1 \\ & Z \leftarrow Z - 1 \\ & IF\ Z \neq 0\ GOTO\ A \end{array}}$$

(ii)

$$\boxed{\begin{array}{ll} & Y \leftarrow X_1 \\ & Z \leftarrow X_2 \\ [B] & IF\ Z \neq 0\ GOTO\ A \\ & GOTO\ E \\ [A] & Z \leftarrow Z - 1 \\ & Y \leftarrow Y + 1 \\ & GOTO\ B \end{array}}$$

c.

- $\Psi_p^{(1)} : \mathbb{N} \to \mathbb{N}$, *computa, por ejemplo* $\Psi_p^{(1)}(x) = k$.
- $\Psi_p^{(2)} : \mathbb{N}^2 \to \mathbb{N}$, *computa, por ejemplo* $\Psi_p^{(2)}(x_1, x_2) = x_1 + x_2$.
- $\Psi_p^{(3)} : \mathbb{N}^3 \to \mathbb{N}$, *computa, por ejemplo* $\Psi_p^{(2)}(x_1, x_2, x_3) = x_1 + x_2$.

Ejercicio 2.2.

a. Sea $\mathcal{C}_\mathcal{S} = \{\Psi_p^{(n)} \mid p$ es un programa en $\mathcal{S}, n \geq 1\}$ la clase de funciones $\mathcal{S}$-computables. Mostrar que $\mathcal{C}_\mathcal{S}$ es una clase PRC.

b. Demostrar (sin definir un programa en $\mathcal{S}$) que la función $* : \mathbb{N}^2 \to \mathbb{N}$ definida por $*(x, y) = x \cdot y$ es $\mathcal{S}$-computables.

c. Si $f : \mathbb{N}^n \to \mathbb{N}$ es una función primitiva recursiva. Qué podemos decir acerca de la existencia de un programa en el lenguaje $\mathcal{S}$ que la compute?

Solución 2.2.

a. Hay que ver que las funciones iniciales están en $\mathcal{C}_\mathcal{S}$ y que $\mathcal{C}_\mathcal{S}$ es cerrada por composición y por resursión primitiva.

- *Funciones iniciales:*
  - *$n(n)$, computa el programa vacío.*
  - *$s(x)$, compunta $Y \leftarrow Y + 1$.*
  - *$u_i(x_1, \ldots, x_n)$, computa $Y \leftarrow X_i$.*
- *Cerrado por Composición: Sean $f, g_1, \ldots, g_k \in \mathcal{C}_\mathcal{S}$, luego el programa P*

$$\begin{array}{l} Z_1 \leftarrow g_1(x_1, \ldots, x_n) \\ Z_2 \leftarrow g_2(x_1, \ldots, x_n) \\ Z_3 \leftarrow g_3(x_1, \ldots, x_n) \\ \vdots \qquad\quad \vdots \\ Z_k \leftarrow g_k(x_1, \ldots, x_n) \\ Y \leftarrow f(Z_1, \ldots, Z_k) \end{array}$$

computa $h(x_1, \ldots, x_n) = f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n))$, *entonces* $h \in \mathcal{C}_\mathcal{S}$.

- *Cerrado por Recursión Primitiva: Sean* $g \in \mathcal{C}_\mathcal{S}$, *luego el programa* P

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

computa h *que se obtiene por recursión primitiva a partir de* g, *entonces* $h \in \mathcal{C}_\mathcal{S}$.

b. Como $*(x, y) = x \cdot y$ *es una función pr, entonces está en toda clase PRC y, por lo tanto, en particular está en* $\mathcal{C}_\mathcal{S}$.

c. Entonces si f *es pr, se puede afirmar que* f *es* $\mathcal{S}$*-computable.*

Ejercicio 2.3.
Sea P *un programa que contiene exactamente una instrucción de la forma*

$$Y \leftarrow Y + 1$$

(etiquetada o no).

Demostrar que para todo $x \in \mathbb{N}$, *si* $\Psi_P^{(1)}(x) \downarrow$, *entonces al ejecutar* P *con* x *como única entrada se pasa por dicha instrucción por lo menos* $\Psi_P^{(1)}(x)$ *veces.*

Solución 2.3.
Intuitivamente, por un lado, sabemos que $\Psi_P^{(1)}(x)$, *el resultado de la ejecución del programa* P, *estará almacenado en la variable* Y *al finalizar dicha ejecución. Además, esta variable comienza valiendo* 0 *y solo puede ser incrementada en uno por una instrucción de la forma* $Y \leftarrow Y + 1$.

Como sólo hay una instrucción de este tipo en el programa, esta instrucción es la única responsable de llevar la variable Y *al valor* $\Psi_P^{(1)}(x)$. *Por lo tanto, tendrá que ser ejecutada al menos* $\Psi_P^{(1)}(x)$ *veces.*

Formalmente, recordemos que las descripciones instantáneas son una representación completa de cómo se encuentra la ejecución de un programa en un momento dado. El resultado del cómputo de un programa se define como el valor que tiene la variable Y *según la última de sus descripciones instantáneas. A su vez, cada descripción instantánea del cómputo está definida a partir de la anterior, partiendo desde una descripción inicial determinada por la misma semántica del lenguaje (las variables* $X_1, X_2, \ldots$ *contienen los valores de entrada, las demás variables comienzan en* 0, *etc.). Esto nos da una estructura inductiva para hacer razonamientos sobre la ejecución de los programas: si logramos demostrar que cierta propiedad vale para la descripción inicial, y que se preserva al pasar de una descripción instantánea a la siguiente, habremos probado que vale al finalizar.*

En nuestro caso, queremos probar que, al terminar de ejecutar el programa, la cantidad de veces que se pasó por la instrucción que incrementa la variable Y *es mayor o igual que el valor que tiene la variable* Y *en ese momento. Es importante notar que esta propiedad se puede generalizar a cualquier instante de la ejecución: el valor de la variable* Y *nunca puede superar la cantidad de veces que fue incrementada. Tratemos de llevar esto al esquema inductivo que acabamos de esbozar la ejecución.*

Dado cualquier $x \in \mathbb{N}$ *tal que* $\Psi_P^{(1)}(x) \downarrow$, *sea* $d_0, d_1, \ldots, d_n$ *el cómputo del programa* P *a partir de la entrada* $x \in \mathbb{N}$ *(*$d_0 = (1, \sigma_0)$ *es la descripción inicial correspondiente). Además, en adelante, llamaremos*

- m *al índice de la instrucción de* P *que incrementa la variable* Y.
- k_i *(para* $i = 0, \ldots, n$*) a la cantidad de veces que se pasó por la* m*-ésima instrucción de* P *tras* i *pasos de su ejecución.*
- $d_i[Y]$ *al valor de* Y *indicado por la descripción* d_i.

Lo que queremos demostrar es que, para todo $i = 0, \ldots, n$, *se cumple* $k_i \geq d_i[Y]$.

(C.B.) Por definición de estado inicial, $d_0[Y] = 0$. *También, claramente,* $k_0 = 0$. *Así, tenemos que* $k_0 \geq d_0[Y]$.

(P.I.) *Supongamos que vale que $k_i \geq d_i[Y]$, para $i \in \{0, \ldots, n\}$. Queremos ver que también $k_{i+1} \geq d_{i+1}[Y]$.*

Sabemos que $d_i = (j_i, \sigma_i)$, donde j_i es el índice de la próxima instrucción a ejecutar. Analicemos los casos posibles.

(i) *Si $j_i = m$, entonces*

$$k_{i+1} = k_i + 1 \geq d_i[Y] + 1 = d_{i+1}[Y]$$

porque tanto el valor de la variable Y como la cantidad de veces que se ejecutó la m-ésima instrucción se incrementan en 1.

(ii) *Si $j_i \neq m$, puede ser que la instrucción que se va a ejecutar directamente no involucre a la variable Y, que la involucre pero sea un salto condicional (que no modifica su valor), o bien que decremente su valor en 1. En los tres casos, $d_i[Y] \geq d_{i+1}[Y]$. Como además $k_{i+1} = k_i$, tenemos que*

$$k_{i+1} = k_i \geq d_i[Y] \geq d_{i+1}[Y].$$

Lo anterior demuestra que la propiedad que planteamos se preserva a lo largo de todos los pasos del programa, y en particular, que vale al final de la ejecución, que es lo que inicialmente queríamos probar.

Ejercicio 2.4.
Dado un programa P, existe un programa P' equivalente (es decir, $\Psi_P^{(n)} = \Psi_{P'}^{(n)}$ para todo n) tal que P' no tiene saltos a etiquetas que P' no contiene.

Solución 2.4.
Sea P' el siguiente programa

	P
$[E_1]$	$Z \leftarrow 0$
$[E_2]$	$Z \leftarrow 0$
$\vdots$	$\vdots$
$[E_k]$	$Z \leftarrow 0$

donde $E_1, \ldots, E_k$ son las etiquetas que aparecen en saltos pero no como etiquetas en P y Z es una variable que no aparece en P. Puede verse inmediatamente que $\Psi_P^{(n)} = \Psi_{P'}^{(n)}$ para todo n.

Ejercicio 2.5.
Decimos que un programa P es autocontenido *si en cada instrucción IF $V \neq 0$ GOTO L que ocurre en P, L es una etiqueta definida en P.*

a. *Demostrar que todo programa P tiene un programa autocontenido p' equivalente (P y P' son programas equivalentes si $\Psi_P^{(n)} = \Psi_{P'}^{(n)}$ para todo $n \in \mathbb{N}$).*

b. *Sean P y Q dos programas autocontenidos con etiquetas disjuntas y sea $r : \mathbb{N}^n \to \mathbb{N}$ un predicado primitivo recursivo. Definir macros para las siguientes pseudo-instrucciones (con su interpretación natural):*

- *IF $r(V_1, \ldots, V_n)$ GOTO L*
- *IF $r(V_1, \ldots, V_n)$ THEN p ELSE q*
- *WHILE $r(V_1, \ldots, V_n)$ P*

c. *Dadas las funciones $f, g : \mathbb{N} \to \mathbb{N}$ definidar por*

$$f(x) = \begin{cases} 1 & x = 3 \\ \uparrow & \text{en otro caso} \end{cases} \qquad y \qquad g(x) = 2x$$

Demostrar que es $\mathcal{S}$-parcial computable la función

$$h(x) = \begin{cases} f(x) & x \geq 3 \vee x = 3 \\ g(x) & \text{en otro caso} \end{cases}$$

Solución 2.5.

a. *Si P ya es un programa autocontenido, entonces es trivial.*

Supongamos ahora que existe al menos una instrucción IF $V \neq 0$ GOTO L tal que L no es una etiqueta de P. Por lo tanto, basta con agregar al final de la lista de instrucciones una como $V \leftrightarrow V$ con la etiqueta L. De manera similiar se pueden ir agregando instrucciones definiendo etiquetas y variables que no estén ya en el código del programa. Luego, el nuevo programa definido resulta equivalente a P y autocontenido.

b.

- *IF* $r(V_1, \ldots, V_n)$ *GOTO* L,

$$\begin{array}{l} X \leftarrow r(V_1, \ldots, V_n) \\ IF\ X \neq 0\ GOTO\ L \end{array}$$

- *IF* $r(V_1, \ldots, V_n)$ *THEN* P *ELSE* Q,

$$\begin{array}{l} X \leftarrow r(V_1, \ldots, V_n) \\ Y \leftarrow P \\ IF\ X \neq 0\ GOTO\ E \\ Y \leftarrow Q \end{array}$$

- *WHILE* $r(V_1, \ldots, V_n)$ P

c.

$$\begin{array}{ll} [A] & IF\ X \geq 5\ GOTO\ A \\ & Y \leftarrow 1 \\ & IF\ X = 3\ GOTO\ E \\ & Y \leftarrow 2 \cdot X \end{array}$$

Ejercicio 2.6.

a. Se definen las siguientes variantes del lenguaje $\mathcal{S}$.

- $\mathcal{S}_1$*: Igual que* $\mathcal{S}$ *pero sin la instrucción* $V \leftarrow V + 1$.
- $\mathcal{S}_2$*: Igual que* $\mathcal{S}$ *pero sin la instrucción IF* $V \neq 0$ *GOTO* L.
- $\mathcal{S}_3$*: Igual que* $\mathcal{S}$ *pero sin la instrucción* $V \leftarrow V \dot{-} 1$.

Demostrar que para cada uno de estos lenguajes existe al menos una función $\mathcal{S}$*-parcial computable que no es computable en este nuevo lenguaje.*

b. Sea $\mathcal{T}$ *el lenguaje de programación definido como* $\mathcal{S}$ *salvo que sus instrucciones (etiquetadas o no) son de los siguientes tres tipos (con su interpretación natural):*

$$\boxed{\begin{array}{l} V \leftarrow V + 1 \\ V \leftarrow V - 1 \\ IF\ V \neq 0\ GOTO\ L \end{array}}$$

Demostrar que una función es parcial computable en $\mathcal{T}$ *si solamente si lo es en* $\mathcal{S}$.

c. Sea $\mathcal{S}'$ *el lenguaje de programación definido como* $\mathcal{S}$ *salvo que sus instrucciones (etiquetadas o no) son de los siguientes tres tipos (con su interpretación natural):*

$$\boxed{\begin{array}{l} V \leftarrow V' \\ V \leftarrow V + 1 \\ IF\ V \neq V'\ GOTO\ L \end{array}}$$

Demostrar que una función es parcial computable en $\mathcal{S}'$ *si solamente si lo es en* $\mathcal{S}$.

Solución 2.6.

a. La idea de este inciso es verificar la minimalidad de $\mathcal{S}$.

$\mathcal{S}_1$: *La idea de la demostración es que si sacamos* $V \leftarrow V + 1$ *no podemos hacer de ninguna manera funciones que devuelvan algo mayor a lo que reciban. Por ejemplo, no podemos escribir* $f(x) = 1$ *(la función constante* 1*) porque no podemos incrementar* Y. *Para mostrarlo, tomemos un* P *cualquiera en este nuevo lenguaje acotado. Podemos hacer inducción en la cantidad de instrucciones de* P, *pero se puede hacer también analizando los posibles cómputos de* P. *Dado cualquier estado* (i, σ) *hay dos opciones: o bien la instrucción que toca realizar a continuación es* $V \leftarrow V - 1$, *o bien es IF* $V \neq 0$ *GOTO* L.

En el primer caso, pasaríamos a un nuevo estado $(i + 1, \sigma')$ *con* σ' *igual a* σ *sólo que con una variable disminuida en* 1. *Sea cual sea, está claro que* Y *no aumenta.*

En el segundo caso, nuestro siguiente estado puede ser $(i + 1, \sigma)$ *si* $V = 0$, *o* (n, σ) *si* $V \neq 0$ *y* n *es el número de instrucción con la primera aparición de la etiqueta* L *en* P, *o* $(|P| + 1, \sigma)$ *si* $V \neq 0$ *pero* L *no aparecía en* P.

En cualquiera de los tres casos σ *no cambia, y por lo tanto* Y *no se incrementa.*

$\mathcal{S}_2$: *Por ejemplo, no se puede computar la función* $f(x, y) = x + y$.

$\mathcal{S}_3$: *Por ejemplo, no se puede computar la función* $f(x) = x - 1$.

b. Para probarlo, alcanza con codificar la instrucción de $\mathcal{S}$ *que nos falta (IF* $V \neq 0$ *GOTO L) con las ya que tenemos.*

$$\begin{array}{ll} & IF\ V = 0\ GOTO\ E \\ & IF\ Z = 0\ GOTO\ L \\ [E] & Z \leftarrow Z - 1 \end{array}$$

Si $V = 0$*, la macro sale sin hacer nada (resta* 1 *a* Z *sóolo porque necesita una instrucción para ocupar con la etiqueta* E*). Si* $V \neq 0$*, hace un salto incondicional (porque sabemos de antemano que* $Z = 0$*) a la etiqueta* L*. Entonces, si tomamos como frescas la etiqueta* E *y la variable* Z *(que empieza en* 0*), esta es una macro de la instrucción IF* $V \neq 0$ *GOTO L como queríamos.*

Ejercicio 2.7.

a. Demostrar que si $p : \mathbb{N}^{n+1} \to \{0, 1\}$ *es un predicado* $\mathcal{S}$*-computable (total), entonces es* $\mathcal{S}$*-parcial computable:*

$$\text{minimoNA}_p(x_1, \ldots, x_n, y) = \begin{cases} \text{mín}\{t \mid y \leq t \wedge p(x_1, \ldots, x_n, t)\} & \text{si existe algún tal } t \\ \uparrow & \text{en otro caso} \end{cases}$$

b. Mostrar, usando el resultado anterior, que si $f : \mathbb{N} \to \mathbb{N}$ *es biyectiva y* $\mathcal{S}$*-computable (total), entonces también lo es su inversa* f^{-1}.

Solución 2.7.

Ejercicio 2.8.
Sea $p : \mathbb{N}^n \to \{0, 1\}$ *un predicado* $\mathcal{S}$*-computable, entonces el existencial no acotado sobre el predicado,*

$$(\exists t)p(x_1, x_2, \ldots, x_n, t) = \begin{cases} 1 & \text{si existe } t \text{ tal que } p(x_1, x_2, \ldots, t) = 1 \\ \uparrow & \text{si no} \end{cases}$$

es $\mathcal{S}$*-parcial computable.*

Solución 2.8.
Para probarlo, podemos escribir un programa en $\mathcal{S}$ *que compute dicho predicado.*

La idea es recorrer todos los posibles valores de t *y, para cada uno de ellos, computar* p*; si existe un valor para el que sea verdadero, en algún momento lo encontraremos y devolveremos* 1*, mientras que si no existe tal valor, el programa continuará buscándolo de manera indefinida.*

$$\begin{array}{ll} [A] & Z_2 \leftarrow p(X_1, X_2, \ldots, X_n, Z_1) \\ & IF\ Z_2 \neq 0\ GOTO\ B \\ & Z_1 \leftarrow Z_1 + 1 \\ & GOTO\ A \\ [B] & Y \leftarrow 1 \end{array}$$

Es importante tener en cuenta que, para que este programa sea válido, el predicado p *debe ser total. En caso contrario, la ejecución podría quedar atrapada en el cómputo de* p *para algún valor de* t *e indefinir el existencial, incluso aunque* p *sea verdadero para otro valor de* t *que no todavía no hayamos verificado.*

- *Otra manera de probarlo:*

Otra manera de computar el existencial no acotado es reutilizar la minimización no acotada vista en las teóricas, es decir, la función

$$\min_t p(x_1, x_2, \ldots, x_n, t)$$

que es $\mathcal{S}$*-parcial computable siempre que* p *sea un predicado* $\mathcal{S}$*-computable. El siguiente programa computa el existencial para un predicado* p *a partir de esta minimización*

$$\begin{array}{l} Z_1 \leftarrow p(x_1, x_2, \ldots, x_n, t) \\ Y \leftarrow 1 \end{array}$$

Ejercicio 2.9.
Demostrar que la siguiente función es $\mathcal{S}$-parcial computable

$$f_1(x,y,z) = \begin{cases} 1 & \text{si la ejecución de } \Phi_x^{(1)}(z) \text{ termina estrictamente en menos pasos que la ejecución de } \Phi_x^{(1)}(y) \\ \uparrow & \text{si no} \end{cases}$$

$$f_2(x,y,z) = \begin{cases} 1 & \text{si } \Phi_x^{(1)} \text{ tiene algún punto fijo} \\ \uparrow & \text{si no} \end{cases}$$

$$f_3(x,y,z) = \begin{cases} 1 & \text{si existe } z \in \mathbb{N} \text{ tal que, durante la ejecución de } \Phi_x^{(1)}(z) \\ & \qquad \text{la variable } Y \text{ alcanza un valor mayor que } y \\ \uparrow & \text{si no} \end{cases}$$

$$f_4(x,y,z) = \begin{cases} 1 & \text{si existe } z \in \mathbb{N}^n \text{ tal que } \Phi_x^{(n)}(z)\downarrow,\ \Phi_y^{(n)}(z)\downarrow \text{ y } \Phi_x^{(n)}(z) \neq \Phi_y^{(n)}(z) \\ \uparrow & \text{si no} \end{cases}$$

Solución 2.9.

1. *La idea es utilizar la función $STP^{(1)}$ para seguir el rastro del programa con número x, y así descubrir cuántos pasos tarda en terminar de ejecutar, si es que en algún momento lo hace.*

   Luego, podemos usar nuevamente $STP^{(1)}$ para averiguar si el programa con número y termina de ejecutar en una cantidad menor o igual de pasos. Nuestro programa debe devolver 1 si y sólo si esto último resulta ser falso.

   Como el predicado $STP^{(1)}$ es primitivo recursivo, es también $\mathcal{S}$-computable, así que podemos usarlo directamente dentro de nuestros programas.

$[A]$	$Z_2 \leftarrow STP^{(1)}(X_3, X_1, Z_1)$
	$IF\ Z_2 \neq 0\ GOTO\ B$
	$Z_1 \leftarrow Z_1 + 1$
	$GOTO\ A$
$[B]$	$Z_2 \leftarrow STP^{(1)}(X_3, X_2, Z_1)$
	$IF\ Z_2 \neq 0\ GOTO\ B$
	$Y \leftarrow 1$

   Las primeras cuatro instrucciones del programa conforman un ciclo, que se repite hasta que la variable Z_1 contiene exactamente la cantidad de pasos que tarda en terminar el programa con número X_1 cuando recibe la entrada X_3. Si, para esta entrada, el programa X_1 no termina, nuestro programa queda atrapado en este ciclo y se indefine, cumpliendo con el comportamiento esperado para este caso.

   Si el ciclo termina, se ejecuta la quinta instrucción, que evalúa la terminación del programa con número X_2 tras la cantidad de pasos que quedó almacenada en Z_1. Si el resultado es 0, entonces X_2 tarda estrictamente más en terminar que X_1 (pudiendo, quizá, no terminar nunca), por lo que devolvemos 1. En caso contrario, es decir, si X_2 termina en una cantidad de pasos menos o igual que X_1, nuestro programa se indefine, quedando atrapado en un ciclo infinito entre las instrucciones quinta y sexta.

   - *Otra manera de probarlo: Las siguientes son dos condiciones necesarias para que $f_1(x,y,z) = 1$:*

   a. *Primero hay que notar que el programa con número x termina para la entrada z. Es decir, debe existir algún $t \in \mathbb{N}$ tal que $STP^{(1)}(z,x,t)$ sea verdadero.*
   b. *Y, por otra parte, el programa con número y, para la entrada z, no termina antes ni al mismo tiempo que el programa x para la misma entrada. Equivalentemente, debe existir algún tiempo t para el cual el programa x ya haya terminado, pero el programa z todavía no; o, más formalmente, existe algún $t \in \mathbb{N}$ que cumple $STP^{(1)}(z,x,t) \wedge \neg STP^{(1)}(z,y,t)$.*

   Dado que (b) implica (a), sólo tendremos en cuenta la última condición.

   Es sencillo advertir que (b) también es condición suficiente para que $f(x,y,z) = 1$. En caso contrario, o bien la ejecución de $\Phi_x^{(1)}(z)$ no termina o, para todo t tal que la ejecución de $\Phi_x^{(1)}(z)$ termina en t o menos pasos, la ejecución de $\Phi_y^{(1)}(z)$ también termina en t o menos pasos; es decir, la ejecución de $\Phi_y^{(1)}(z)$ termina en una cantidad de pasos menor o igual que la ejecución de $\Phi_x^{(1)}(z)$. En cualquiera de estas dos situaciones, $f_1(x,y,z)\uparrow$. Por lo tanto, podemos reescribir

$$f_1(x,y,z) = (\exists t)\left(STP^{(1)}(z,x,t) \wedge \neg STP^{(1)}(z,y,t)\right)$$

   donde el predicado al que hace referencia el existencial es primitivo recursivo y, por lo tanto, $\mathcal{S}$-computable (total), de donde se sigue que f_1 es $\mathcal{S}$-parcial computable.

2. *Podemos ver que una solución satisfactoria a este problema debería permitirnos recorrer, en algún orden, todas las combinaciones posibles del tipo*

   (valor de entrada, cantidad de pasos de la ejecución)

   sin dejar afuera ninguna de ellas. Ahora bien, estas combinaciones no son más que pares de números naturales. Recorrerlos en forma ordenada equivale a poner estos pares en biyección con $\mathbb{N}$ usando la función codificadora de pares. En otras palabras, como a cada par le corresponde un único número natural, si recorremos primero el par que se codifica con 0, luego el par que se codifica con 1, a continuación el que se codifica con 2, y así sucesivamente, la biyectividad de la codificación nos permite estar seguros de que vamos a recorrerlos todos.

   Partiendo de estas ideas, es posible escribir un programa en $\mathcal{S}$ que compute f_2. Sin embargo, en lugar de hacer esto, tratemos de aplicarlas para llevar el problema a la forma de un existencial no acotado. Una manera sencilla de escribir como existencial la propiedad expresada por f_2 es

   $$(\exists y)\left(\Phi_x^{(1)}(y)\downarrow \wedge \Phi_x^{(1)}(y) = y\right) \equiv (\exists y)\left[(\exists t)\left(STP^{(1)}(y,x,t) \wedge r(SNAP^{(1)}(x,y,t))[1] = y\right)\right]$$

   Ahora bien, no podemos estar seguros de que este existencial sea $\mathcal{S}$-parcial computable, porque no está construido a partir de un predicado $\mathcal{S}$-computable: la ejecución del programa x, para alguna entrada y, podría indefinirse.

   Por lo tanto, la solución pasa por utilizar la codificación de pares para iterar simultáneamente sobre las variables y y t. Esto nos permite reescribir f_2, a partir de la expresión anterior, utilizando un único existencial

   $$(\exists \langle y,t\rangle)\left(STP^{(1)}(y,x,t) \wedge r(SNAP^{(1)}(y,x,t))[1] = y\right)$$

   de donde, utilizando las funciones observadoras de pares, podemos reescribir a f_2 en la forma de un existencial no acotado:

   $$f_2(x) = (\exists z)\left[STP^{(1)}(\ell(z),x,r(z)) \wedge r\left(SNAP^{(1)}(\ell(z),x,r(z))\right)[1] = \ell(z)\right]$$

3. *Escribiendo más formalmente la propiedad expresada por f_3, nos queda algo como*

   $$(\exists z)\left((\exists t)\left(r(SNAP^{(1)}(z,x,t))[1] > y\right)\right)$$

   Notar que esta vez no se necesitó usar $STP^{(1)}$ para verificar que el programa haya terminado, ya que alcanza con que la propiedad valga en un momento cualquiera de la ejecución.

   Como el existencial escrito de esa forma no está bien construido, ya que no hace referencia a un predicado $\mathcal{S}$-computable (total), sino a otro existencial, que podría indefinirse. Luego, recorriendo simultáneamente los valores posibles de z y t gracias a la codificación de pares, podemos rescribir f_3 como

   $$f_3(x,y) = (\exists w)\left(r\left(SNAP^{(1)}(\ell(w),x,r(w))\right)[1] > y\right)$$

   Y, finalmente, como el existencial no acotado está aplicado sobre un predicado primitivo recursivo, está correctamente formado y podemos afirmar que es $\mathcal{S}$-parcial computable.

4. *Haremos la demostración para un $n \geq 1$ cualquiera.*

   Es posible llevar la propiedad a la que hace referencia f_4 a la forma de un existencial. En este caso, una traducción casi directa del enunciado nos deja con

   $$(\exists z)\left(\Phi_x^{(n)}(z)\downarrow, \Phi_y^{(n)}(z)\downarrow \text{ y } \Phi_x^{(n)}(z) \neq \Phi_y^{(n)}(z)\right)$$

   Reescribiendo la condición del existencial en términos de las funciones $STP^{(n)}$ y $SNAP^{(n)}$, obtenemos

   $$(\exists z)\left[(\exists t)\left(STP^{(n)}(z,x,t) \wedge STP^{(n)}(z,y,t) \wedge r(SNAP^{(n)}(z,x,t))[1] \neq r(SNAP^{(n)}(z,y,t))[1]\right)\right]$$

   El problema de esta expresión es el mismo que se nos presentaba antes: tenemos dos existenciales anidados. A esto se le suma el hecho de que el existencial más externo no predica sobre un número natural, sino sobre un elemento de $\mathbb{N}^n$. En definitiva, en la expresión entran en juego $n+1$ variables.

   La manera de resolverlo es, nuevamente, iterar sobre todas las variables simultáneamente utilizando la codificación de tuplas. En este caso, no podemos usar pares, sino que necesitaremos $n+1$-uplas. Esto no es un problema, dado que, para cualquier $n \geq 1$, las $n+1$-uplas están en biyección con $\mathbb{N}$ y, más aún, las funciones codificadoras y observadoras de $n+1$-uplas son primitivas recursivas (Ejercicio de la práctica 1).

A partir de esta idea, podemos reescribir la expresión anterior como un único existencial no acotado de la siguiente manera:

$$(\exists\langle z_1, z_2, \ldots, z_n, t\rangle)\Big(STP^{(n)}(z_1, z_2, \ldots, z_n, x, t) \wedge STP^{(n)}(z_1, z_2, \ldots, z_n, y, t) \\ \wedge\, r(SNAP^{(1)}(z_1, z_2, \ldots, z_n, x, t))[1] \neq r(SNAP^{(1)}(z_1, z_2, \ldots, z_n, y, t))[1]\Big)$$

de donde, formalizando un poco más, podemos reescribir a f_4 como

$$f_4(x,y) = (\exists w)\Big[STP^{(n)}(\pi_1(w), \ldots, \pi_n(w), x, \pi_{n+1}(w)) \wedge STP^{(n)}(\pi_1(w), \ldots, \pi_n(w), y, \pi_{n+1}(w)) \wedge \\ r\left(SNAP^{(n)}(\pi_1(w), \ldots, \pi_n(w), x, \pi_{n+1}(w)\right)[1] \neq r\left(SNAP^{(n)}(\pi_1(w), \ldots, \pi_n(w), y, \pi_{n+1}(w)\right)[1]\Big]$$

donde $\pi_i : \mathbb{N} \to \mathbb{N}$ es la función observadora de la i-ésima componente de una $n+1$-upla, para todo $i \in \{1, \ldots, n+1\}$.

Como el existencial no acotado hace referencia a un predicado primitivo recursivo, podemos afirmar que f_4 es una función $\mathcal{S}$-parcial computable, como queríamos probar.

Ejercicio 2.10.
Sea P un programa en el lenguaje $\mathcal{S}$ con instrucciones $I_1, I_2, \ldots, I_n$.

(i) P se dice mala onda si no tiene instrucciones de la form

$$X_i \to X_i + 1$$

Es decir, no aumenta el valor de las entradas.

Demostrar que el siguiente predicado es primitivo recursivo:

$$r_1(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ es mala onda} \\ 0 & \text{caso contario} \end{cases}$$

(ii) P se dice optimista si $\forall i = 1, \ldots, n$, si I_i es la instrucción IF $V \neq 0$ GOTO L entonces L no aparece como etiqueta de ninguna instrucción I_j con $j \leq i$.

Demostrar que el siguiente predicado es primitivo recursivo:

$$r_2(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ es optimista} \\ 0 & \text{caso contario} \end{cases}$$

(iii) P se cuelga si posee dos instrucciones consecutivas de la forma

	$V \leftarrow V + 1$
$[L]$	IF $V \neq 0$ GOTO L

Demostrar que el siguiente predicado es primitivo recursivo:

$$r_3(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ se cuelga} \\ 0 & \text{caso contario} \end{cases}$$

(iv) P tiene un salto al final si posee alguna instrucción del tipo

$$\text{IF } V \neq 0 \text{ GOTO } L$$

tal que ninguna de las instrucciones de P está etiquetada con L.

Demostrar que el siguiente predicado es primitivo recursivo:

$$r_4(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ tiene un salto al final} \\ 0 & \text{caso contario} \end{cases}$$

(v) P respeta sus entradas si no tiene instrucciones de los tipos

$$X_i \leftarrow X_i + 1 \text{ o } X_i \leftarrow X_i \dot{-} 1$$

para ningún $\mathbb{N}$.

Demostrar que el siguiente predicado es primitivo recursivo:

$$r_5(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ respeta sus entradas} \\ 0 & \text{caso contario} \end{cases}$$

Solución 2.10. *En general, la idea del ejercicio es "decodificar" el programa cuyo número recibimos por parámetro, para así poder analizar sus instrucciones. Sabemos que si el programa con número x consiste en las instrucciones $I_1, I_2, I_3, \ldots, I_k$, entonces x satisface*

$$x + 1 = [\#I_1, \#I_2, \#I_3, \ldots, \#I_k]$$

y, además, las instrucciones de un programa se codifican en la forma

$$\langle a, \langle b, c\rangle\rangle$$

donde a codifica la etiqueta de la instrucción, b codifica el tipo de instrucción y c, la variable involucrada.

(i) Buscamos que para toda línea, si es del tipo "incrementar 1", la variable a la que hace referencia sea una variable en posición impar (recordemos que se ordenan como $Y, X_1, Z_1, X_2, Z_2, \ldots$).

$$r_1(x) = \forall_{t\leq|x+1|}\left[\ell(r((x+1)[t])) = 1 \to (1 - par(r(r((x+1)[t]))))\right]$$

Notar que siempre se mira $x + 1$ es porque la codificación de programas le resta 1 a la lista.

(ii)

(iii) Basta con notar que el predicado es un existencial acotado sobre t (menor al largo del programa menos uno) que cumpla lo siguiente:

- *La instruccción t es de la forma $V \leftarrow V + 1$. Esto es: $\ell(r((x+1)[t])) = 1$.*
- *La instruccción $t+1$ es de la forma $[L]$ IF $V \neq 0$ GOTO L'. Esto es: $\ell[r((x+1)[t+1])] > 2$.*
- *$L' = L$. Es decir, $\ell(\ell((x+1)[t+1])) = \ell(r(x+1)[t+1]) - 2$.*
- *$V = W$. Se traduce en $r(r((x+1)[t])) = r(r((x+1)[t+1]))$.*

Juntando todo

$$\begin{aligned} r_3(x) = \exists_{t\leq|x+1|-1}[&\ell(r((x+1)[t])) = 1] \wedge [\ell(r((x+1)[t+1])) > 2] \wedge \\ &\wedge [\ell)\ell((x+1)[t+1])) = \ell(r(x+1)[t+1]) - 2] \wedge \\ &\wedge [r(r((x+1)[t])) = r(r((x+1)[t+1]))] \end{aligned}$$

(iv) Para saber si una instrucción es un salto al final, alcanza con conocer el valor de b, que podemos obtener a partir de una instrucción y como $\ell(r(y))$. Sabemos que la instrucción y es del tipo IF $V \neq 0$ GOTO L si y sólo si $\ell(r(y)) \geq 3$ y, que, en tal caso $\#(L) = \ell(r(y)) - 2$.

Además, debemos verificar si la etiqueta L en cuestión aparece o no en alguna instrucción del programa. Podemos hacerlo inspeccionando el valor de a en la codificación de cada una de las instrucciones; este valor se obtiene, para cada instrucción $\tilde{y}$, como $\ell(\tilde{y})$. En conclusión, el predicado r_4 puede escribirse como

$$g(x, y) = \ell(r(y)) \geq 3 \wedge \neg(\exists)_{t\leq|x+1|}\,(t > 0 \wedge \ell((x+1)[t]) = \ell(r(y)) \dot{-} 2)$$

Es decir, la instrucción codificada por y es un salto, y no existe ningún t tal que la t-ésima instrucción del programa cuyo número es x tiene la misma etiqueta a la que apunta el salto codificado por y.

(v) Sea el predicado

$$\tilde{r}(x) = \begin{cases} 1 & \text{si la instrucción codificada por } y \text{ respeta las entradas} \\ 0 & \text{caso contario} \end{cases}$$

Para determinar si una instrucción y codificada en la forma $y = \langle a, \langle b, c\rangle\rangle$ respeta las entradas, debemos tener en cuenta:

- *El tipo de instrucción, es decir, el valor de b, que se obtiene como $\ell(r(y))$. Una instrucción que no respeta las entradas tendrá el valor 1 (correspondiente a $V \leftarrow V + 1$) o el valor 2 (que codifica $V \leftarrow N \dot{-} 1$).*
- *La variable involucrada, es decir, el valor de c, que se obtiene como $r(r(y))$. Una instrucción que no respeta las entradas tendrá aquí un número correspondiente a una variable X_i. Como las variables se enumeran $Y, X_1, X_1, X_2, X_2, \ldots$, las variables X_i son todas aquellas que ocupan posiciones pares. No obstante, como el número de la variable se disminuye en 1 al codificar la instrucción, las instrucciones que no respetan las entradas tendrán un valor de c impar*

Es decir, la instrucción codificada por y no respeta las entradas si y sólo si $\ell(r(y)) \in \{1, 2\}$ y, simultáneamente, $r(r(y))$ es un número impar. Podemos, entonces, escribir $\tilde{r}$ como sigue:

$$\tilde{r} = \neg\left[(\ell(r(y)) = 1 \vee \ell(r(y)) = 2) \wedge impar(r(r(y)))\right]$$

y, en definitiva,

$$r_5(x) = \forall_{t\leq|x+1|}\,(t = 0 \vee \tilde{r}((x+1)[t]))$$

Ejercicio 2.11.
Utilizando las funciones primitivas-recursivas $STP^{(n)}$ y $SNAP^{(n)} : \mathbb{N}^{n+2} \to \mathbb{N}$ vistas en clase, mostrar que las siguientes son funciones $\mathcal{S}$-parciales computables:

$$f_1(x,y) = \begin{cases} 1 & y \in Dom\Phi_x^{(1)} \\ \uparrow & si\ no \end{cases} \qquad f_2(x) = \begin{cases} 1 & Dom\Phi_x^{(1)} \neq \emptyset \\ \uparrow & si\ no \end{cases}$$

$$f_3(x,y) = \begin{cases} 1 & y \in Im\Phi_x^{(1)} \\ \uparrow & si\ no \end{cases} \quad f_4(x,y) = \begin{cases} 1 & Dom\Phi_x^{(1)} \cap Dom\Phi_y^{(1)} \neq \emptyset \\ \uparrow & si\ no \end{cases}$$

$$f_5(x,y) = \begin{cases} 1 & si\ \Phi_x^{(1)}(y)\downarrow\ y\ \Phi_x^{(1)}(y) < x \\ \uparrow & si\ no \end{cases} \quad f_6(x,y) = \begin{cases} 1 & si\ existe\ z\ tal\ que\ \Phi_x^{(1)}(z)\downarrow,\ \Phi_y^{(1)}(z)\downarrow\ y\ \Phi_x^{(1)}(z) = \Phi_y^{(1)}(z) \\ \uparrow & si\ no \end{cases}$$

Comentario:
Recordar que

- *$STP^{(n)}(x_1,\ldots,x_n\#P,t)$ es un predicado p.r. (para cada $n>0$) que dice si el programa P termina en tiempo menor o igual a t con entradas $x_1,\ldots,x_n$.*
- *$\langle i,\sigma\rangle = SNAP^{(n)}(x_1,\ldots,x_n\#P,t)$ es una función p.r. (para cada $n>0$) que devuelve la descripción instantánea del programa después de t pasos. De esta manera, i representa el número de línea que hay que ejecutar y σ es una lista con los valores de todas las variables que aparecen en P después de t pasos $[Y,X_1,Z_1,X_2,Z_2,\ldots]$.*

Solución 2.11.

1.
2.
3.
4.
5. *Basta con notar que*
$$g(x,y) = \min_t \left(STP^{(2)}(y,x,t) \wedge r(SNAP^{(n)}(x,y,t))[1] < x \right)$$
es computable (es p.r. puesto que el argumento es p.r.). Luego, la minimización es parcial computable.
Finalmente, considerando la función computable $h(x) = 1$,, computo $f_5(x,y) = h(g(x,y))$.
6. *Basta con notar que*
$$g(x,y) = \min_{\langle t,z\rangle} \left[STP^{(1)}(z,x,t) \wedge STP^{(1)}(z,y,t) \wedge r(SNAP^{(1)}(z,x,t))[1] = r(SNAP^{(1)}(z,y,t))[1] \right]$$
Finalmente, considerando la función computable $h(x) = 1$,, computo $f_6(x,y) = h(g(x,y))$.

Ejercicio 2.12.
Sea $f : \mathbb{N} \to \mathbb{N}$ una función $\mathcal{S}$-parcial computable en tiempo polinomial (i.e., existe un programa P tal que $\Psi_P^{(1)}(x) = f(x)$ y tal que, para algún polinomio $Q(x)$, P no requiere más que $Q(\lfloor \log_2 x \rfloor)$ pasos para terminar).

a. Mostrar que f es primitiva recursiva.

b. >Sucede lo mismo si la cota es exponencial, doblemente exponencial, etc.?

c. >Qué podemos decir, en general, sobre la complejidad temporal de una función computable que no sea primitiva recursiva?

Solución 2.12.

a. Sea $T(x) = \min_{t \leq Q(\lfloor \log_2 x \rfloor)} STP(x, \sharp P, t) = 1$ (es decir, buscamos el mínimo tiempo en que el programa P se detiene con entrada x), luego
$$f(x) = r(SNAP(x, \sharp P, T(x)))[1]$$
(recordar que $r(SNAP(x,\sharp P,T(x)))$ es el estado final del programa P con la entrada x y, por lo tanto, $r(SNAP(x,\sharp P,T(x)))[1]$ es el valor de la variable Y en ese estado final).
Notar que T es pr por ser composición de pr (la minimización acotada es pr, Q es pr por ser un polinomio, $\lfloor \log_2 x \rfloor$ es pr y STP es pr). Luego, f es pr por ser composición de pr (puesto que T y $SNAP$ son pr).

b. Siempre que la cota sea pr, f resulta pr.

Ejercicio 2.13.
Se dice que un programa P en el lenguaje $\mathcal{S}$ se pisa con n entradas si para alguna entrada $x_1, x_2, \ldots, x_n$ y algún tiempo t, la variable de salida Y luego de t pasos de la ejecución de P con entradas $x_1, x_2, \ldots, x_n$ vale $\sharp P$. Demostrar que para cualquier $n \in \mathbb{N}$ es $\mathcal{S}$-parcial computable la función:

$$f_n(x) = \begin{cases} 1 & \text{si el programa cuyo número es } x \text{ se pisa con } n \text{ entradas} \\ \uparrow & \text{caso contario} \end{cases}$$

Práctica 3

Funciones no-computables y conjuntos c.e.

Ejercicio 3.1.
Sean dos funciones $f : \mathbb{N}^n \to \mathbb{N}$ y $g : \mathbb{N}^m \to \mathbb{N}$. Decimos que la función f se reduce a g si existen funciones computables (totales) $H : \mathbb{N} \to \mathbb{N}$ y $h_1, \ldots, h_m : \mathbb{N}^n \to \mathbb{N}$ tales que

$$f(x_1, \ldots, x_n) = H(g(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n)))$$

Probar que si f se reduce a g y g es parcial computable, entonces f es parcial computable.

Solución 3.1.
Si f se reduce a g, entonces f se escribe como una composición de funciones parciales computables. Como la clase de funciones parciales computables es PRC, entonces f también es parcial computable.

Ejercicio 3.2.
Probar, usando una diagonalización, que las siguientes funciones no son computables:

$$f_1(x,y) = \begin{cases} 1 & \text{si } Halt(x,y) \\ 0 & \text{en otro caso} \end{cases} \qquad f_6(x,y) = \begin{cases} 1 & \Phi_x^{(1)}(y)\downarrow \text{ y } \Phi_x^{(1)}(y) = 0 \\ 0 & \text{en otro caso} \end{cases}$$

$$f_2(x,y,z) = \begin{cases} 1 & \Phi_x^{(1)}(y)\downarrow \text{ y } \Phi_x^{(1)}(y) > z \\ 0 & \text{en otro caso} \end{cases} \qquad f_7(x) = \begin{cases} 1 & \Phi_x^{(1)}(x)\downarrow \text{ y } \Phi_x^{(1)}(x) \neq x \\ 0 & \text{en otro caso} \end{cases}$$

$$f_3(x,y) = \begin{cases} 1 & \Phi_x^{(1)}(y)\downarrow \text{ y es par} \\ 0 & \text{en otro caso} \end{cases} \qquad f_8(x,y,z) = \begin{cases} \Phi_z^{(1)}(z) & \Phi_{x+y}^{(1)}(x \dot{-} y)\downarrow \\ 0 & \text{en otro caso} \end{cases}$$

$$f_4(x,y) = \begin{cases} 3x & \text{si } \Phi_x^{(1)}(x) = x \\ 2x & \text{en otro caso} \end{cases} \qquad f_9(x,y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}xy)\uparrow \text{ ó } \Phi_x^{(1)}(x) \leq 2014 \\ 0 & \text{en otro caso} \end{cases}$$

$$f_5(x,y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y)\uparrow \text{ ó } \Phi_x^{(1)}(y)\downarrow \\ 0 & \text{en otro caso} \end{cases} \qquad f_{10}(x,y,z,w) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y)\downarrow \text{ y } \Phi_x^{(1)}(x)\downarrow \text{ y } (\Phi_x^{(1)}(z) + \Phi_x^{(1)}(y) = y + z) \\ 0 & \text{en otro caso} \end{cases}$$

Solución 3.2.

1. *Supongamos que f_1 es computable, entonces $f_1(x,x)$ también es computable.*
   Luego, sea P el programa que computa,

$$\begin{array}{ll} & Y \leftarrow f_1(x,x) \\ [A] & IF\ Y \neq 0\ GOTO\ A \end{array}$$

   y e el código del programa, entonces $\forall x$ se tiene que

$$\Psi_P(x)\downarrow \Leftrightarrow f_1(x,x) = 0 \Leftrightarrow \Phi_e^{(1)}(x)\uparrow$$

   luego, si $x = e$

$$\Phi_e^{(1)}(e)\downarrow = \Psi_P(e)\downarrow \Leftrightarrow f_1(e,e) = 0 \Leftrightarrow \Phi_e^{(1)}(e)\uparrow, \text{ lo que resulta una contradicción.}$$

   Por lo tanto, f_1 no es computable.

2. *Supongamos que f_2 es computable, entonces la función (predicado) $h(x) = f_2(x,x,x)$ también es computable.*
   Sea $g(x) = \begin{cases} x-1 & h(x)=1 \\ x+1 & h(x)=0 \end{cases}$ *y e el código del programa que la computa.*
   Si $h(e)=1 \Rightarrow g(e)\downarrow$ y $g(e) > e$, pero $g(e) = e-1 < e$. Lo que resulta en una contradicción.
   Si $h(e)=0 \Rightarrow g(e)\uparrow$ o bien $g(e) \leq e$, pero $g(e) = e+1 > e$. Lo que resulta en una contradicción.
   Luego, h no es computable y, por lo tanto, f_2 tampoco lo es.

3. *Supongamos que f_3 es computable, entonces la función (predicado) $h(x) = f_3(x,x)$ también es computable.*
   Sea $g(x) = \begin{cases} 1 & h(x)=1 \\ 2 & h(x)=0 \end{cases}$ *y e el código del programa que la computa.*
   Si $h(e)=1 \Rightarrow g(e)\downarrow$ y g es par, pero $g(e)=1$. Lo que resulta en una contradicción.
   Si $h(e)=0 \Rightarrow g(e)\uparrow$ o bien g es impar, pero $g(e)=2$. Lo que resulta en una contradicción.
   Luego, h no es computable y, por lo tanto, f_3 tampoco lo es.

4. *Supongamos que f_4 es computable y sea P el programa que la computa.*
   Sea P' definido como:

	P
	$Y \leftarrow Y - 2 * X$
$[L]$	$IF\ Y \neq 0\ GOTO\ L$
	$Y \leftarrow X$

   donde L es una etiqueta fresca (es decir, que no aparece en P).
   Luego,

$$\Psi_{P'}(x)\downarrow \Leftrightarrow \Psi_P(x) = 2x \text{ ó } x = 0$$

   y, por definición de f_4, para todo x vale que

$$f_4(x) = 2x \Leftrightarrow \Phi_x(x)\uparrow \text{ ó } \Phi_x(x) \neq x$$

   Por lo tanto, si $e = \#P'$ con $\Psi_P(x) = f_4(x)$ se tiene que para $x = e$

$$\Phi_e(e)\downarrow \Leftrightarrow \Phi_e(e)\uparrow \text{ ó } \Phi_e(e) \neq e \text{ ó } e = 0$$

   Ahora bien, sabemos que $e \neq 0$ porque $e = \#P'$ y P' no es el programa vacío. Y, además, sabemos que $\Phi_e(e) \neq e$ es falso, porque de terminar, P' devuelve siempre su entrada. De esta manera, llegamos al absurdo $\Phi_e(e)\downarrow \Leftrightarrow \Phi_e(e)\uparrow$

5. *Si se reduce a $h(x) = f_5(x,0)$, entonces* $h(x) = \begin{cases} 1 & \Phi_x(0)\uparrow \text{ ó } \Phi_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases}$ *y supongamos que $h(x)$ es computable con Q el programa que la computa. Sea el programa Q' con $e = \#Q'$ definido como:*

	Q
	$IF\ X = 0\ GOTO\ F$
$[L]$	$IF\ Y \neq 0\ GOTO\ L$
$[F]$	$Y \leftarrow 0$

   Luego, $\Psi_{Q'}(x) = \Phi_e(x) = \begin{cases} 0 & \text{si } x = 0 \text{ ó } h(x) = 0 \\ \uparrow & \text{en otro caso} \end{cases}$ *entonces* $\Phi_e(e)\downarrow \Leftrightarrow e = 0 \text{ ó } h(e) = 0 \underbrace{\Leftrightarrow}_{e \neq 0} h(e) = 0 \Leftrightarrow \Phi_e(0)\downarrow$ *y $\Phi_e(e)\uparrow$.*
   Por lo tanto, por la definición de $\Psi_{Q'}(x)$, se deduce que $\Phi_e(0) = 0$ (entonces $\Phi_e(0)\downarrow$).

$$\Phi_e(e)\uparrow \Leftrightarrow e \neq 0 \text{ y } h(e) = 1 \Leftrightarrow e \neq 0 \text{ y } (\Phi_e(0)\uparrow \text{ ó } \Phi_e(e)\downarrow) \underbrace{\Leftrightarrow}_{\Phi_e(0)\downarrow} \Phi_e(e)\downarrow$$

   lo que resulta una contradicción.

6. *Supongamos que f_6 es computable, entonces la función (predicado) $h(x) = f_6(x,x)$ también es computable.*
   Sea $g(x) = \begin{cases} x+1 & h(x)=1 \\ 0 & h(x)=0 \end{cases}$ *y e el código del programa que la computa.*
   Si $h(e)=1 \Rightarrow g(e)\downarrow$ y $\underbrace{g(e)}_{=e+1\neq 0} = 0$. *Lo que resulta en una contradicción.*

Si $h(e) = 0 \Rightarrow g(e) \uparrow$ *o bien* $\underbrace{g(e)}_{=0} \neq 0$. *Lo que resulta en una contradicción.*

Luego, h *no es computable y, por lo tanto,* f_6 *tampoco lo es.*

7. *Supongamos que* f_7 *es computable, entonces la función (predicado)* $h(x) = f_7(x, x)$ *también es computable.*

   Sea $g(x) = \begin{cases} x & h(x) = 1 \\ x+1 & h(x) = 0 \end{cases}$ *y* e *el código del programa que la computa.*
   Si $h(e) = 1 \Rightarrow g(e) \downarrow$ *y* $g(e) \neq e \neq 0$, *pero* $e = g(e)$. *Lo que resulta en una contradicción.*
   Si $h(e) = 0 \Rightarrow g(e) \uparrow$ *o bien* $g(e) = e$, *pero* $g(e) = e + 1 \neq e$. *Lo que resulta en una contradicción.*
   Luego, h *no es computable y, por lo tanto,* f_7 *tampoco lo es.*

8. *Supongamos que* f_8 *es computable, entonces si* $y = 0$

$$h(x, 0, z) = \begin{cases} \Phi_z^{(1)}(z) & \Phi_x^{(1)}(x) \downarrow \\ 0 & \text{en otro caso} \end{cases}, \text{ es computable}$$

   Si e *es el código del programa que computa a la función contante* 1, *entonces*

$$h(x, 0, e) = \begin{cases} 1 & \Phi_x^{(1)}(x) \downarrow \\ 0 & \text{en otro caso} \end{cases}, \text{ es computable}$$

   pero $h(x, 0, e) = Halt(x, x)$ *que no es computable.*

9. *Supongamos que* f_9 *es computable y sea* P *el programa que la computa.*
   Sea P' *definido como:*

	P
$[L]$	$IF\ Y \neq 0\ GOTO\ L$
	$Y \leftarrow 2015$

   Sea $e = \#P'$ *y, por lo tanto, hay dos posibilidades:*

$$\Psi_{P'}(e) \uparrow \text{ o bien } \Psi_{P'}(e) = 2015$$
$$\Phi_e(e) \uparrow \text{ o bien } \Phi_e(e) = 2015$$

   Si $\Phi_e(e) \uparrow$ *es porque* $\Psi_{P'}(e) = 0 \Rightarrow f_8(e) = 0 \Rightarrow \Phi_e(e) \downarrow$, *lo que resulta una contradicción.*

   En el otro caso, $\Phi_e(e) = 2015 \Rightarrow \Psi_{P'}(e) = 1 \Rightarrow f_8(e) = 1 \Rightarrow \Phi_e(e) \uparrow$ *ó* $\Phi_e(e) \leq 2014$ *pero ambas opciones son contradictorias con la premisa* $\Phi_e(e) = 2015$.

10. *Supongamos que* f_{10} *es computable, entonces la función (predicado)* $h(x) = f_{10}(x, x, x, x)$ *también es computable.*
    Luego, $h(x) = \begin{cases} 1 & si\ \Phi_x^{(1)}(x) \downarrow\ y\ \Phi_x^{(1)}(x) = x \\ 0 & en\ caso\ contrario \end{cases}$.

    Sea $g(x) = \begin{cases} x+1 & h(x) = 1 \\ x & h(x) = 0 \end{cases}$ *y* e *el código del programa que la computa.*
    Si $h(e) = 1 \Rightarrow g(e) \downarrow$ *y* $g(e) = e$ *es par, pero* $g(e) = e + 1$. *Lo que resulta en una contradicción.*
    Si $h(e) = 0 \Rightarrow g(e) \uparrow$ *o bien* $g(e) \neq e$, *pero* $g(e) = e$. *Lo que resulta en una contradicción.*
    Luego, h *no es computable y, por lo tanto,* f_{10} *tampoco lo es.*

Ejercicio 3.3.
Probar, reduciendo cualquier función del Ejercicio 2, que las siguientes funciones no son computables:

$$g_1(x, y) = \begin{cases} 1 & si\ \neg Halt(y, x) \\ 0 & en\ otro\ caso \end{cases} \qquad g_2(x, y, z, w) = \begin{cases} 1 & si\ \Phi_x^{(1)}(z) \downarrow\ y\ \Phi_y^{(1)}(w) \downarrow\ y\ \Phi_x^{(1)}(z) > \Phi_y^{(1)}(w) \\ 0 & en\ otro\ caso \end{cases}$$

$$g_3(x, y, z) = \begin{cases} z+1 & \Phi_x^{(1)}(y) \downarrow\ y\ \Phi_x^{(1)}(y) \neq z \\ 0 & en\ otro\ caso \end{cases} \qquad g_4(x, y, z) = \begin{cases} (\Phi_x^{(1)} \circ \Phi_y^{(1)})(z) & si\ \Phi_x^{(1)}(z) \downarrow\ y\ (\Phi_x^{(1)} \circ \Phi_y^{(1)})(z) \downarrow \\ 0 & en\ otro\ caso \end{cases}$$

$$g_5(x, y, z, w) = \begin{cases} x + w & \Phi_x^{(1)}(z) \uparrow\ y\ \Phi_y^{(1)}(z) \uparrow \\ w & \Phi_x^{(1)}(z) \downarrow,\ \Phi_y^{(1)}(z) \downarrow\ y\ \Phi_x^{(1)}(z) = \Phi_y^{(1)}(z) \\ w & en\ otro\ caso \end{cases}$$

$$g_6(x_1,y_1,x_2,y_2)=\begin{cases}1 & si\ \Phi^{(1)}_{x_1}(y_1)\ es\ divisible\ por\ \Phi^{(1)}_{x_2}(y_2)\\ 0 & en\ otro\ caso\end{cases}$$

$$g_7(x_1,y_1,x_2,y_2)=\begin{cases}y_1^2 \dot{-} 33 & si\ \Phi^{(1)}_{x_1}(y_1)\downarrow\ y\ \Phi^{(1)}_{x_2}(y_2)\downarrow\ y\ \left(\Phi^{(1)}_{x_1}(y_1)+2\right)\Big|\left(\Phi^{(1)}_{x_2}(y_2)+6\right)\\ 0 & en\ otro\ caso\end{cases}$$

Solución 3.3.

1. *Basta con notar que $g_1(x,y)=\alpha(f_1(x,y))$. Por lo tanto, como f_1 no es computable, g_1 tampoco puede serlo.*

2. *Supongamos g_2 computable y sea $\Phi^{(1)}_y(w)$ el intérprete de la función computable constantemente z, luego*

$$g_2(x,y,z,w)=\begin{cases}1 & \Phi^{(1)}_x(z)\downarrow\ y\ \Phi^{(1)}_x(z)>z\\ 0 & en\ caso\ contrario\end{cases}\Rightarrow g_2(x,y,y,w)=f_2(x,y,z)$$

$$\Rightarrow g_2(x,y,y,w)\ no\ es\ computable\ \Rightarrow g_2\ no\ es\ computable\ .$$

3. *Supogamos que g_3 es computable, entonces $g_3(x,y,0)$ también es computable. Luego*

$$g_3(x,y,0)=\begin{cases}1 & \Phi^{(1)}_x(y)\downarrow\ y\ \Phi^{(1)}_x(y)\neq 0\\ 0 & en\ caso\ contrario\end{cases}=\begin{cases}1 & \Phi^{(1)}_x(y)\downarrow\ y\ \Phi^{(1)}_x(y)>0\\ 0 & en\ caso\ contrario\end{cases}=f_2(x,y,0)$$

$$\Rightarrow g_3(x,y,0)=f_2(x,y,0)\ \Rightarrow g_3(x,y,0)\ no\ es\ computable\ \Rightarrow g_3\ no\ es\ computable\ .$$

4. *Supongamos que g_4 es computable y sea $\Phi^{(1)}_x$ el intérprete de la función computable identidad. Luego,*

$$g_4(\cdot,y,z)=\begin{cases}\Phi^{(1)}_y(z) & si\ \Phi^{(1)}_x(z)\downarrow\\ 0 & en\ caso\ contrario\end{cases}=\begin{cases}\Phi^{(1)}_y(z) & si\ \Phi^{(1)}_x(z)\downarrow\ y\ \Phi^{(1)}_x(z)\neq 0\\ 0 & si\ \Phi^{(1)}_x(z)\downarrow\ y\ \Phi^{(1)}_x(z)=0\\ 0 & en\ caso\ contrario\end{cases}=\begin{cases}\Phi^{(1)}_y(z) & si\ \Phi^{(1)}_x(z)\downarrow\ y\ \Phi^{(1)}_x(z)\neq 0\\ 0 & en\ caso\ contrario\end{cases}$$

$$\Rightarrow g_4(\cdot,y,z)=f_3(x,z,0)\ con\ f_3\ no\ conputable,\ entonces\ g_4\ no\ es\ computable.$$

5. *Supongamos que g_5 es computable, y sea $h(x)=g_5(x,d,x,2x)$ conde d es el número del programa*

$$Y\leftarrow X$$

De esta manera, $\Phi^{(1)}_d(x)=x$ para todo x. Supongamos que Q es un programa que computa g_5 y sea Q' el siguiente programa

$$\boxed{\begin{array}{l}X_2\leftarrow d\\ X_3\leftarrow X_1\\ X_4\leftarrow 2*X_1\\ Q\end{array}}$$

Recordando que X y X_1 son la misma variable, se ve que Q' computa $h(x)=f_4$ pero como f_4 no es computable h no puede ser computable y, por lo tanto, g_5 tampoco.

6. *Supongamos que g_6 es computable.*

   Notar que si se puede computar el hecho de que si resultado de un programa es divisible por el resultado de otro, en particular se puede computar si es divisible por 2. Es decir, que si no se puede saber si un programa va a producir un resultado par, tampoco se podría saber si el resultado de un programa será divisible por el resultado de otro.

   Sea e_{id} el número de algún programa que compute la función identidad (existe, pues es primitiva recursiva y por lo tanto, computable).

   Sea $h:\mathbb{N}^2\to\mathbb{N}$ la función definida como $h(x,y)=\alpha(g_5(x,y,e_{id},2))$. Como g_5 es computable y α es una función primitiva recursiva, h también debe ser computable. Por definición de h

$$h(x,y)=0\Leftrightarrow\alpha(g_5(x,y,e_{id},2))=0,\ \ es\ decir\ h(x,y)=1\Leftrightarrow g_5(x,y,e_{id},2)\neq 0$$

   entonces

$$h(x,y)=0\Leftrightarrow g_5(x,y,e_{id},2)=1\Leftrightarrow\Phi^{(1)}_x(y)\ es\ divisible\ por\ \Phi^{(1)}_{e_{id}}(2)\Leftrightarrow\Phi^{(1)}_x(y)\ es\ divisible\ por\ 2$$

   con lo cual se pudo sacar de encima los dos últimos argumentos de g_6.

   De acuerdo a la definición de f_3 se obtuvo que

$$h(x,y)=0\Leftrightarrow f_3(x,y)=0\ y\ h(x,y)=1\Leftrightarrow f_3(x,y)=1$$

   Por lo tanto, se deduce que $h=f_3$, y, entonces, debe ser f_3 computable.

7. *Supongamos que g_7 es computable, entonces si $\Phi_{x_1}^{(1)}$ el intérprete de la función computable constantemente nula. Además, como $2|6$, entonces $2|\left(\Phi_{x_2}^{(1)}(y_2)+6\right)$ si y s'olo si $\Phi_{x_2}^{(1)}(y_2)$ es par. Luego,*

$$g_7(e,y_1,x_2,y_2)=\begin{cases} y_1^2 \dot{-} 33 & si\ \Phi_{x_2}^{(1)}(y_2)\downarrow\ y\ \Phi_{x_2}^{(1)}(y_2)\ es\ par \\ 0 & en\ otro\ caso \end{cases} \Rightarrow$$

$$g_7(e,6,x_2,y_2)=\begin{cases} 3 & si\ \Phi_{x_2}^{(1)}(y_2)\downarrow\ y\ \Phi_{x_2}^{(1)}(y_2)\ es\ par \\ 0 & en\ otro\ caso \end{cases} = 3 * f_3(x_2,y_2)$$

Por lo tanto, si g_7 fuese computable entonces f_3 también lo sería. Lo que resulta una contradicción.

Ejercicio 3.4.
Probar que la siguiente función no es computable reduciendo la función f_7 del Ejercicio 2.

$$g_3'(x,y,z)=\begin{cases} z & \Phi_x^{(1)}(y)\downarrow\ y\ \Phi_x^{(1)}(y)\neq z \\ 0 & en\ otro\ caso \end{cases}$$

Sugerencia*: Revisar que la reducción maneje correctamente el caso $f_7(0)$.*

Solución 3.4.
Basta con notar que $\alpha\left(\alpha(g_3'(x,x,x))\right)=f_7(x)$. Por lo tanto si $g_3'(x,y,z)$ es computable, $f_7(x)$ es computable pero, por el Ejercicio 1 $f_7(x)$ no es computable.

Ejercicio 3.5.
Supongamos que $h_1,h_2:\mathbb{N}\to\mathbb{N}$ son funciones computables (totales). Supongamos que f y g son funciones que verifican la siguiente relación:

$$f(h_1(x),h_2(x))=g(x)$$

Decidir si las siguientes afirmaciones son verdaderas o falsas:

a. Si g no es computable, entonces f no es computable.

b. Si f no es computable, entonces g no es computable.

Solución 3.5.

a. La relación entre f y g implica que g se reduce a f. Por lo tanto, si g no es computable, entonces f tampoco lo es.

b. Esta es la afirmación recíproca de a, y es Falsa. Por ejemplo,

- *Sean $h_1(x)=107$ y $h_2(x)=42$ computables por ser funciones contantes. Si $f(x,y)=Halt(x,y)$ y g el resultado de componer f con h_1 y h_2*

  $$g(x)=Halt(107,42)$$

  Entonces cumplimos con las hipótesis de y, sin embargo, g es una función computable. Esto se sigue de que $Halt(107,42)$ es independiente de la variable x. Por lo tanto, la función g es constante, y en particular es computable.

Ejercicio 3.6.
Decimos que una función parcial computable f es extensible *si existe g computable tal que $g(x)=f(x)$ para todo $x\in dom\ f$. Probar que existe una función parcial computable que no es extensible.*
Sugerencia: considerar una función tal que con su extensión se podría computar alguna variante del halting problem.

Solución 3.6.
Supongamos que toda f parcial computable es extensible*. Por lo tanto, sea $f(x)=\Phi_x^{(1)}(x)+1$ parcial computable y $g:\mathbb{N}\to\mathbb{N}$ total computable que extiende a f.*
Sea $e\in\mathbb{N}$ tal que $\Phi_e^{(1)}=g$, luego como g es total

$$\Phi_e^{(1)}(e)=g(e)\downarrow \Leftrightarrow f(e)\downarrow\ \ y\ f(e)=\Phi_e^{(1)}(e)+1\Rightarrow \Phi_e^{(1)}(e)=\Phi_e^{(1)}(e)+1,$$

lo que resulta una contradicción.

- *Otra idea para resolverlo:*
  Sea $f(x)=\min_t STP(x,x,t)$ función parcial computable (notar que hay programas x con entrada x que no paran) y sea g computable extensible *de f.*

  Luego, para cada $x\in dom\ g$ se tiene que:

(1) Si $x \in dom\ f$, entonces $g(x)$, asumiendo que f está definido en ese x, será una cantidad de pasos de un programa (ie, el programa x con entrada x tardará lo que de $g(x)$).

(2) Si el programa x con entrada x no para en f, entonces ese x está en el dom g pero no está en el dom f.

Ejercicio 3.7.
Dadas dos funciones parciales $f, g : \mathbb{N} \to \mathbb{N}$, decimos que g domina fuertemente a f si $g(x) \geq f(x)$ para todo x tal que $x \in Domf$ y $x \in Domg$.

Sea $h : \mathbb{N} \to \mathbb{N}$ definida como

$$h(x) = \begin{cases} \min\limits_{t} STP^{(1)}(x,x,t) & \text{si Halt(x,x)} \\ \uparrow & \text{en otro caso} \end{cases}$$

Demostrar que si $g : \mathbb{N} \to \mathbb{N}$ es una función total que domina fuertemente a h, entonces g no es computable.

Solución 3.7.
Supongamos que g es computable y veamos que entonces podemos computar el predicado $Halt(x,x)$.

Algunas observaciones, sobre el significado de h y de g:

(i) $h(x)$ es la (mínima) cantidad de pasos que necesita el programa x con entrada x para terminar. Si el programa x con entrada x no termina, esta cantidad no está definida y tampoco h.

(ii) Como g domina fuertemente a h, entonces h está definida, $g(x) \geq h(x)$ y, por lo tanto, $g(x)$ es una cantidad de pasos para que el programa x con entrada x termine.

Por lo tanto, se distinguen dos casos

- *Si $h(x)\downarrow$, entonces $STP^{(1)}(x,x,g(x))$ por (ii).*
- *Si $h(x)\uparrow$, entonces $\neq STP^{(1)}(x,x,t)$ para cualquier t, en particular, $\neg STP^{(1)}(x,x,g(x))$ por (i).*

Es decir, $h(x)\downarrow \Leftrightarrow STP^{(1)}(x,x,g(x))$.

Sea, ahora, $h'(x) = STP^{(1)}(x,x,g(x))$ entonces $h'(x) = \begin{cases} 1 & h(x)\downarrow \\ 0 & \text{en otro caso} \end{cases} = \begin{cases} 1 & Halt(x,x) \\ 0 & \text{en otro caso} \end{cases}$.

Por lo tanto, $\underbrace{STP^{(1)}(x,x,g(x))}_{\substack{computable,\ pues \\ g\ lo\ es}} = h' = Halt(x,x)$ lo que resulta una contradicción. Luego, g no es computable.

Ejercicio 3.8.
Demostrar o refutar las siguientes afirmaciones.

a. Si $f : \mathbb{N}^n \to \mathbb{N}$ es una función total tal que, para alguna constante $k \in \mathbb{N}$, $f(x_1, \ldots, x_n) \leq k$ para todo $(x_1, \ldots, x_n) \in \mathbb{N}^n$, entonces f es computable.

b. La función $g(x) = \begin{cases} 1 & Halt(650,323) \\ 0 & \text{en otro caso} \end{cases}$ es computable.

c. La función $h(x) = \begin{cases} 1 & \text{si Dios existe} \\ 0 & \text{en otro caso} \end{cases}$ es computable.

d. La función dada por $t(x) = Halt(Halt(x,x))$ es primitiva recursiva.

Solución 3.8.

a. ***Falso***.
Por ejemplo,

$$f(x) = \begin{cases} 1 & Halt(x,x) \\ 0 & \text{en otro caso} \end{cases}$$

En este caso $f(x) \leq 1$ para todo x. Sin embargo f no es computable.

b. ***Verdadero***.
Al analizar la condición de g se observa que:

$$Halt(650,323) \Leftrightarrow \text{ El programa número } 323 \text{ con entrada } 650 \text{ se detiene.}$$

La sentencia que expresa $Halt(650,323)$ es o bien verdadera o bien falsa para todo x. En decir, a pesar de que no se sepa cuál es el valor exacto de $Halt(650,323)$, se sabe que es constante: 1 ó 0.

En el primer caso, g es computada por el programa

$$Y \leftarrow 1$$

En cambio, si es falsa, g es computada por el programa vacío.

En cualquier caso, g se trata de una función computable: la función constante 1 o la función constante 0.

c. ***Verdadero***.
O bien Dios existe o bien Dios no existe. Sea cual fuere, la función h será la constante 1 o la constante 0. En cualquiera de ambos casos, h será computable.

d. ***Verdadero***.
Si bien en principio no sabemos cuánto vale $Halt(x)$, éste no puede ser otra cosa que 0 o 1. Por lo tanto, para cualquier x, $t(x) = Halt(0)$, o bien $t(x) = Halt(1)$.

En cualquiera de los dos casos, $t(x) = 1$, con lo cual, t es una función constante, y por lo tanto, primitiva recursiva.

Ejercicio 3.9.
Dado una función total $f : \mathbb{N} \to \mathbb{N}$, un aproximador de f es una función total $g : \mathbb{N}^2 \to \mathbb{N}$ tal que para todo x, $g(x,t) = f(x)$ para todo t salvo finitos valores. Dicho de otra manera, $\lim_{t\to\infty} g(x,t) = f(x)$.

Decidir si son verdaderas o falsas las siguientes afirmaciones.

a. *Si f es computable entonces tiene un aproximador computable.*

b. *Si f tiene un aproximador computable entonces f es computable.*

Solución 3.9.

1. ***Verdadero***.
Si se sabe el verdadero valor de f, es fácil aproximarlo: basta con definir $g(x,t) = f(x)$.

Es fácil ver que "ignorar" el parámetro t es una reducción computable: si P es un programa que computa f ($\Psi_P^{(1)} = f$) entonces se deduce inmediatamente que el mismo P computa g como se definió anteriormete ($\Psi_P^{(2)} = g$).

2. ***Falso***.
La ventaja que tiene el aproximador g es que cuenta con un parámetro t con el que puede "acota" las computaciones y así asegurarse terminar siempre. Como toda computación que termina lo hace en una cantidad de pasos fija t, g se va a mantener constante cuando su segundo parámetro sea más grande que t.

Sea la función no computable $f(x) = \begin{cases} 1 & \Phi_x^{(1)}(x)\downarrow \\ 0 & \text{en otro caso} \end{cases}$ y sea $g(x,y) = \begin{cases} 1 & \Phi_x^{(1)}(x) \text{ termina en } t \text{ o menos pasos} \\ 0 & \text{en otro caso} \end{cases}$

Para completar el contraejemplo falta ver que

(i) $\lim_{t\to\infty} g(x,t) = f(x)$.

(ii) $g(x,y)$ *es computable.*

Lo primero se sigue inmediatamente de las definiciones. Si $\Phi_x^{(1)}(x)$ termina, entonces hay alguna cantidad de pasos t en la que termina y, por lo tanto,

$$g(x,t) = g(x,t+1) = g(x,t+2) = \ldots = f(x)$$

La segunda parte sale inmediatamente de ver que $g(x,t) = STP^{(1)}(x,x,t)$, con lo cual g es primitiva recursiva, y por lo tanto, computable.

Ejercicio 3.10.

a. *Sea $h : \mathbb{N} \to \mathbb{N}$ una función parcialmente computable.*

(i) Demostar que, para todo $u \in \mathbb{N}$ fijo, existe P_u tal que

$$\Phi_{P_u}^{(1)}(x) = h\left(\Phi_u^{(1)}(x)\right).$$

(ii) Demostrar que existe $f : \mathbb{N} \to \mathbb{N}$ primitiva recursiva tal que, para todo $u \in \mathbb{N}$,

$$\Phi_{f(u)}^{(1)}(x) = h\left(\Phi_u^{(1)}(x)\right).$$

b. Demostar que existe $f' : \mathbb{N}^2 \to \mathbb{N}$ primitiva recursiva tal que, para todo $u, v \in \mathbb{N}$,

$$\Phi^{(1)}_{f'(u,v)}(x) = \left(\Phi^{(1)}_v \circ \Phi^{(1)}_u\right)(x).$$

c. Sea $f : \mathbb{N}^2 \to \mathbb{N}$ parcial computable. Mostrar que existe $g : \mathbb{N} \to \mathbb{N}$ primitiva recursiva tal que

$$\Phi^{(1)}_{g(u,v)}(x) = f\left(\Phi^{(1)}_u(x), \Phi^{(1)}_v(x)\right).$$

Solución 3.10.

a. (i) Dado un programa fijo cuyo número es u, se quiere construir otro programa que le aplique la función h a cada una de sus salidas. Se sabe que tanto el intérprete de programas como h son funciones parcialmente computables, por lo tanto el siguiente programa cumple con lo pedido.

$$\boxed{\begin{array}{l} Y \leftarrow \Phi^{(1)}_u(X_1) \\ Y \leftarrow h(Y) \end{array}}$$

(ii) Sea P el programa

$$\boxed{\begin{array}{l} Y \leftarrow \Phi^{(1)}_{X_2}(X_1) \\ Y \leftarrow h(Y) \end{array}}$$

Donde, por el inciso anterior,

$$\Phi^{(2)}_P(x, u) = \Phi^{(1)}_{P_u}(x) = h\left(\Phi^{(1)}_u(x)\right)$$

tal que si $e = \#P$, se tiene que $\Psi^{(2)}_{S(u,e)}(x, u) = h\left(\Phi^{(1)}_u(x)\right)$.

Por el Teorema del parámetro, sabemos que existe una funciń primitiva recursiva $S : \mathbb{N}^2 \to \mathbb{N}$ tal que

$$\Phi^{(1)}_{S(u,e)}(x) = \Psi^{(2)}_e(x, u)$$

Por lo tanto, si $f(u) = S(u, e)$ (que claramente es primitiva recursiva, porque S lo es y e está fijo) se tiene que

$$\Phi^{(1)}_{f(u)}(x) = \Phi^{(1)}_{S(u,e)}(x) = \Psi^{(2)}_e(x, u) = h\left(\Phi^{(1)}_u(x)\right)$$

b. Si u y v están fijos, es sencillo construir un programa de una entrada que componga las funciones computadas por ambos (basta con correrlos uno detrás del otro). La idea será, ver a u y v como argumentos de un programa Q más general, de tres entradas, que podemos definir como sigue:

$$\boxed{\begin{array}{l} Y \leftarrow \Phi^{(1)}_{X_2}(X_1) \\ \\ Y \leftarrow \Phi^{(1)}_{X_3}(Y) \end{array}}$$

Luego,

$$\Phi^{(3)}_Q(x, u, v) = \Phi^{(1)}_v\left(\Phi^{(1)}_u(x)\right) = \left(\Phi^{(1)}_v \circ \Phi^{(1)}_u\right)(x)$$

Hay que fijar dos de los elementos de $\Phi^{(3)}_Q$ (u y v). Para ésto hay que notar que existe una función $S^2_1 : \mathbb{N}^3 \to \mathbb{N}$ tal que $\Phi^{(1)}_{S^2_1(u,v,d)}(x) = \Phi^{(3)}_d(x, u, v)$ donde $d = \#Q$.

Definiendo por último $f'(u, v) = S^2_1(u, v, d)$ se tiene que la función primitiva recursiva f' cumple con lo pedido ya que

$$\Phi^{(1)}_{f'(u,v)}(x) = \Phi^{(1)}_{S^2_1(u,v,d)}(x) = \Phi^{(3)}_d(x, u, v) = \left(\Phi^{(1)}_v \circ \Phi^{(1)}_u\right)(x)$$

c. Sea $h(u, v, x) = f(\Phi^{(1)}_u(x), \Phi^{(1)}_v(x))$. Vale que h es parcial computable pues es la composición de la función f que es parcial computable, y del intérprete universal que también lo es.

Dado que h es parcial computable, existe un programa que la computa. Sea e el número de dicho programa.

Por Teorema del Parámetro, vale que $\exists S$ una función primitiva recursiva tal que $\Phi^{(1)}_{S^2_1(u,v,d)}(x) = \Phi^{(1)}_e(u, v, x)$.

Sea, entonces, $g(a, b) = S(a, b, e)$ (que resulta primitiva recursiva puesto que S lo es).

Entonces,

$$\Phi^{(1)}_{g(a,b)}(x) = \Phi^{(1)}_{S(a,b,e)}(x) = \Phi^{(1)}_e(a, b, x) = h(a, b, x) = f\left(\Phi^{(1)}_u(x), \Phi^{(1)}_v(x)\right)$$

es decir, $\Phi^{(1)}_{g(a,b)}(x) = f\left(\Phi^{(1)}_u(x), \Phi^{(1)}_v(x)\right)$.

Ejercicio 3.11.
Demostrar, usando reducciones y/o el teorema del parámetro, que las siguientes funciones no son computables:

$$g_1(x) = \begin{cases} 1 & si\ Halt(1337, x) \\ 0 & en\ otro\ caso \end{cases} \qquad g_2(x,y,z) = \begin{cases} 1 & si\ \Phi_x^{(1)}(z)\downarrow\ y\ \Phi_y^{(1)}(z)\downarrow\ y\ \Phi_x^{(1)}(z) > \Phi_y^{(1)}(z) \\ 0 & en\ otro\ caso \end{cases}$$

$$g_3(x) = \begin{cases} 1 & si\ \Phi_x^{(1)} = f_0\ y\ f_0\ es\ total\ computable\ fija \\ 0 & en\ otro\ caso \end{cases} \qquad g_4(x,y) = \begin{cases} 1 & si\ \Phi_x^{(1)}(y)\downarrow\ y\ \Phi_y^{(1)}(x)\downarrow\ y\ \Phi_x^{(1)}(y) \neq \Phi_y^{(1)}(x) \\ 0 & en\ otro\ caso \end{cases}$$

$$g_5(x) = \begin{cases} 13 & si\ \Phi_x^{(1)}\ es\ la\ constante\ 7 \\ 0 & en\ otro\ caso \end{cases} \qquad g_6(x) = \begin{cases} 1 & si\ \Phi_x^{(1)}(42)\downarrow \\ 0 & en\ otro\ caso \end{cases}$$

$$g_7(x,z) = \begin{cases} 1 & si\ \Phi_x^{(1)}(z)\ computa\ la\ identidad \\ 0 & en\ otro\ caso \end{cases}$$

Solución 3.11.

1. *Supongamos que g_1 es computable y sea la función computable $m(y,x) = \begin{cases} 1 & \Phi_x^{(1)}(x)\downarrow \\ 0 & en\ otro\ caso \end{cases}$ con m_e el número del programa que computa a m.*

   Por el Teorema del Parámetro existe S p.r. tal que $\Phi^{(1)}_{S(x,m_e)}(y) = \Phi^{(1)}_{m_e}(y,x) = m(y,x)$.

   Como g_1 y S son computables, entonces su composición f también lo es. Luego,

   $$f = g_1(S(x,m_e)) = \begin{cases} 1 & Halt(1337, S(x,m_e)) \\ 0 & en\ otro\ caso \end{cases} = \begin{cases} 1 & \Phi^{(1)}_{S(x,m_e)}(1337)\downarrow \\ 0 & en\ otro\ caso \end{cases} = \begin{cases} 1 & \Phi^{(1)}_{m_e}(1337,x)\downarrow \\ 0 & en\ otro\ caso \end{cases}$$
   $$= \begin{cases} 1 & \Phi_x^{(1)}(x)\downarrow \\ 0 & en\ otro\ caso \end{cases} = Halt(x,x)$$

   lo que resulta ser una contradicción puesto que $Halt(x,x)$ no es computable.

2. *Supongamos que g_2 es computable y sea y_e el número del progrma que computa la función nula, entonces*

   $$g(x,z) = g_2(x, y_e, z) = \begin{cases} 1 & \Phi_x^{(1)}(z)\downarrow\ y\ \Phi_x^{(1)}(z) > 0 \\ 0 & en\ otro\ caso \end{cases} \quad es\ computable.$$

   Sea la función computable $m(z,x) = \begin{cases} 1 & \Phi_x^{(1)}(x)\downarrow \\ 0 & en\ otro\ caso \end{cases}$ con m_e el número del programa que computa a m.

   Por el Teorema del Parámetro existe S p.r. tal que $\Phi^{(1)}_{S(x,m_e)}(y) = \Phi^{(1)}_{m_e}(z,x) = m(z,x)$.

   Como g y S son computables, entonces su composición $g \circ S = f$ también lo es. Luego,

   $$f(z,x) = 1 \Leftrightarrow g(S(x,m_e)) \Leftrightarrow \Phi^{(1)}_{S(x,m_e)}\downarrow\ y\ \Phi^{(1)}_{S(x,m_e)}(z) > 0 \Leftrightarrow \Phi^{(1)}_{m_e}\downarrow\ y\ \Phi^{(1)}_{m_e}(z) > 0$$
   $$\Leftrightarrow m(z,x) = 1 \Leftrightarrow \Phi_x^{(1)}(x)\downarrow \Leftrightarrow Halt(x,x)$$

   lo que resulta ser una contradicción puesto que $Halt(x,x)$ no es computable.

3. *Supongamos g_3 computable. Se busca una función h total computable tal que $f(h(x)) = Halt(x,x)$.*

   Además, por Teorema del Parámetro, también podemos suponer que, de existir h, ésta provino de definir una función parcial computable g tal que $g(y,x) = \Phi^{(1)}_{h(x)}(y)$. Luego, sea

   $$g(y,x) = \begin{cases} f_0(y) & \Phi_x^{(1)}(x)\downarrow \\ \uparrow & en\ otro\ caso \end{cases}$$

   que resulta ser parcial computable puesto que

   $$\boxed{\begin{array}{l} Y \leftarrow \Phi_x^{(1)}(x) \\ Y \leftarrow f_0(y) \end{array}}$$

Y, además, $g(\cdot, x) = f_0 \Leftrightarrow Halt(x, x) = 1$.

Por el Teorema del Parámetro, como g *es parcial computable existe* h *p.r. (en particular, total computable) tal que* $g(y, x) = \Phi^{(1)}_{h(x)}(y)$. *Luego,*

$$g_3(h(x)) = 1 \Leftrightarrow \Phi^{(1)}_{h(x)}(y) \equiv f_0 \Leftrightarrow g(\cdot, x) = f_0 \Leftrightarrow Halt(x, x) = 1$$

Notar que como h *es total y* g_3 *es sólo devuelve los valores* 0 *y* 1*, entonces la composición* $(g_3 \circ h)(x)$ *es total y sólo devuelve los valores* 0 *y* 1*. Entonces* $(g_3 \circ h)(x) = Halt(x, x)$ *y, por lo tanto,* g_3 *no es computable.*

4. *Supongamos que* g_4 *es computable y sea* y_e *el número del progrma que computa la función nula, entonces*

$$g(x) = g_4(x, y_e) = \begin{cases} 1 & \Phi^{(1)}_x(y_e)\downarrow \text{ y } \Phi^{(1)}_x(y_e) \neq 0 \\ 0 & \text{en otro caso} \end{cases} \quad \text{es computable.}$$

Sea la función computable $m(y, x) = \begin{cases} 1 & \Phi^{(1)}_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases}$ *con* m_e *el número del programa que computa a* m.

Por el Teorema del Parámetro existe S *p.r. tal que* $\Phi^{(1)}_{S(x,m_e)}(y) = \Phi^{(1)}_{m_e}(y, x) = m(y, x)$.

Como g *y* S *son computables, entonces su composición* $g \circ S$ *también lo es. Luego,*

$$g(S(x, m_e)) = \begin{cases} 1 & \Phi^{(1)}_{S(x,m_e)}(y_e)\downarrow \text{ y } \Phi^{(1)}_{S(x,m_e)}(y_e) \neq 0 \\ 0 & \text{en otro caso} \end{cases} = \begin{cases} 1 & \Phi^{(1)}_{m_e}(y_e, x)\downarrow \text{ y } \Phi^{(1)}_{m_e}(y_e, x) \neq 0 \\ 0 & \text{en otro caso} \end{cases} = \begin{cases} 1 & m(y_e, x) \\ 0 & \text{en otro caso} \end{cases}$$

$$\Rightarrow g(S(x, m_e)) = \begin{cases} 1 & \Phi^{(1)}_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases} = Halt(x, x)$$

lo que resulta ser una contradicción puesto que $Halt(x, x)$ *no es computable.*

5. *Supongamos que* g_4 *es computable, entonces* $\lfloor \frac{g_5}{13} \rfloor = g_3$ *es computable, lo que resulta ser una contradicción.*

6. *Supongamos que* g_6 *es computable y sea la función parcial computable* $m(x) = \begin{cases} 1 & \Phi^{(1)}_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases}$

Sea e *el número del programa que computa* m*; es decir,* $\Phi^{(1)}_e(x) = m(x)$.

Por el Teorema del Parámetro existe S *función primitiva recursiva tal que* $\Phi^{(1)}_{S(x,e)}(\cdot) = \Phi^{(1)}_e(x)$.

Sea $f(x) = g_6(S(x, e))$ *computable total (pues es la composición de* g_6 *que supusimos computable total y de* S *que es primitiva recursiva). Luego,*

$$f(x) = \begin{cases} 1 & \Phi^{(1)}_{S(x,e)}(42)\downarrow \\ 0 & \text{en otro caso} \end{cases}$$

Ahora bien,

$$\Phi^{(1)}_{S(x,e)}(42)\downarrow \Leftrightarrow \Phi^{(1)}_{S(x,e)}()\downarrow \Leftrightarrow \Phi^{(1)}_e(x)\downarrow \Leftrightarrow m(x) = 1 \Leftrightarrow \Phi^{(1)}_x(x)\downarrow$$

Entonces se puede reescribir f *de la siguiente manera,*

$$f(x) = \begin{cases} 1 & \Phi^{(1)}_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases} \Rightarrow f(x) = Halt(x, x)$$

lo que resulta ser una contradicción. Por lo tanto, g_6 *no es computable.*

7. *Supongamos que* g_7 *es computable y sea la función parcial computable* $m(x, y) = \begin{cases} y & \Phi^{(1)}_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases}$

Sea e *el número del programa que computa* m*; es decir,* $\Phi^{(1)}_e(x, y) = m(x, y)$.

Por el Teorema del Parámetro existe S *función primitiva recursiva tal que* $\Phi^{(1)}_{S(x,e)}(y) = \Phi^{(1)}_e(x, y)$.

Sea $f(x) = g_7(S(x, e))$ *computable total (pues es la composición de* g_8 *que supusimos computable total y de* S *que es primitiva recursiva). Luego,*

$$f(x) = \begin{cases} 1 & \text{si } \Phi^{(1)}_{S(x,e)} \text{ computa la identidad} \\ 0 & \text{en otro caso} \end{cases}$$

Ahora bien,

$$\Phi^{(1)}_{S(x,e)} \text{ computa la identidad} \Leftrightarrow \Phi^{(1)}_{S(x,e)}(y) = y \ \forall y \Leftrightarrow \Phi^{(1)}_e(x,y) = y \ \forall y \Leftrightarrow m(x,y) = y \ \forall y \Leftrightarrow \Phi^{(1)}_x(x)\downarrow$$

Entonces se puede reescribir f de la siguiente manera,

$$f(x) = \begin{cases} 1 & \Phi^{(1)}_x(x)\downarrow \\ 0 & \text{en otro caso} \end{cases} \Rightarrow f(x) = Halt(x,x)$$

lo que resulta ser una contradicción. Por lo tanto, g_7 no es computable.

Ejercicio 3.12.
Demostrar que existe un programa P tal que $\Psi^{(1)}_P(x)\downarrow$ si y sólo si $x = \#P$. Es decir, probar que existe P tal que dom $\Phi^{(1)}_{\#P} = \{P\}$.

Solución 3.12.
Sea $g(x,y) = \begin{cases} 1 & x = y \\ \uparrow & x \neq y \end{cases}$, tal que dom $g = \{y\}$ y, entonces, por el Teorema del Parámetro, existe una S total computable tal que para todo $y \in \mathbb{N}$ vale que dom $\Phi^{(1)}_{S(y)} = \{y\}$.

Por el Teorema del Punto Fijo, como f es total computable existe P tal que dom $\Phi^{(1)}_{\#P}$ = dom $\Phi^{(1)}_{S(\#P)} = \{P\}$.

- *Otra manera de resolverlo:*
  Podemos suponer que $\#P$ viene de aplicar el Teorema de la Recursión a una f parcial computable de dos variables como $\Phi^{(1)}_{\#P} = f(\cdot, P)$.

  Por ejemplo, si $f(x,y) = \begin{cases} 1 & x = y \\ \uparrow & x \neq y \end{cases}$ se tiene que dom $f(\cdot, y) = \{y\}$ y por el Teorema de la Recursión existe P tal que $f(\cdot, P) = \Phi^{(1)}_{\#P}$. Entonces dom $\Phi^{(1)}_{\#P} = \{P\}$.

Ejercicio 3.13.
Demostrar, usando el teorema de la recursión, que las siguientes funciones no son computables:

$$h_1(x) = \begin{cases} 1 & si\ x \in Im\,\Phi^{(1)}_x \\ 0 & en\ otro\ caso \end{cases} \qquad h_2(x,y) = \begin{cases} 1 & si\ \Phi^{(1)}_x(y)\downarrow \ y\ \Phi^{(1)}_x(y) > x \\ 0 & en\ otro\ caso \end{cases}$$

$$h_3(x) = \begin{cases} 1 & si\ Im\,\Phi^{(1)}_x\ es\ infinita \\ 0 & en\ otro\ caso \end{cases} \qquad h_4(x) = \begin{cases} 1 & si\ |\,Dom\,\Phi^{(1)}_x| = x \\ 0 & en\ otro\ caso \end{cases}$$

$$h_5(x,y,z) = \begin{cases} 1 & si\ \Phi^{(1)}_x(y) = z \\ 0 & en\ otro\ caso \end{cases} \qquad h_6(x,y) = \begin{cases} 1 & si\ \Phi^{(1)}_x\ tiene\ algún\ punto\ fijo \\ 0 & en\ otro\ caso \end{cases}$$

$$h_7(x,y) = \begin{cases} 1 & si\ x \in Dom\,\Phi^{(1)}_y \cap Im\,\Phi^{(1)}_y \\ 0 & en\ otro\ caso \end{cases}$$

Solución 3.13.

1. *Supongamos h_1 computable y sea la función f parcialmente computable definida como*

$$f(x,u) = \begin{cases} \uparrow & h_1(u) = 1 \\ x & en\ otro\ caso \end{cases}$$

Por el Teorema de la Recursión, existe $e \in \mathbb{N}$ tal que $\Phi^{(1)}_e(x) = f(x,e)$. Luego, para cualquier $x \in \mathbb{N}$ se tiene que

- *Si $h_1(e) = 1$, entonces $e \in Im\,\Phi^{(1)}_e$ pero*

$$h(e) = 1 \Rightarrow f(x,e)\uparrow \ \forall x \Rightarrow e \notin Im\,\Phi^{(1)}_e$$

- *Si $h_1(e) = 0$, entonces $x \notin Im\,\Phi^{(1)}_e$ pero*

$$h_1(e) = 0 \Rightarrow f(x,e) = x\ \forall x,\ \text{en particular para } x = e \Rightarrow e \in Im\,\Phi^{(1)}_e$$

En ambos casos se obtuvo una contradicción que demuestra la falsedad de nuestra suposición inicial. Es decir, h_1 no es computable.

2. *Supongamos h_2 computable y sea la función f parcialmente computable definida como*

$$f(x,y) = \begin{cases} x & h_1(x,y) = 1 \\ x+1 & \text{en otro caso} \end{cases}$$

Por el Teorema de la Recursión, existe $e \in \mathbb{N}$ tal que $\Phi_e^{(1)}(x) = f(x,e)$. Luego, para cualquier $x \in \mathbb{N}$ se tiene que

- *Si $h_1(e,e) = 1$, entonces $\Phi_e^{(1)}(e) \downarrow$ y $\Phi_e^{(1)}(e) > e$ pero*

$$h(e,e) = 1 \Rightarrow e < \Phi_e^{(1)}(e) = f(e,e) = e$$

- *Si $h_1(e) = 0$, entonces $\Phi_e^{(1)}(e) \uparrow$ ó $\Phi_e^{(1)}(e) \leq e$ pero*

$$h_1(e) = 0 \Rightarrow e \geq \Phi_e^{(1)}(e) = f(e,e) = e+1$$

En ambos casos se obtuvo una contradicción que demuestra la falsedad de nuestra suposición inicial. Es decir, h_2 no es computable.

3. *Supongamos h_3 computable y sea la función f parcialmente computable definida como*

$$f(x,u) = \begin{cases} \uparrow & h_1(u) = 1 \\ x & \text{en otro caso} \end{cases}$$

Por el Teorema de la Recursión, existe $e \in \mathbb{N}$ tal que $\Phi_e^{(1)}(x) = f(x,e)$. Luego, para cualquier $x \in \mathbb{N}$ se tiene que

- *Si $h_1(e) = 1$, entonces $Im\,\Phi_e^{(1)}$ es inifita pero*

$$h(e) = 1 \Rightarrow f(x,e) \uparrow \ \forall x \Rightarrow Im\, f(\cdot,e) = \emptyset \Rightarrow Im\,\Phi_e^{(1)} = \emptyset$$

- *Si $h_1(e) = 0$, entonces $Im\,\Phi_e^{(1)}$ finita pero*

$$h_1(e) = 0 \Rightarrow f(x,e) = x\ \forall x \Rightarrow Im\, f(\cdot,e) \text{ es infinita} \Rightarrow Im\,\Phi_e^{(1)} \text{ es infinita.}$$

En ambos casos se obtuvo una contradicción que demuestra la falsedad de nuestra suposición inicial. Es decir, h_3 no es computable.

4. *Supongamos h_4 computable y sea la función f parcialmente computable definida como*

$$f(x,u) = \begin{cases} x & h_1(u) = 1 \\ \uparrow & \text{en otro caso} \end{cases}$$

Por el Teorema de la Recursión, existe $e \in \mathbb{N}$ tal que $\Phi_e^{(1)}(x) = f(x,e)$. Luego, para cualquier $x \in \mathbb{N}$ se tiene que

- *Si $h_1(e) = 1$, entonces $|\,Dom\,\Phi_e^{(1)}| = e$ pero*

$$h(e) = 1 \Rightarrow f(x,e) = x\ \forall x \Rightarrow Dom\, f(\cdot,e) = \mathbb{N} \Rightarrow Dom\,\Phi_e^{(1)} = \mathbb{N}$$

- *Si $h_1(e) = 0$, entonces $|\,Dom\,\Phi_e^{(1)}| \neq e$ pero*

$$h_1(e) = 0 \Rightarrow f(x,e) \uparrow \Rightarrow Dom\,\Phi_e^{(1)} = \emptyset.$$

En ambos casos se obtuvo una contradicción que demuestra la falsedad de nuestra suposición inicial. Es decir, h_4 no es computable.

5. *Supongamos que h_5 es computable. Definimos la siguiente función*

$$m(a,b) = \begin{cases} a+1 & \text{si } h_5(a,a,a) = 1 \\ a & \text{en otro caso} \end{cases}$$

computable total, pues h_5 es computable total. Entonces por Teorema de la Recursión podemos afirmar que existe e tal que $\Phi_e^{(1)}(a,b) = m(a,b)$. Notar que e es un número determinado, fijo.

$$h_5(e,e,e) = 1 \Leftrightarrow \Phi_e^{(1)}(e) = e \Leftrightarrow m(e,e) = e \Leftrightarrow h_5(e,e,e) = 0$$

lo que resulta ser una contradicción.

6. *Supongamos que h_6 es computable. Definimos la siguiente función*

$$m(y,x) = \begin{cases} y & si\ \Phi_x^{(1)}(x)\downarrow \\ \uparrow & en\ otro\ caso \end{cases}$$

parcial computable y sea e el número de programa que computa m; es decir, $\Phi_e^{(1)}(y,x) = m(y,x)$.

Por el Teorema de la Recursión existe S p.r. tal que $\Phi_{S(x,e)}^{(1)}(y) = \Phi_e^{(1)}(y,x)$.

Sea $g_e(x) = \begin{cases} 1 & si\ \forall y,\ \Phi_{S(x,e)}^{(1)}(y) = y \\ 0 & en\ otro\ caso \end{cases}$, *entonces*

$$g_e(x) = \begin{cases} 1 & si\ \forall y,\ \Phi_e^{(1)}(y,x) = y \\ 0 & en\ otro\ caso \end{cases} = \begin{cases} 1 & si\ \forall y,\ m(y,x) = y \\ 0 & en\ otro\ caso \end{cases} = \begin{cases} 1 & si\ \Phi_x^{(1)}(x)\downarrow \\ 0 & en\ otro\ caso \end{cases} = Halt(x,x)$$

Por lo tanto, h_6 no puede ser computable.

7. *Supongamos h_7 computable y sea la función f parcialmente computable definida como*

$$f(x,u) = \begin{cases} \uparrow & h_7(x,u) = 1 \\ x & en\ otro\ caso \end{cases}$$

Por el Teorema de la Recursión, existe $e \in \mathbb{N}$ tal que $\Phi_e^{(1)}(x) = f(x,e)$. Luego, para cualquier $x \in \mathbb{N}$ se tiene que

- *Si $h_7(x,e) = 1$, entonces $x \in Dom\,\Phi_e^{(1)} \cap Im\,\Phi_e^{(1)}$ pero*

$$h_7(x,e) = 1 \Rightarrow f(x,e)\uparrow \Rightarrow \Phi_e^{(1)}(x)\uparrow \Rightarrow x \notin Dom\,\Phi_e^{(1)} \Rightarrow x \notin Dom\,\Phi_e^{(1)} \cap Im\,\Phi_e^{(1)}$$

- *Si $h_7(x,e) = 0$, entonces $x \notin Dom\,\Phi_e^{(1)} \cap Im\,\Phi_e^{(1)}$ pero*

$$h_7(x,e) = 0 \Rightarrow f(x,e) = x \Rightarrow \Phi_e^{(1)}(x) = x \Rightarrow x \in Dom\,\Phi_e^{(1)} \cap Im\,\Phi_e^{(1)}$$

En ambos casos se obtuvo una contradicción que demuestra la falsedad de nuestra suposición inicial. Es decir, h_7 no es computable.

Ejercicio 3.14.
Llamamos W_e al dominio del programa con número e. Es decir, $W_e = Dom\,\Phi_e^{(1)} = \{x | \Phi_e^{(1)}(x)\downarrow\}$.

1. *Probar que existe $m \in \mathbb{N}$ tal que $W_m = m$.*
2. *Probar que existe $m \in \mathbb{N}$ tal que $W_m = \{m, 2m, 3m, \ldots\}$.*

Solución 3.14.

1. *Sea la siguiente función parcial computable* $g(x,y) = \begin{cases} 1 & y = x \\ \uparrow & caso\ contrario \end{cases}$.

Por el Teorema de la Recursión, existe $e \in \mathbb{N}$ tal que $\Phi_e^{(1)}(y) = g(e,y)$. Por lo tanto,

$$\Phi_e^{(1)}(y) = g(e,y) = \begin{cases} 1 & y = e \\ \uparrow & caso\ contrario \end{cases}$$

Es decir, $\Phi_e^{(1)}(y)\downarrow \Leftrightarrow y = e$; ie, el programa e termina sólo con entrada e y, por lo tanto, $Dom\,\Phi_e^{(1)} = \{e\}$.

2. *Sea la siguiente función parcial computable* $f(x,y) = \begin{cases} 1 & y|x \\ \uparrow & caso\ contrario \end{cases}$.

Por el Teorema de la Recursión, existe $e \in \mathbb{N}$ tal que $\Phi_e^{(1)}(x) = f(x,e)$. Por lo tanto,

$$\Phi_e^{(1)}(x) = f(x,e) = \begin{cases} 1 & y|e \\ \uparrow & caso\ contrario \end{cases}$$

Es decir, por el Teorema de la Recursión, existe $e \in \mathbb{N}$ tal que $\Phi_e^{(1)}(x) = f(x,e)$ para todo x, entonces $Dom\,\Phi_e^{(1)}(x) = \{e, 2e, 3e, \ldots\}$.

Ejercicio 3.15.
Sean $C_1, \ldots, C_k$ conjuntos de índices de programas y sea $C = C_1 \cap \ldots \cap C_k$.

a. Demostrar que C es un conjunto de índices de programas (i.e., $C = \{x : \Phi_x \in \mathcal{C}\}$, con C una clase de funciones).

b. Proponer un conjunto, que no sea un conjunto de índices de programas y no sea computable.

Solución 3.15.

a. $x \in C \Leftrightarrow x \in C_i \ \forall 1 \leq i \leq k \Leftrightarrow \Phi_x \in \mathcal{C}_i \ \forall 1 \leq i \leq k \Leftrightarrow \Phi_x \in \bigcap_{i=1}^{k} \mathcal{C}_i = \mathcal{C}$.

Luego, $C = \left\{ x : \Phi_x \in \bigcap_{i=1}^{k} \mathcal{C}_i = \mathcal{C} \right\}$ *y, por lo tanto, es un conjunto de índices de programas.*

b. *Supongamos que el conjunto* $A = \{x \in \mathbb{N} : \Phi_x^{(1)} \equiv x\}$ *sea un conjunto de índices y sea un* $x_0 \in A$*, luego existe* $x_1 \neq x_0$ *tal que* $\Phi_{x_1}^{(1)} \equiv \Phi_{x_0}^{(1)} \equiv x_0$ *y, por lo tanto,* $\Phi_{x_1}^{(1)} \in A \Rightarrow \Phi_{x_1}^{(1)} = x_1$ *pero* $x_0 = \Phi_{x_0}^{(1)} = \Phi_{x_1}^{(1)} = x_1$*. Lo que resulta una contradicción. Luego, A no es un conjunto de índices.*

Por otra parte, Supongamos que A es computable y sea la función parcial computable $f(x,y) = \begin{cases} x+1 & \Phi_y^{(1)} \equiv y \\ y & \Phi_y^{(1)} \neq y \end{cases}$

Por el Teorema de la Recursión, existe $e \in \mathbb{N}$ *tal que* $\Phi_e^{(1)}(\cdot) = f(\cdot, e)$.

- *si* $\Phi_e^{(1)}(\cdot) = e \Rightarrow f(x,e) = e$ *para todo x, lo cual es una contradicción porque f no es una función constante.*
- *si* $\Phi_e^{(1)}(\cdot) \neq e \Rightarrow f(x,e) \neq e$ *para todo x pero* $f(x,e) = e$ *para todo x , lo que resulta ser una contradicción.*

Por lo tanto, el conjunto $\{x \in \mathbb{N} : \Phi_x^{(1)} \equiv x\}$ *no es un conjunto de índices de programas y no es computable.*

Ejercicio 3.16.
Probar que son equivalentes

I. *D es un conjunto de índices de programas ($D = \{\#P : \Psi_P \in \mathcal{C}\}$, con $\mathcal{C}$ una clase de funciones).*

II. *Para todo par de programas P y Q, si $\#P \in D$ y $\Psi_P = \Psi_Q$ entonces $\#Q \in D$.*

Solución 3.16.

- *$(I) \Rightarrow (II)$: trivial.*
- *$(II) \Rightarrow (I)$: Se quiere ver que D es el conjunto de índices de una clase Γ.*
  - *Si $D = \emptyset$, entonces $D = \{x : \Phi_x \in \{\emptyset\}\}$ (es decir, $\Gamma = \{\emptyset\}$).*
  - *Si $D \neq \emptyset$, basta con definir $D = \{x : \Phi_x \in \Gamma\}$ con $\Gamma = \{\Phi_{\#P} : \#P \in D\}$.*

Ejercicio 3.17.
Demostrar, usando reducciones y el teorema de Rice, que las siguientes funciones no son computables:

$$g_1(x) = \begin{cases} 1 & \text{si } Dom\,\Phi_x^{(1)} = \mathbb{N} \\ 0 & \text{en otro caso} \end{cases} \qquad g_2(x,y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \text{ está definida y es par} \\ 0 & \text{en otro caso} \end{cases}$$

$$g_3(x,y) = \begin{cases} 1 & \text{si } Dom\,\Phi_x^{(1)}(y) = 0 \text{ y } \Phi_y^{(1)}(0) = 0 \\ 0 & \text{en otro caso} \end{cases} \qquad g_4(x,y) = \begin{cases} 1 & \text{si } \Phi_x^{(1)} \text{ y } \Phi_y^{(1)} \text{ computan la misma función} \\ 0 & \text{en otro caso} \end{cases}$$

$$g_5(x) = \begin{cases} 1 & \text{si } Dom\,\Phi_x^{(1)} = \emptyset \\ 0 & \text{en otro caso} \end{cases} \qquad g_6(x,y) = \begin{cases} 1 & \text{si } Dom\,\Phi_x^{(1)} \cup Dom\,\Phi_y^{(1)} = \mathbb{N} \\ 0 & \text{en otro caso} \end{cases}$$

$$g_7(\langle x,y \rangle) = \begin{cases} 1 & \text{si } y \in Dom\,\Phi_x^{(1)} \\ 0 & \text{en otro caso} \end{cases} \qquad g_8(\langle x,y \rangle) = \begin{cases} \Phi_x^{(1)}(\Phi_y^{(1)}(72)) & \text{si } \Phi_x^{(1)} \circ \Phi_y^{(1)} \text{ es total} \\ 73 & \text{en otro caso} \end{cases}$$

Observación*: Notar que la composición de dos funciones parciales puede ser una función parcial.*

Solución 3.17.

1. Sea $G = \{a | g_1(a) = 1\} = \{a | \, Dom\,\Phi_a = \mathbb{N}\}$*, hay que ver que G es un conjunto de índices no trivial*

- *G es un conjunto de índices:* *Sea $p \in G$ tal que $\Phi_p^{(1)} = \Phi_q^{(1)}$ y supongamos entonces que $q \notin G$. Es decir que existe $y_0 \in \mathbb{N}$ tal que $\Phi_q^{(1)}(y_0) \uparrow$.*

  Entonces, como sucede que $\Phi_p^{(1)} = \Phi_q^{(1)}$, en particular $\Phi_p^{(1)}(y_0) = \Phi_q^{(1)}(y_0)$. Es decir que $\Phi_p^{(1)}(y_0) \uparrow$. Luego $\Phi_p^{(1)}$ no es total y, por lo tanto, $p \notin G$. Lo que resulta ser una contradicción. Luego debe ser $q \in G$.
- *G es no trivial:*
  - *Sea e_0 el número de un programa que se indefine para la entrada 42. Es claro que $Dom\,\Phi_{e_0}^{(1)} \neq \mathbb{N}$ (pues $42 \notin Dom\,\Phi_{e_0}^{(1)}$). Luego $e_0 \notin G$. Entonces $G \neq \mathbb{N}$.*
  - *Sea e_1 el número de un programa que computa la funcón constante 73. Luego $Dom\,\Phi_{e_1}^{(1)} = \mathbb{N}$. Entonces $e_0 \notin G$. Por lo tanto $G \neq \emptyset$.*

2. *Se puede reducir a a una función más simple que, además, es la característica de un conjunto de núumeros naturales fijando una variable. Conviene fijar y (lo que quedaría tendrá más pinta de conjunto de índices ya que x aparece en el núumero de programa). Entonces g_2 fuese computable, también lo sería*

$$g(x) = g_2(x,0) = \begin{cases} 1 & \text{si } \Phi_x^{(1)}(y) \text{ está definida y es par} \\ 0 & \text{caso contrario} \end{cases}$$

Ahora, g es la característica del conjunto

$$A = \{x : \Phi_x^{(1)}(0) \text{ está definida y es par}\}.$$

que es un conjutno de índices, el correspondiente a la clase de funciones

$$\mathcal{C}_A = \{g : g(0) \text{ está definida y es par}\}$$

Además, A es un conjunto no trivial: el número de cualquier programa que compute la constante 0 está en A, por lo que $A \neq \emptyset$, pero el número de cualquier programa que se indefina siempre no está en A, por lo que $A \neq \mathbb{N}$. Luego, por el teorema de Rice, g no es computable. Llegamos a un absurdo que provino de suponer que g_2 era computable.

3. *Fijando el parámetro y en algún valor e apropiado, entonces*

$$\Phi_x^{(1)}(e) = 0 \text{ ó } \Phi_e^{(1)}(0) = 0$$

Se quiere que la segunda parte de la disyunción sea falsa para cualquier valor de x. Esto sucede si tomamos $e = \#P$, donde P es un programa que computa la función constante 1. Para este e y para cualquier x,

$$\Phi_x^{(1)}(e) = 0 \text{ ó } \Phi_e^{(1)}(0) = 0 \Leftrightarrow \Phi_x^{(1)}(e) = 0$$

Se puede argumentar, entonces, que si g_3 fuera computable, también lo sería

$$f(x) = g_3(x,e) = \begin{cases} 1 & \Phi_x^{(1)}(e) = 0 \\ 0 & \text{en otro caso} \end{cases}$$

donde f es la función característica del conjunto

$$A = \{x : \Phi_x^{(1)}(e) = 0\}$$

que es un conjunto de índices, el correspondiente a la clase de funciones

$$\mathcal{C}_A = \{g : g(e) = 0\}$$

(con e valor fijo).

Además, A es no trivial: el número de cualquier programa que compute la constante 0 está en A, por lo que $A \neq \emptyset$. Mientras que el número de cualquier programa que compute la constante 678 no está en A, por lo que $A \neq \mathbb{N}$.

Aplicando el teorema de Rice, concluimos que f no es computable, por lo que tampoco puede ser computable g_4.

4. *Fijado y en el número de algún programa, digamos e, y sea P el programa tal que $\#P = e$. Luego, se define*

$$f(x) = g_4(x,e) = \begin{cases} 1 & \text{si } \Phi_x^{(1)} \text{ computa la función } \Phi_P(1) \\ 0 & \text{en otro caso} \end{cases}$$

donde f es la función característica del conjunto de índices

$$A = \{x : \Phi_x^{(1)} \text{ computa la función } \Phi_P\} = \{x : \Phi_x^{(1)} \in \mathcal{P}\}, \text{ con } \mathcal{P} = \{\Phi_P^{(1)}\}$$

de donde resulta que el conjunto de índices no es trivial.

Luego, por el Teorema de Rice, este conjunto no es computable, o lo que es lo mismo, f no es computable.

Notar que este argumento vale para cualquier e, y en particular, si se fija en el número de algún programa que computa la identidad, se tendría que f es la función que decide si un programa con número x computa la función identidad (se sabe que no es computable).

5. *Sea $G = \{a | \, Dom\, \Phi_a = \emptyset\}$, hay que ver que G es un conjunto de índices no trivial*

   - *G es un conjunto de índices: Sea $p \in G$ tal que $\Phi_p^{(1)} = \Phi_q^{(1)}$ y supongamos entonces que $q \notin G$. Es decir que existe $y_0 \in \mathbb{N}$ tal que $\Phi_q^{(1)}(y_0)\downarrow$.*

     Entonces, como sucede que $\Phi_p^{(1)} = \Phi_q^{(1)}$, en particular $\Phi_p^{(1)}(y_0) = \Phi_q^{(1)}(y_0)$. Es decir que $\Phi_p^{(1)}(y_0)\downarrow$. Luego $p \notin G$. Lo que resulta ser una contradicción. Luego debe ser $q \in G$.
   - *G es no trivial:*
     - *Sea e_0 el número de un programa que se indefine para toda entrada. Es claro que $Dom\, \Phi_{e_0}^{(1)} = \emptyset$. Luego $e_0 \in G$. Entonces $G \neq \emptyset$.*
     - *Sea e_1 el número de un programa que computa la funcón constante 73. Luego $Dom\, \Phi_{e_1}^{(1)} = \mathbb{N}$. Entonces $e_0 \notin G$. Por lo tanto $G \neq \mathbb{N}$.*

Ejercicio 3.18.
Decidir si los siguientes conjuntos son o no computables y justificar la respuesta:

a. *$A = \{x : \Phi_x^{(1)}(5) = f(10)\}$, donde f es una función computable (total) fija.*

b. *$B = \{x : \Phi_x^{(1)}(y) = 0$ para infinitos $y \in \mathbb{N}\}$.*

c. *$C = \{x :$ el programa con número x tiene la instrucción $Y \leftarrow Y + 1$ en algún lugar de su código$\}$.*

d. *$D = \{x : \Phi_x^{(1)}(y) = Halt(y, y)\}$.*

e. *$E = \{x : \Phi_x^{(1)}(y)$ es par siempre que y es par$\}$.*

f. *$F = \{x :$ el programa número x termina si y sólo si recibe como entrada un número par$\}$.*

g. *$G = \{x :$ el programa número x tiene a lo sumo 5 instrucciones$\}$.*

h. *$H = \{x : \Phi_x^{(1)}(y) = x$ para todo $y\}$.*

Solución 3.18.

a. *Observar que $f(10)$ es un número fijo, entonces A es un conjunto de índices ya que es el conjunto de los números de programas que computan funciones en la clase de las funciones que en 5 devuelven ese número ($f(10)$).*

   Además, $A \neq \emptyset$ pues se puede hacer un programa que en 5 devuelva $f(10)$ (por ejemplo el que devuelve siempre $f(10)$) y $A \neq \mathbb{N}$ pues se puede hacer un programa que no devuelva $f(10)$ (por ejemplo el que devuelve siempre $f(10) + 1$). Luego, por el teorema de Rice, A no es computable.

b. *B es un conjunto de índices ya que es el conjunto de los números de programas que computan funciones en la clase de las funciones que devuelven 0 en infinitos valores.*

   Además, $B \neq \emptyset$ pues se puede hacer un programa que devuelva 0 en infinitos valores (por ejemplo el que devuelve siempre 0) y $B \neq \mathbb{N}$ pues podemos hacer un programa que no devuelva 0 en infinitos valores (por ejemplo el que devuelve siempre 1). Luego, por el teorema de Rice, B no es computable.

c. *El conjunto C es computable, es más, es p.r. pues se puede ver que su función caracterśtica es p.r. escribiéndola usando funciones que ya vimos que son p.r.*

   En este caso C no es un conjunto de índices. Se puede ver que la propiedad que caracteriza a C se refiere a programas y no a funciones. A modo de contraejemplo, estos dos programas:

$$P_c : \begin{matrix} Y \leftarrow Y + 1 \\ Y \leftarrow Y \dot{-} 1 \end{matrix} \qquad y \qquad Q_c : [\text{el programa vacío}]$$

   computan la misma función, la constante 0. Sin embargo, $\#P_c \in C$, mientras que $\#Q_c \notin C$.

d. *D es un conjunto de índices, el correspondiente a la clase de funciones*

$$\mathcal{C}_D = \{g : g(x) = Halt(x, x)\}$$

   Sin embargo, se sabe que no es posible construir un programa que compute una función semejante. Esto quiere decir que $D = \emptyset$ y, por lo tanto, D es computable (más aún, su función característica es la función constante 0).

e. No, es un conjunto de índices no trivial: son los índices de las funciones que en los números pares devuelven un resultado par, y, además, no es trivial (por ejemplo, la función identidad pertenece a E, con lo cual no es $\emptyset$, y la función constante 17 no pertenece a E, con lo cual no es $\mathbb{N}$).

f. No, es un conjunto de índices no trivial: son los índices de las funciones parciales computables cuyo dominio son exactamente los números pares.

g. El conjunto G es p.r. $G(x) = \begin{cases} 1 & si\ |x+1| \leq 5 \\ 0 & caso\ contrario \end{cases}$.

G no es un conjunto de índices porque dada una función hay programas que la computan con una cantidad de instrucciones tan grande como se quiera, y, por lo tanto, dada cualquier clase de funciones, no todos los números de los programas que computen alguna de sus funciones estarán en el conjunto.

h. H no es un conjunto de índices porque la propiedad expresada depende del programa que estemos considerando para computar una función dada. Notar que, para cualquier función parcial computable, existen infinitos programas que la computan. Si $e_1 \neq e_2$ tales que $\Phi_{e_1}^{(1)} = \Phi_{e_2}^{(1)}$ con $e_1 \in H$, entonces claramente $e_2 \notin H$.

Ejercicio 3.19.
Sean A y B dos subconjuntos de $\mathbb{N}$. Decimos que A es reducible a B, denotado $A \leq B$, si existe una función (total) computable $f : \mathbb{N} \to \mathbb{N}$ tal que para todo x se tiene que $x \in A$ si y sólo si $f(x) \in B$.
Luego, si $A \leq B$ entonces

a. B computable, entonces A computable.

b. B c.e., entonces A es c.e.

c. B co-c.e., entonces A co-c.e.

Solución 3.19.

a.

b. Como B es c.e. existe g parcial computable tal que $B = Dom\, g$. Sea f una reducción de A en B. Se tiene que $g \circ f$ es parcial computable.

Para todo $x \in \mathbb{N}$ tal que $x \in A$ si y sólo si $f(x) \in B$ si y sólo si $g(f(x)) \downarrow$ si y sólo si $x \in Dom\, g \circ f$.

Entonces $A = Dom\, g \circ f$ y, por lo tanto, A es c.e.

c.

Ejercicio 3.20.
Demostrar o refutar cada una de las siguientes afirmaciones.

1. *Todo conjunto finito B es c.e.*
2. *Todo conjunto B cofinito es computable (un conjunto es cofinito si tiene complemento finito).*
3. *Si $f : \mathbb{N} \to \mathbb{N}$ y tanto su dominio como su imagen son computables, entonces f es computable.*
4. *Si $B \subset \mathbb{N}$ tal que para todo $S \subset B$ finito, S es computable, entonces B es computable.*
5. *Si $B \subset \mathbb{N}$ no es computable, entonces B es un conjunto de índices.*
6. *Si B es c.e. y f una función total computable, entonces $f(B) = \{f(x) : x \in B\}$ es c.e..*
7. *Si B es computable y f una función total computable, entonces $f(B)$ es computable.*
8. *Si B es p.r y f una función total computable, entonces $f(B)$ es computable.*
9. *Si B es p.r. y f p.r., entonces $f(B)$ es c.e.*
10. *Si B es co-c.e. y f una función parcial computable, entonces $f(B)$ es co-c.e.*
11. *Si B es c.e. y A computable, entonces $B \setminus A$ es c.e.*
12. *Si B es c.e. y A computable, entonces $A \setminus B$ es c.e.*
13. *Si B es computable entonces es c.e.*
14. *Si B es c.e. entonces B es computable o su complemento lo es.*

15. *Si B es c.e. entonces su complemento es c.e.*

Solución 3.20.

1. ***Vedadero.***
   Sea $B = \{b_1, \ldots, b_n\}$ y sea el programa P

   $$\begin{array}{ll} & IF\ X_1 = b_1\ GOTO\ E \\ & IF\ X_1 = b_2\ GOTO\ E \\ & \vdots \\ & IF\ X_1 = b_n\ GOTO\ E \\ [B] & GOTO\ B \end{array}$$

   basta, entonces, notar que $A = Dom\,\Psi_P^{(1)}$.

2. ***Vedadero.***
   Sea B cofinito, entonces $\overline{B}$ es computable (puesto que es finito). Entonces B es c.e. puesto que $\overline{B}$ es co-c.e. y como $\overline{B}$ es c.e. se tiene que B es co-c.e.

   Luego, B es computable.

3. ***Falso.***
   Por ejemplo, Halt.

4. ***Falso.***
   Por ejemplo K (o cualquier no computable porque todos los conjuntos finitos son computables).

5. ***Falso.***
   Por ejemplo, $B = TOT - \{e\}$ con $e \in TOT$. En este caso, B es un conjunto de índices y si fuese computable, entonces, $TOT = B \cup \{e\}$ también lo sería.

6. ***Vedadero.***
   Como B es c..e, entonces para alguna función parcial cumputable g_B se tiene que

   $$B = \{g_B(0), g_B(1), \ldots\} = Im(g_B)$$

   Luego, $f(B) = \{g(g_B(0)), f(g_B(1)), \ldots\})\, Im(f \circ g_B)$.

   Como f total computable (en particular parcial computabl) y g es parcial computable, entonces $f \circ g$ es parcial computable y, por lo tanto, su imagen es un conjunto c.e.

7. ***Falso.***
   Por ejemplo si $B = \mathbb{N}$ y $A = K$ un conjunto c.e., se tiene que $A = f(B)$ con B conjunto computable y f una función total computable.

   Entonces, A sería computable lo que resultaría en una contradicción.

8. ***Falso.***
   Por ejemplo si $B = \mathbb{N}$ (que es p.r.) y $A = K$ un conjunto c.e., se tiene que $A = f(B)$ con B conjunto computable y f una función total computable.

   Entonces, A sería computable lo que resultaría en una contradicción.

9. ***Vedadero.***
   Si B es p.r., entonces B es computable y, entonces B es c.e. y que f sea p.r. implica que f es total computable y, entonces, $f(B)$ es un conjunto c.e.

10. ***Falso.***
    Sea $B = \mathbb{N}$ y, dado i, sea

    $$f_i(x) = \begin{cases} x & \Phi_i^{(1)}(x)\downarrow \\ \uparrow & \text{en otro caso} \end{cases}$$

    Notar que f_i es parcial computable para todo i, y además, $f_i(\mathbb{N}) = Dom\,\Phi_i^{(1)}$.

    Si la afirmación fuera cierta, debería ser $f_i(\mathbb{N})$ co-c.e. pero $f_i(\mathbb{N})$ es el dominio de una función parcial computable, y por lo tanto, es c.e.

    Luego $f_i(\mathbb{N})$ es computable para cualquier para todo i, o lo que es lo mismo, que todos los conjuntos c.e. son computables.

    En particular, si $\Phi_k^{(1)}(x)\downarrow \Leftrightarrow x \in K$ (existe por ser K c.e.), y se tiene que $f_k(\mathbb{N}) = Dom\,Dom\,\Phi_k^{(1)} = K$. Luego, K sería co-c.e., y entonces sería computable, lo que resulta ser una contradicción.

11. ***Vedadero.***
Sean f_B la función cuyo dominio es B y g_A la función característica de A. Luego,

$$\begin{array}{ll} & Z_1 \leftarrow f_B(X_1) \\ & Z_2 \leftarrow g_A(X_1) \\ [P] & IF\ Z_2 \neq 0\ GOTO\ P \end{array}$$

Ese programa termina si $X_1 \in B \setminus A$ y se indefine si no. Luego, $B \setminus A$ es c.e.

12. ***Falso.***
Por ejemplo, si $B = K$ y $A = \mathbb{N}$, entonces $A \setminus B = \mathbb{N} \setminus K = \overline{K}$ es c.e., lo que resultaría en una contradicción.

13. ***Vedadero.***
Sea P_B un programa para [la función característica de] B. Consideremos el siguiente programa P:

$$\begin{array}{ll} [C] & IF\ P_B(X) = 0\ GOTO\ C \end{array}$$

Se tiene que $\Psi_P(x) = \begin{cases} 0 & x \in B \\ \uparrow & en\ otro\ caso \end{cases}$ y, por lo tanto, $B = \{x : \Psi_P(x)\downarrow\}$. Entonces B es c.e.

14. ***Falso.***
Por ejemplo, el conjunto K es c.e. pero no es computable.

15. ***Falso.***
Por ejemplo, el conjunto K es c.e. pero si fuera co-c.e., K sería computable.

Ejercicio 3.21.
Decidir cuáles de los siguientes conjuntos son p.r., cuáles son computables, cuáles son c.e., cuáles son co-c.e. y demostrar en cada caso:

$C_1 = \{x : x \geq 10^{10} \wedge \Phi_x^{(1)}\downarrow\}$
$C_2 = \{\langle x, y\rangle : \Phi_x^{(1)}(y)\downarrow \wedge \Phi_x^{(1)}(y+1)\downarrow \wedge \Phi_x^{(1)}(y) + 1 = \Phi_x^{(1)}(y+1)\}$
$C_3 = \{x : \Phi_x^{(1)}\ computa\ la\ función\ constante\ 5\}$
$C_4 = \{\langle x, y\rangle : \forall x \in (Dom\,\Phi_x^{(1)} \cap Dom\,\Phi_y^{(1)})\ \Phi_x^{(1)}(x) < \Phi_y^{(1)}(x)\}$
$C_5 = \{x : 1 \in Dom\,\Phi_x^{(1)}\}$
$C_6 = \{x : Dom\,\Phi_x^{(1)} \subseteq \{0, \ldots, x\}\}$
$C_7 = \{x : W_x = \emptyset\}$
$C_8 = \{x : \Phi_x^{(1)}\ es\ total\ y\ \Phi_x^{(1)} < \Phi_x^{(1)}(x+1)\ \forall x \in \mathbb{N}\}$
$C_9 = \{\min W_i : i \in \mathbb{N} \wedge W_x \neq \emptyset\}$
$C_{10} = \{x : \Phi_x^{(1)}(x) = 2 * x\}$
$C_{11} = \{\langle x, y, t, i\rangle : \exists\sigma.SNAP^{(1)}(x, y, t) = \langle i, \sigma\rangle\}$
$C_{12} = \{\langle x, y, i\rangle : \exists\sigma, t.SNAP^{(1)}(x, y, t) = \langle i, \sigma\rangle\}$

Solución 3.21.

1. *Notar que el conjunto C_1 se parece a $K = \{x \mid \Phi_x^{(1)}\downarrow\}$ salvo por una cantidad finita de elementos.*

   Supongamos que C_1 es computable y como $\overline{C_1}$ (es decir, su complemento) es finito y, por lo tanto, computable y $K = C_1 \cup \overline{C_1}$. Entonces K debe ser computable, lo que resulta una contradicción.

   Luego C_1 no es computable y, por lo tanto, tampoco es p.r.

   Como K es c.e., se tiene que $K = \{x : (\exists t)R(x, t)\}$ para algún predicado binario R.

   Luego, para ver que C_1 es c.e., basta con notar que $C_1 = \{x : (\exists t)(x > 10^{10}) \wedge R(x, t)\}$. Entonces $C_1 = Dom\,f$, con f parcial computable. Por lo tanto, C_1 no es co-c.e. (puesto que de lo contario sería computable).

2. *Para ver que es c.e. se puede armar un programa que se defina exactamente en los elementos del conjunto. Sea el programa P*

$$\begin{array}{ll} [A] & IF\ \Phi_{\ell(x_1)}^{(1)}(r(x_1)) + 1 = \Phi_{\ell(x_1)}^{(1)}(r(x_1) + 1)\ GOTO\ E \\ & GOTO\ A \end{array}$$

   Por otra parte, si fuese computable, entonces también sería computable el conjunto

$$A' = \{\langle x, 0\rangle \in C_2\} = \{x : \Phi_x^{(1)}\downarrow \wedge \Phi_x^{(1)}\downarrow \wedge \Phi_x^{(1)}(0) + 1 = \Phi_x^{(1)}(1)\}$$

   pero éste es un conjunto de índices no trivial y, por lo tanto, no es computable por el Teorema de Rice.

   Luego C_2 no es computable y, por lo tanto, tampoco es p.r.

3. *Se intenta encontrar una $f(x)$ total computable tal que $x \in TOT$ si y sólo si $f(x)$ da un número que computa la función constante 5.*

   Así observando si $f(x)$ pertenece a C_3 se podría saber si $x \in TOT$, lo que constituye un absurdo.

   Se busca que para todo y, $\Phi_x^{(1)}(y)\downarrow \Leftrightarrow \ \forall y\ \Phi_{S(x,e)}^{(1)}(y) = 5$ y usando el Teorema del Parámetro $\Phi_{S(x,e)}^{(1)}(y) = \Phi_e^{(2)}(y,x)$ donde e es tal que

$$\Phi_e^{(2)}(y,x) = \begin{cases} 5 & \Phi_x^{(1)}(y)\downarrow \\ \uparrow & \text{en otro caso} \end{cases}$$

   con $\Phi_e^{(2)}(y,x)$ función parcial computable.

   Si C_3 fuera c.e. o co-c.e., entonces $S(x,e) \in C_3 \Rightarrow \Phi_{S(x,e)}^{(1)}$ computa la constante 5. Por lo tanto, $\Phi_x^{(1)}(y)\downarrow$ para todo y y, entonces, $x \in TOT$. Es decir, C_3 es una reducción de TOT y, entonces, C_3 no es c.e. ni co-c.e.

4.

5.

6.

7. *Sea $\mathcal{C} = \{\lambda\}$ donde λ es la única función con dominio vacío a los naturales. De esta manera,*

$$C_7 = \{x : \Phi_x^{(1)} = \lambda\} = \{x : \Phi_x^{(1)} \in \mathcal{C}\}.$$

   El conjunto C_7 es no vacío puesto que λ es computable y tiene números de programa. Tampoco es $\mathbb{N}$ puesto que no todo programa computa λ. Luego, por el Teorema de Rice, C_7 no es computable (y, por lo tanto, tampoco es p.r.).

   Si C_7 fuera c.e., debería existir un programa que pueda terminar siempre que otro programa esté indefinido para todas las entradas. Por lo tanto no es c.e.

   El predicado $R(x,z) = STP(\ell(z), r(z))$ verifica que el número z contiene la información de una entrada y de un tiempo de cómputo para los cuales el programa de número x se detiene. Entonces $\overline{C_4} = \{x \mid (\exists z)R(x,z)\}$ y por lo tanto $\overline{C_2}$ es co-c.e.

8. *Reduciendo TOT al conjunto C_8, se verá que C_5 no es c.e. ni co-c.e.*

   Sea la función parcial computable $g(x,y) = \begin{cases} y & \Phi_x^{(1)}(y)\downarrow \\ \uparrow & \text{en otro caso} \end{cases}$ y sea e el número del programa g tal que

   $g(x,y) = \Phi_e^{(1)}(x,y)$ y por el Teorema del Parámetro existe S p.r. tal que $\Phi_{S(x,e)}^{(1)}(y) = g(x,y)$.

   Sea, entonces, $x \in TOT$,

$\Phi_x^{(1)}(y)\downarrow$ *para todo $y \in \mathbb{N} \Rightarrow \Phi_{S(x,e)}^{(1)}(y) = g(x,y) = y$ para todo $y \in \mathbb{N} \Rightarrow S(x,e) \in C_5$ pues $\Phi_{S(x,e)}^{(1)}$ es total y creciente.*

   Si $x \notin TOT$,

$$\exists y_0 \in \mathbb{N} \text{ tal que } \Phi_x^{(1)}(y_0)\uparrow \Rightarrow \Phi_{S(x,e)}^{(1)}(y_0) = g(x,y_0)\uparrow \Rightarrow S(x,e) \notin B \text{ pues } \Phi_{S(x,e)}^{(1)} \text{ no es total.}$$

9. *Dado que $C_9 \subset \mathbb{N}$, se verá que $\mathbb{N} \subset C_9$ (es decir, que para cualquier $n \in \mathbb{N}$ se tiene que n es el mínimo del dominio de alguna función parcial computable. Por ejemplo,*

$$f_n(x) = \begin{cases} 1 & x = n \\ \uparrow & \text{en otro caso} \end{cases}$$

   Como f_n es parcial computable, entonces la computa $\Phi_e^{(1)}$ para algún $e \in \mathbb{N}$, y, entonces, se tiene que $n = \min W_e$ y $W_e \neq \emptyset$, con lo cual $n \in C_9$. Como n puede ser cualquier natural, se tiene que $\mathbb{N} \subset C_9$.

   Por lo tanto. C_9 es p.r. y por lo tanto computable y, entonces, c.e. y co-c.e.

Ejercicio 3.22.
Demostrar o refutar cada una de las siguientes afirmaciones.

a. *Si $B_1, \ldots, B_k$ son c.e. entonces $\bigcup_{n\in\{1,\ldots,k\}} B_n$ es c.e.*

b. *Si $(B_n)_{n\in\mathbb{N}}$ es una familia de conjuntos c.e. entonces $\bigcup_{n\in\mathbb{N}} B_n$ es c.e.*

c. Si $B_1, \ldots, B_k$ son c.e. entonces $\bigcap_{n\in\{1,\ldots,k\}} B_n$ es c.e.

d. Si $(B_n)_{n\in\mathbb{N}}$ es una familia de conjuntos c.e. entonces $\bigcap_{n\in\mathbb{N}} B_n$ es c.e.

Solución 3.22.

Ejercicio 3.23.
Sea B un conjunto infinito.

a. Probar que B es c.e. sii existe una función $f : \mathbb{N} \to \mathbb{N}$ inyectiva y computable tal que $Im\, f = B$ (o sea, f es una enumeración computable *de los elementos de B).*

b. Probar que B es computable sii existe una función $f : \mathbb{N} \to \mathbb{N}$ computable y estrictamente creciente tal que $Im\, f = B$ (o sea, f es una enumeración ordenada y computable *de los elementos de B).*

c. Probar que si B es c.e. entonces B contiene (al menos) un subconjunto computable infinito.

Solución 3.23.

Ejercicio 3.24.
Sea $TOT = \{x : \Phi_x^{(1)}$ es una función total$\}$.

a. Demostrar que TOT no es co-c.e.

b. Demostrar que TOT no es c.e. (Sugerencia: considerar la función $g(x) = \Phi_{f(x)}(x)+9$ donde f es una enumeración computable de TOT).

Solución 3.24.

Ejercicio 3.25.
Demostrar que los siguientes conjuntos no son c.e. ni co-c.e.

$$ID = \{x : \Phi_x^{(1)} \text{ es la función identidad}\}$$
$$S = \{x : Im\Phi_x^{(1)} = \mathbb{N}\}$$

Sugerencia: utilice una reducción de TOT usando el Teorema del Paráametro.

Solución 3.25.

Ejercicio 3.26.
Exhibir un conjunto no vacío *$C \subseteq \mathbb{N}$ tal que, para toda función f primitiva recursiva, exista un k que cumpla $k \in (C \cup Im\, f)$ pero $k \notin (C \cap Im\, f)$. Justificar apropiadamente.*

Solución 3.26.

Ejercicio 3.27.
Demostrar o refutar cada una de las siguientes afirmaciones.

a. Para toda función computable y total $f : \mathbb{N} \to \mathbb{N}$ existe una función primitiva recursiva $g : \mathbb{N} \to \mathbb{N}$ tal que $(f \circ g)$ y $(g \circ f)$ son primitivas recursivas.

b. Sea C un conjunto infinito. Si para todo $x \in C$ vale $\neg Halt(x, x)$ entonces C no es computable.

c. Si C y D son conjuntos c.e. infinitos, entonces existe $f : \mathbb{N} \to \mathbb{N}$, total, computable, inyectiva, y tal que $Im\, f = C \cup D$.

d. Si $f : \mathbb{N}^2 \to \mathbb{N}$ es una función tal que, para todo $n \in \mathbb{N}$, la función $g_n(x) = f(n, x)$ es total y computable, entonces f es total y computable.

e. La siguiente función es primitiva recursiva:

$$f(x) = \begin{cases} 1 & \text{si } Im\, \Phi_x^{(1)} \text{ es un conjunto c.e.} \\ \uparrow & \text{en otro caso} \end{cases}$$

f. Sea e un número fijo, la siguiente función no es computable:

$$f(x) = \begin{cases} x+1 & \text{si } Halt(e, e) \\ 0 & \text{en otro caso} \end{cases}$$

g. Existe una función computable f tal que para todo programa P, la función computada por el programa con número $f(\#P)$ es distinta *a la computada por P, es decir, $\Psi_P^{(1)} \neq \Phi_{f(\#p)}^{(1)}$.*

Solución 3.27.

Práctica 4

Lógica Proposicional

Ejercicio 4.1.
Sea $v : \mathbf{Prop} \to \{0,1\}$ *una valuación, donde* **Prop** *denota el conjunto de símbolos proposiciones del cálculo proposicional.*

Si sólo se conocen $v(p_1), v(p_2)$ *y* $v(p_3)$*, siendo* $v(p_1) = v(p_2) = v(p_3) = 0$*, argumentar si es posible decidir* $v \models \alpha$ *o* $v \not\models \alpha$ *en los siguientes casos:*

1. $\alpha = \neg p_1$.
2. $\alpha = ((p_5 \vee p_3) \to p_1)$.
3. $\alpha = ((p_1 \vee p_2) \to p_3)$.
4. $\alpha = \neg p_4$.
5. $\alpha = ((p_8 \to p_5) \to (p_8 \wedge \neg p_0))$.
6. $\alpha = (\neg p_1 \wedge (\neg p_2 \wedge (\neg p_3 \vee p_4)))$.
7. $\alpha = (\neg p_5 \to \neg p_4) \to (p_4 \to p_5)$.
8. $\alpha = \neg(p_6 \to p_6)$.

Solución 4.1.
Recordemos que

- $v \models \neg\alpha \Leftrightarrow v \not\models \alpha$.
- $v \models (\alpha \to \beta) \Leftrightarrow v \not\models \alpha$ *o* $v \models \beta$.
- $v \models (\alpha \wedge \beta) \Leftrightarrow v \models \alpha$ *y* $v \models \beta$.
- $v \models (\alpha \vee \beta) \Leftrightarrow v \models \alpha$ *o* $v \models \beta$.

Luego,

1. *Como* $v(p_1) = 0$*, entonces* $v(\alpha) = v(\neg p_1) = 1 \Rightarrow v \models \alpha$.
2. $v \models \alpha$ *si y sólo si* $v \models p_1$ *o* $v \not\models (p_5 \vee p_3)$*. Como* $v(p_1) = 0$*, entonces* $v \models \alpha$ *si y sólo si* $v \not\models (p_5 \vee p_3)$ *si y sólo si* $v \not\models p_5$ *y* $v \not\models p_3$ *si y sólo si* $v \not\models p_5$ *(puesto que* $v(p_1) = v(p_3) = 0$*).*
   Luego, no es posible decidir si $v \models \alpha$ *o* $v \not\models \alpha$ *puesto que no se conoce el valor de* $v(p_5)$.
3. $v \models \alpha$ *si y sólo si* $v \models p_3$ *o* $v \not\models (p_1 \vee p_2)$ *si y sólo si* $v \models p_3$ *o* $(v \not\models p_1$ *y* $v \not\models p_2)$.
   Luego, como $v(p_1) = v(p_2) = 0$ *entonces* $v \models \alpha$.
4. *Como no se conoce el valor de* $v(p_4)$*, entonces no es posible decidir si* $v \models \alpha$ *o* $v \not\models \alpha$.
5. $v \models \alpha$ *si y sólo si* $(v \models (p_8 \wedge \neg p_0))$ *o* $(v \not\models (p_8 \to p_5))$ *si y sólo si* $v \models p_8$ *y* $v \models \neg p_0$ *o* $v \models p_8$ *y* $v \not\models p_5$ *si y sólo si* $v \models p_8$ *y* $(v \models \neg p_0$ *o* $v \not\models p_5)$
   Luego, como no se conocen los valores de $v(p_0), v(p_5)$ *y* $v(p_8)$ *no es posible decidir si* $v \models \alpha$ *o* $v \not\models \alpha$.
6. $v \models \alpha$ *si y sólo si* $v \models \neg p_1$ *y* $v \models (\neg p_2 \wedge (\neg p_3 \vee p_4))$ *si y sólo si* $v \not\models p_1$ *y* $v \models \neg p_2$ *y* $v \models (\neg p_3 \vee p_4)$ *si y sólo si* $v \not\models p_1$ *y* $v \not\models p_2$ *y* $(v \models \neg p_3$ *o* $v \models p_4)$ *si y sólo si* $v \not\models p_1$ *y* $v \not\models p_2$ *y* $(v \not\models p_3$ *o* $v \models p_4)$.
   Luego, como $v(p_1) = v(p_2) = v(p_3) = 0$ *entonces* $v \not\models p_i$ *para* $i = 1, 2, 3$ *y, por lo tanto,* $v \models \alpha$.
7. $v \models \alpha$ *si y sólo si* $v \not\models (\neg p_5 \to \neg p_4)$ *o* $v \models (p_4 \to p_5)$ *si y sólo si* $(v \models \neg p_5$ *y* $v \not\models \neg p_4)$ *o* $(v \not\models p_4$ *o* $v \models p_5)$ *si y sólo si* $(v \not\models p_5$ *y* $v \models p_4)$ *o* $(v \not\models p_4$ *o* $v \models p_5)$.
   Luego, como no se conocen los valores de $v(p_4)$ *y* $v(p_5)$ *no es posible decidir si* $v \models \alpha$ *o* $v \not\models \alpha$.
8. $v \models \alpha$ *si y sólo si* $v \not\models (p_6 \to p_6)$ *si y sólo si* $v \models p_6$ *y* $v \not\models p_6$.

Ejercicio 4.2.
Dadas las siguientes fórmulas del cálculo proposicional:

- $\alpha_1 = (\neg p_1 \to (p_3 \vee p_4))$.
- $\alpha_2 = \neg(p_2 \to (p_3 \wedge p_1))$.
- $\alpha_3 = ((\neg p_2 \to p_3) \to (p_2 \vee (p_5 \to p_3)))$.

Hallar todas las valuaciones v tales que:

1. $v \models \alpha_i$.
2. $v \models \alpha_i$ *y* $v(p_j) = 0$ *si* $p_j \notin \mathbf{Var}(\alpha)$.

donde **Var** *denota al conjunto de variables proposicionales y* $\mathbf{Var}(\alpha)$ *al subconjunto de* **Var** *cuyos elementos son las variables proposicionales que aparecen en* α.

Solución 4.2.

1. - $\underline{\alpha_1 = (\neg p_1 \to (p_3 \vee p_4))}$:

$$v \models \alpha_1 \Leftrightarrow v \not\models \neg p_1 \ o\ v \models (p_3 \vee p_4)) \Leftrightarrow v \models p_1 \ o\ v \models p_3 \ o\ v \models p_4.$$

   - $\underline{\alpha_2 = \neg(p_2 \to (p_3 \wedge p_1))}$:

$$v \models \alpha_2 \Leftrightarrow v \not\models (p_2 \to (p_3 \wedge p_1)) \Leftrightarrow v \models p_2 \ y\ v \not\models (p_3 \wedge p_1) \Leftrightarrow v \models p_2 \ y\ (v \not\models p_3 \ o\ v \not\models p_1).$$

   - $\underline{\alpha_3 = ((\neg p_2 \to p_3) \to (p_2 \vee (p_5 \to p_3)))}$:

$$\begin{aligned} v \models \alpha_3 &\Leftrightarrow v \not\models (\neg p_2 \to p_3) \ o\ v \models (p_2 \vee (p_5 \to p_3)) \\ &\Leftrightarrow (v \models \neg p_2 \ y\ v \not\models p_3) \ o\ (v \models p_2 \ o\ v \models (p_5 \to p_3)) \\ &\Leftrightarrow (v \not\models p_2 \ y\ v \not\models p_3) \ o\ (v \models p_2 \ o\ (v \models p_3 \ o\ v \not\models p_5)) \end{aligned}$$

2. *Usando el inciso anterior,*

   - $\underline{\alpha_1 = (\neg p_1 \to (p_3 \vee p_4))}$:
     Son los v tales que $v(p_i) = 1$ para algún $i = 1, 3, 4$ y $v(p_j) = 0$ para todo $j \in \mathbb{N} \setminus \{1, 3, 4\}$.
   - $\underline{\alpha_2 = \neg(p_2 \to (p_3 \wedge p_1))}$:
     Son los v tales que $v \models \alpha_2$ si y sólo si $v(p_2) = 1$ y $v(p_3) = 0$ o $v(p_1) = 0$ y $v(p_j) = 0$ para todo $j \in \mathbb{N} \setminus \{1, 2, 3\}$.
   - $\underline{\alpha_3 = ((\neg p_2 \to p_3) \to (p_2 \vee (p_5 \to p_3)))}$:
     Son los v tales que $v(p_i) = 0$ para todo $i \in \mathbb{N} \setminus \{2, 3, 5\}$.

Ejercicio 4.3.
Sea $\phi \in$ **Form**.

a. Entonces tiene la misma cantidad de paréntesis derechos que de paréntesis izquierdos.

b. Se define como el peso de φ (notado por $p(\varphi)$) a la diferencia entre el número de paréntesis izquierdos contra los paréntesis derechos.

Probar que si $*$ es un conectivo binario que aparece en α entonces la expresión que aparece en α a la derecha de $*$ tiene peso negativo.

Solución 4.3.

a. Llamo $\ell(\phi)$ y $r(\phi)$ a la cantidad de paréentesis izquierdos y derechos de ϕ respectivamente. Hacemos la demostración por inducción estructural.

- *Caso Base: Sea $p \in$* **Prop** *tal que $\phi = p$.*
  Entonces $\ell(\phi) = r(\phi) = 0$.
- *Paso Inductivo: Sean $\phi, \varphi \in$* **Form** *tales que $\ell(\phi) = r(\phi)$ y $\ell(\varphi) = r(\varphi)$. Veamos que vale para $\neg\varphi$ y $\varphi \to \phi$.*

$\neg\varphi$: $\ell(\neg\varphi) = \ell(\varphi) = r(\varphi) = r(\neg\varphi)$.
$\varphi \to \phi$: $\ell(\varphi \to \phi) = \ell(\varphi) + \ell(\phi) = r(\varphi) + r(\phi) = r(\varphi \to \phi)$.

b. Hacemos la demostración por inducción estructural.

- *Caso Base: El resultado es claro si $\varphi = p$ con $p \in$* **Prop** *ya que al no haber conectivos binarios el antecedente de la implicación del enunciado es falsa y por ende la implicación es verdadera.*
  Notar que aquí hemos usado la interpretación de la implicación $\Rightarrow$ en el lenguaje declarativo del enunciado.

- *Paso Inductivo:* *De acuerdo a la definición inductiva de fórmula, φ debe ser alguno de los siguientes dos tipos:*

$$\alpha = \neg \ldots \neg(\beta_1 * \beta_2) \quad \text{o bien} \quad \alpha = (\beta_1 * \beta_2), \text{ con } \beta_1, \beta_2 \in \textbf{Form}$$

En el primer caso la negación aparece un número finito de veces. Tomemos un conectivo binario que aparece en α.

Si dicho conectivo aparece en β_1 la expresión que aparece a la derecha de dicho conectivo es $E * \beta_2$, donde E es la expresión que aparece a la derecha del mencionado conectivo en β_1.

Como tanto β_1 y β_2 tienen complejidad binaria menor que la complejidad binaria de α se sigue de la hipótesis inductiva que $p(E) < 0$. Luego, $p(E * \beta_2)) = p(E) + p(\beta_2) - 1 = p(E) - 1 < 0$ ya que de acuerdo a lo probado en el inciso anterior $p(\beta_2) = 0$.

Si el conectivo binario es $*$ la expresión que aparece a la derecha de dicho conectivo en α es β_2) que tiene peso -1, y si el conectivo binario aparece en β_2, la expresión que aparece a la derecha de dicho conectivo en α es F), donde F es la expresión que aparece a la derecha de dicho conectivo en β_2.

Aplicando de nuevo la hipótesis inductiva para β_2 se tiene que $p(F) < 0$ y luego $p(F)) = p(F) - 1 < 0$.

Lo que termina de completar la prueba.

Ejercicio 4.4.
*Sea $\phi \in$ **Form** y sean $\beta_1, \beta_2, \beta_3$ y β_4 fórmulas y sean $*_1$ y $*_2$ conectivos binarios tales que $\alpha = (\beta_1 *_1 \beta_2) = (\beta_3 *_2 \beta_4)$. Probar que, entonces, $\beta_1 = \beta_3$, $*_1 = *_2$ y $\beta_2 = \beta_4$.*

Solución 4.4.
Notemos que el primer símbolo de α en ambas expresiones es (y el último símbolo de α en las mismas es). Luego podemos simplificar estos dos paréntesis y obtenemos la igualdad

$$\beta_1 *_1 \beta_2 = \beta_3 *_2 \beta_4$$

Si $\beta_1 = \beta_3$ simplificamos ambas fórmulas y se llega a que $*_1 = *_2$ y $\beta_2 = \beta_4$.

Supongamos que $\beta_1 \neq \beta_3$. Luego tenemos dos posibilidades:

$$\beta_1 = \beta_3 *_2 E \quad \text{o bien} \quad \beta_3 = \beta_3 *_1 F, \text{ con } E \text{ y } F \text{ expresiones}$$

En el primer caso $p(\beta_1) = 0 = p(\beta_3) + p(E) = 0 + p(E) = p(E)$

Luego la expresión que aparece a la derecha de $*_1$ tiene peso 0 lo que contradice el ejercicio 4.3.

En forma análoga se prueba que el caso $\beta_3 = \beta_1 *_1 F$.

Luego vale la unicidad de escritura.

Notar que no hemos considerado el conectivo $\neg$ ya que para este caso es obvio, en efecto si $\alpha = \neg\beta_1 = \neg\beta_2$ entonces claramente $\beta_1 = \beta_2$ ya que el primer símbolo de ambos es $\neg$.

Ejercicio 4.5.
*Sean $\alpha, \beta \in$ **Form**. Decimos que α es satisfacible cuando existe una valuación v tal que $v \models \alpha$. Demostrar que:*

1. *α es tautología si y sólo si $\neg\alpha$ no es satisfacible.*
2. *$(\alpha \wedge \beta)$ es tautología si y sólo si α y β son tautologías.*
3. *$(\alpha \vee \beta)$ es contradicción si y sólo si α y β son contradicciones.*
4. *$(\alpha \to \beta)$ es contradicción si y sólo si α es tautología y β es contradicción.*

Solución 4.5.

1. *$\neg\alpha$ no es satisfacible si y sólo si para toda v valuación se tiene que $v \not\models \neg\alpha$ si y sólo si para toda v valuación se tiene que $v \models \alpha$ si y sólo si α es una tautología.*
2. *$(\alpha \wedge \beta)$ es tautología si y sólo si para toda v valuación, se tiene que $v \models \alpha \wedge \beta$ si y sólo si para toda v valuación, $v \models \alpha$ y $v \models \beta$ si y sólo si α y β son tautologías.*
3. *$(\alpha \vee \beta)$ es contradicción si y sólo si para toda v valuación se tiene que $v \not\models (\alpha \vee \beta)$ si y sólo si para toda v valuación se tiene que $v \not\models \alpha$ y $v \not\models \beta$ si y sólo si α y β son contradicciones.*
4. *$(\alpha \to \beta)$ es contradicción si y sólo si para toda v valuación se tiene que $v \not\models (\alpha \to \beta)$ si y sólo si para toda v valuación se tiene que $v \models \alpha$ y $v \not\models \beta$ si y sólo si α es tautología y β es contradicción.*

Ejercicio 4.6.
*Sea $\alpha \in$ **Form**, denotamos con $c(\alpha)$ a la complejidad de α (esto es el número de conectivos binarios que aparecen en α).*

1. a. *Sea α una tautología de la lógica proposicional. Encontrar una fórmula de complejidad mínima que sea equivalente a α. >Es única?.*

   b. *Sea α una contradicción de la lógica proposicional. Encontrar una fórmula de complejidad mínima que sea equivalente a α. >Es única?.*

2. *Encontrar para $\alpha = ((p_1 \wedge p_2) \vee (p_1 \wedge p_3))$ una fórmula de complejidad mínima que sea equivalente.*

Solución 4.6.

1. a. *Notemos primero que una fórmula es equivalente a una tautología si y sólo si dicha fórmula es también una tautología. Luego el problema se reduce a encontrar una tautología β que tenga complejidad mínima.*

   Es claro que $c(\beta) > 0$ ya que una variable proposicional no es una tautología.

   >Existe alguna tautología con complejidad 1?. Para ello hay que chequear cuáles de los cuatro conectivos posibles aparece en β.

   Es claro que en este caso el único candidato es la implicación, y podemos definir la siguiente tautología

   $$\beta = (p_1 \to p_1)$$

   Es claro que existen infinitas fórmulas de este tipo cambiando la variable proposicional p_1 por p_j con $j \neq 1$.

   b. *Al igual que en el ejemplo anterior una fórmula es equivalente a una contradicción si y sólo si dicha fórmula lo es. Sin embargo es interesante notar que cualquier contradicción que tenga complejidad mínima debe tener complejidad mayor que 2.*

   En efecto, veamos que si β es una fórmula tal que $c(\beta) = 1$ entonces no puede ser una contradicción:

   - *De la definición inductiva de fórmula las posibles fórmulas β son de alguno de los siguientes tipos:*

   (a) $\beta = \neg p_i$.
   (b) $\beta = (p_j \vee p_i)$.
   (c) $\beta = (p_j \wedge p_i)$.
   (d) $\beta = (p_j \to p_i)$.

   donde p_i, p_j son variables proposicionales arbitrarias (no necesariamente distintas).

   Es claro que ninguna de estas fórmula es una contradicción.

   Un ejemplo inmediato de una contradicción que tiene complejidad 2 es

   $$\beta = (p_1 \wedge \neg p_1)$$

   Es claro que existen infinitas fórmulas de este tipo cambiando la variable proposicional p_1 por p_j con $j \neq 1$.

2. *Para encontrar la fórmula de complejidad mínima, es interesante interpretar los conectivos como operaciones conjuntistas.*

   Para ver como es esta interpretación, supongamos que A y B son conjuntos, entonces si llamamos $p(x)$ a la proposición $x \in A$ y llamamos $q(x)$ a la proposición $x \in B$, entonces la conjunción $p(x) \wedge q(x)$ y la disyunción $p(x) \vee q(x)$ son equivalentes a la proposición $x \in A \cap B$ y $x \in A$ o $x \in B$ respectivamente.

   Luego para interpretar α desde la teoría de conjuntos podemos poner tres conjuntos A, B y C e interpretar $p_1(x)$ como la proposición $x \in A$, y análogamente interpretar $p_2(x)$ como la proposición $x \in B$ e interpretar $p_3(x)$ como la proposición $x \in C$.

   Por ende podemos interpretar a la fórmula α como la proposición $\alpha(x)$ que expresa

   $$x \in (A \cap B) \cup (A \cap C) = A \cap (B \cup B).$$

   Por medio de este argumento se traslada a las valuaciones y se prueba que α es equivalente a $(p_1 \wedge (p_2 \vee p_3))$ (notar que esta fórmula tiene complejidad 2 mientras que α tiene complejidad 3).

   *Supongamos que existe $\beta \in$ **Form** de complejidad 2 equivalente a α. Entonces β debe tener a lo sumo dos variables proposicionales, lo que implica que alguna de las variables p_1, p_2 o p_3 no debe aparecer en β. Sin pérdida de generalidad, supongamos que p_1 no aparece en β y, por lo tanto, el valor de verdad de β no depende del valor que le asignemos a p_1.*

   Sea v una valuación que vale 1 en estas tres variables. Luego $v(\alpha) = v(\beta) = 1$ ya que estamos asumiendo que α y β son equivalentes.

   Sea w la valuación que coincide con v en las variables proposicionales que aparecen en β y que toma el valor 0 en las otras variables. En particular $w(p_1) = 0$ lo que implica que $w(\alpha) = w(\beta) = 0$.

   Pero esto es imposible ya que w y v coinciden en las variables de β y luego $v(\beta) = 1 = w(\beta)$.

Ejercicio 4.7.
*Sea $\alpha \in$ **Form** una contradicción y $\varphi \in$ **Form**. Defino φ_α como la fórmula que se obtiene al reemplazar todas las variables proposicionales en φ por α. Probar que φ_α es una contradicción o una tautología.*

Solución 4.7.
Usando inducción estructural.

- *Caso Base: Sea $p \in$ **Prop** tal que $\varphi = p$.*
  Así $\varphi_\alpha = \alpha$ es una contradicción pues α lo es por hipótesis.
- *Paso Inductivo: Sean $\phi, \varphi \in$ **Form** tales que φ_α y ϕ_α son contradicciones o tautologías. Veamos que vale para $\neg\varphi$ y $\varphi \to \phi$. Notando que $(\neg\varphi)_\alpha = \neg\varphi_\alpha$ y $(\varphi \to \phi) = \varphi_\alpha \to \phi_\alpha$, la siguiente tabla presenta todas las posibilidades.*

φ_α	ϕ_α	$\neg\varphi_\alpha$	$\varphi_\alpha \to \phi_\alpha$
T	T	C	T
T	C	C	C
C	T	T	T
C	C	T	T

Ejercicio 4.8.
*Sean $\alpha, \beta \in$ **Form**.*

1. *Probar que si $\alpha \wedge \beta$ es una contingencia, entonces α es contingencia o β es contingencia.*
2. *Dadas dos valuaciones v, v', probar que si $v(p_i) = v'(p_i)$ para toda $p_i \in \mathbf{Var}(\alpha)$ entonces $v \models \alpha$ si y sólo si $v' \models \alpha$.*
3. *Usando el resultado anterior, mostrar que si $\mathbf{Var}(\alpha) \cap \mathbf{Var}(\beta) = \emptyset$, entonces*
   (i) $(\alpha \to \beta)$ es tautología si y sólo si α es contradicción ó β es tautología.
   (ii) $(\alpha \vee \beta)$ es tautología entonces α es tautología o bien β lo es.
   Mostrar con un ejemplo que el resultado es falso si las fórmulas tienen variables en común.
4. *Análogamente, probar que si α y β son contingencias y no tienen variables proposicionales en común, entonces $\alpha \wedge \beta$ es contingencia.*
5. *Sea $(\alpha \to \beta)$ una tautología. Probar que si α y β no son equivalentes, entonces existe una fórmula δ que satisface las siguientes condiciones:*
   (i) $(\alpha \to \delta)$ y $(\delta \to \beta)$ son tautologías.
   (ii) δ no es equivalente a α ni a β.

Solución 4.8.

1. *$\alpha \wedge \beta$ es una contingencia entonces existen v y w valuaciones tales que $v \models (\alpha \wedge \beta)$ y $w \not\models (\alpha \wedge \beta)$, entonces existen v y w valuaciones tales que $v \models \alpha$, $v \models \beta$ y $w \not\models \alpha$ o $w \not\models \beta$.*
   Entonces, existen v y w valuaciones tales que $v \models \alpha$ y $w \not\models \alpha$ o $v \models \beta$ y $w \not\models \beta$, entonces α es contingencia o β es contingencia.
2. *Usando Inducción Estructural.*
   - *Caso Base: Sea $p \in$ **Prop** tal que $p \in \mathbf{Var}(\alpha)$*
   - *Sea $\alpha = p$,*
     $$v \models \alpha \Leftrightarrow v(p) = 1 \underbrace{\Leftrightarrow}_{v(p)=v'(p)} v'(p) = 1 \Leftrightarrow v' \models \alpha.$$
   - *Paso Inductivo: Sean $\beta, \gamma \in$ **Form** tal que $v \models \beta$ si y sólo si $v' \models \beta$ y $v \models \gamma$ si y sólo si $v' \models \gamma$*
   - *Sea $\alpha = \neg\beta$,*
     $$v \models \alpha \Leftrightarrow v \models \neg\beta \Leftrightarrow v \not\models \beta \underbrace{\Leftrightarrow}_{v \models \beta \Leftrightarrow v' \models \beta} v' \not\models \beta \Leftrightarrow v' \models \neg\beta \Leftrightarrow v' \models \alpha.$$
   - *Sea $\alpha = (\beta \to \gamma)$,*
     $$v \models \alpha \Leftrightarrow v \not\models \beta \text{ o } v \models \gamma \underbrace{\Leftrightarrow}_{\substack{v \models \beta \Leftrightarrow v' \models \beta \\ v \models \gamma \Leftrightarrow v' \models \gamma}} v' \not\models \beta \text{ o } v' \models \gamma \Leftrightarrow v' \models (\beta \to \gamma) \Leftrightarrow v' \models \alpha.$$

3. *(i)* $(\alpha \to \beta)$ *es tautología si y sólo si para toda valuación* $v \models (\alpha \to \beta)$ *si y sólo si para toda valuación* $v \not\models \alpha$ *o* $v \models \beta$ *si y sólo si (puesto que* $\mathbf{Var}(\alpha) \cap \mathbf{Var}(\beta) = \emptyset$*) para toda valuación* $v \not\models \alpha$ *o para toda valuación* $v \models \beta$ *si y sólo si* α *es contradicción ó* β *es tautología.*

   (ii) Para ver esto asumamos por el absurdo que ni α *ni* β *son tautologías. Luego existen dos valuaciones* v_1, v_2 *tales que* $v_1 \not\models \alpha$ *y* $v_2 \not\models \beta$.

   Por otro lado para definir una valuación basta definir el valor de verdad de dicha valuación en las variables proposicionales.

   Definamos una tercer valuación v_3 *a partir de* v_1 *y* v_2 *como sigue: Si* p_j *es una variable proposicional arbitraria definimos*

$$v_3(p_j) = \begin{cases} v_1(p_j) & p_j \in \mathbf{Var}(\alpha) \\ v_2(p_j) & p_j \in \mathbf{Var}(\beta) \\ 0 & \text{en otro caso} \end{cases}$$

   Como v_3 *coincide con* v_1 *en* $\mathbf{Var}(\alpha)$ *y coincide con* v_2 *en* $\mathbf{Var}(\beta)$ *deducimos que* $v_1(\alpha) = v_3(\alpha) = 0$ *y* $v_2(\beta) = v_3(\beta) = 0$.

   Luego $v_3(\alpha \vee \beta) = 0$ *lo que es imposible pues* $(\alpha \vee \beta)$ *es una tautología. Notemos que usamos dos hechos importantes en la prueba y es que en primer lugar para definir el valor de verdad de una valuación basta definirla en las variables proposicionales; y en segundo lugar que si dos valuaciones coinciden en las variables proposicionales que aparecen en una fórmula coinciden en dicha fórmula.*

   El resultado es falso si se omite la hipótesis que $\mathbf{Var}(\alpha) \cap \mathbf{Var}(\beta) = \emptyset$*. En efecto tomemos* $\alpha = p_1$ *y* $\beta = \neg\beta$*. Es claro que* $(\alpha \vee \beta) = (p_1 \vee \neg p_1)$ *es una tautología pero ni* p_1 *ni* $\neg p_1$ *lo son.*

4. *Sean* α *y* β *contingencias, entonces existen* v_1, v_2, w_1 *y* w_2 *valuaciones tales que* $v_1 \models \alpha$, $w_1 \not\models \alpha$, $v_2 \models \beta$ *y* $w_2 \not\models \beta$.

   Sea

$$v(p) = \begin{cases} v_1(p) & \text{si } p \in \mathbf{Var}(\alpha) \\ v_2(p) & \text{si } p \in \mathbf{Var}(\beta) \\ 0 & \text{en otro caso} \end{cases}, \text{ entonces } v \models (\alpha \wedge \beta) \Leftrightarrow v \models \alpha \text{ y } v \models \beta \underbrace{\Leftrightarrow}_{\mathbf{Var}(\alpha)\cap\mathbf{Var}(\beta)=\emptyset} v_1 \models \alpha \text{ y } v_2 \models \beta$$

   Sea

$$w(p) = \begin{cases} w_1(p) & \text{si } p \in \mathbf{Var}(\alpha) \\ w_2(p) & \text{si } p \in \mathbf{Var}(\beta) \\ 0 & \text{en otro caso} \end{cases}, \text{ entonces } w \not\models (\alpha \wedge \beta) \Leftrightarrow w \not\models \alpha \text{ o } w \not\models \beta \underbrace{\Leftrightarrow}_{\mathbf{Var}(\alpha)\cap\mathbf{Var}(\beta)=\emptyset} w_1 \models \alpha \text{ y } w_2 \models \beta$$

   Por lo tanto, existen v *y* w *valuaciones tales que* $w \not\models (\alpha \wedge \beta)$ *y* $v \models (\alpha \wedge \beta)$*. Luego,* $\alpha \wedge \beta$ *es contingencia.*

5. *Para construir una fórmula* δ *que satisfaga las condiciones (i) y (ii) elijamos en primer lugar una variable proposicional* q *que no aparezca en* α *ni en* β*. Sea* $\delta = ((\alpha \wedge q) \vee (\beta \wedge \neg q))$.

   Para probar que $(\alpha \to \delta)$ *es una tautología tomemos una valuación* v *tal que* $v \models \alpha$*. Como* $(\alpha \to \beta)$ *es una tautología* $v \models \beta$*. Luego el valor de* v *en* δ *es*

$$v(\delta) = \max\{v(\alpha \wedge q), v(\beta \wedge \neg q)\} = \max\{v(q), v(\neg q)\} = \max\{v(q), 1 - v(q)\} = 1 \Rightarrow v \models \delta$$

   Veamos ahora que $(\delta \to \beta)$ *es una tautología. Como antes tomamos una valuación* v *tal que* $v \models \delta$.

   De la definición de δ *se tiene que* $v(\alpha \wedge q) = 1$ *o bien* $v(\beta \wedge \neg q) = 1$*. En el primer caso inferimos que* $v \models \alpha$ *y por ende* $v \models \beta$ *(pues* $(\alpha \to \beta)$ *es una tautología); mentras que en el segundo caso* $v \models \beta$.

   Luego en cualquiera de los dos casos obtenemos que $v \models \beta$ *lo que prueba que* $(\delta \to \beta)$ *es una tautología.*

   Para ver la condición (ii) sabemos, por hipótesis, que α *no es equivalente a* β*. Luego existe una valuación* v *tal que* $v(\alpha) \neq v(\beta)$*. Como* $(\alpha \to \beta)$ *es una tautología entonces esto obliga a que* $v(\alpha) = 0$ *y* $v(\beta) = 1$.

   Si $v(q) = 1$ *entonces de la definición de* δ *tenemos que* $v(\delta) = 0$ *lo que prueba que* $v(\delta) \neq v(\beta)$*. Luego* δ *y* β *no son equivalentes.*

   Para probar que δ *no es equivalente a* α *definamos una segunda valucación* w *tal que* w *coincide con* v *en las variables proposicionales que aparecen en* $\mathbf{Var}(\beta) \cup \mathbf{Var}(\alpha)$ *y que toma el valor* 0 *en las variables que no aparecen en* $\mathbf{Var}(\beta) \cup \mathbf{Var}(\alpha)$.

   Luego $v(\alpha) = w(\alpha) = 0$ *y* $v(\beta) = w(\beta) = 1$*. Como* $w(\neg q) = 1$ *entonces* $w(\delta) = 1$ *lo que implica que* $w \models \delta$ *y* $w \not\models \alpha$.

Ejercicio 4.9.
Sea $\varphi \in$ **Form** *tal que todos sus símbolos proposicionales aparecen una única vez, entonces* φ *es una contingencia.*

Solución 4.9.
Lo resolvemos por inducción estructural:

- *Caso Base: Si $\varphi = p$ con $p \in$ **Prop**, entonces basta con tomar como v a la valuación que manda todo a 1 y w a la que manda todo a 0.*
- *Paso Inductivo: Supongamos que vale para $\varphi, \phi \in$ **Form** y veamos que vale para $\neg\varphi$ y $\varphi \to \phi$.*

$\neg\varphi$: Observemos que si $\neg\varphi$ no tiene símbolos proposicionales repetidos entonces φ tampoco. Luego, por H.I. existen valuaciones v', w' tales que $v' \models \varphi$ y $w' \not\models \varphi$.

Basta tomar entonces $w = v'$ y $v = w'$.

$\varphi \to \phi$: Observemos que si $\varphi \to \phi$ no tiene símbolos proposicionales repetidos entonces φ y ϕ tampoco.

Luego, por H.I. existen valuaciones v_1, w_1, v_2 y w_2 tales que $v_1 \models \varphi$, $w_1 \not\models \varphi$, $v_2 \models \phi$ $w_2 \not\models \phi$.

Como v basta tomar w_1. Para armar un w, la idea es combinar v_1 con w_2 y podremos hacerlo porque los símbolos proposicionales donde nos interesan los valores de cada una no tienen intersección.

Definimos $w(x) = \begin{cases} v_1(x) & \text{si } x \text{ aparece en } \varphi \\ w_2(x) & \text{en otro caso} \end{cases}$.

Ahora, por ejercicio 4.8, resulta $w \models \varphi$; ya que $v_1 \models \varphi$ y w y v_1 coinciden en $\mathbf{Var}(\varphi)$ (es decir, coindicen en los símbolos proposicionales de φ).

Además, como $\varphi \to \phi$ no tiene símbolos repetidos resulta $w(x) = w_2(x)$ para todo símbolo x que aparece en ϕ. Luego, también por el ejercicio ejercicio 4.8 y como $w_2 \not\models \phi$, vale $w \not\models \phi$.

Concluimos entonces que $w \not\models \varphi \to \phi$.

Ejercicio 4.10.
Se dice que un conjunto de conectivos es adecuado si con ellos se puede representar cualquier función booleana.

a. Demostrar que $\{\neg, \wedge, \vee\}$ es un conjunto adecuado de operadores (sin suponer que otro conjunto es adecuado).

b. Probar, usando el resultado anterior, que también son adecuados $\{\neg, \wedge\}$, $\{\neg, \vee\}$ y $\{\neg, \to\}$.

c. Demostrar que $\{\neg\}$, $\{\vee, \wedge\}$ y $\{\vee, \to\}$ no son adecuados.

Solución 4.10.

a. Sea $f : \{0,1\}^m \to \{0,1\}$. Queremos una fórmula ψ que cumpla $f(x_1, \ldots, x_m) = 1$ si y sólo si $v_{x_1,\ldots,x_m} \models \psi$.

- *Si $f(x_1, \ldots, x_m) = 0$ para todo $(x_1, \ldots, x_m) \in \{0,1\}^m$, quiero una ψ que no valga nunca y que sólo use $\{\neg, \vee, \wedge\}$.*
  Luego, $p \wedge \neg p$.
- *Supongamos que existe $(x_1, \ldots, x_m) \in \{0,1\}^m$ tal que $f(x_1, \ldots, x_m) = 1$. Tomemos*

$$T_f = \{(e_1^1, \ldots, e_m^1), \ldots, (e_1^k, \ldots, e_m^k)\} \subset \{0,1\}^m$$

  el conjunto de todos las m-tuplas que f manda a 1.
  - *Queremos ψ tal que $v_{e_1^i,\ldots,e_m^i} \models \psi$ para todo $1 \leq i \leq k$ y que $v_{g_1^i,\ldots,g_m^i} \not\models \psi \notin T_f$.*
  - *Podemos armar ψ como unión $\bigvee_{i=1}^{k} \phi_i$ donde para todo $i \leq i \leq k$ $v_{e_1^i,\ldots,e_m^i} \models \phi_i$.*
  - *Como $v_{e_1^i,\ldots,e_m^i}(p_j) = e_j^i$, entonce si $e_j^i = 1$ se tiene que $v_{e_1^i,\ldots,e_m^i}(p_j) \models p_j$ y si si $e_j^i = 0$ se tiene que $v_{e_1^i,\ldots,e_m^i}(p_j) \models \neg p_j$.*
  - *Entonces, se puede usar $\phi_i = \bigwedge_{j=1}^{m} \gamma_j^i$ con $\gamma_j^i = \begin{cases} p_j & \text{si } e_j^i = 1 \\ \neg p_j & \text{en otro caso} \end{cases}$*

b.

- *$\{\neg, \wedge\}$: Como $\{\neg, \wedge, \vee\}$ es un conjunto adecuado de operadores, basta con escribir al elemento $\vee$ como combinación de $\{\neg, \wedge\}$. Luego,*

$$\alpha \vee \beta \equiv \neg(\neg\alpha \wedge \neg\beta)$$

- *$\{\neg, \vee\}$: Como $\{\neg, \wedge, \vee\}$ es un conjunto adecuado de operadores, basta con escribir al elemento $\wedge$ como combinación de $\{\neg, \vee\}$. Luego,*

$$\alpha \wedge \beta \equiv \neg(\neg\alpha \vee \neg\beta)$$

- $\{\neg, \to\}$: *Como* $\{\neg, \wedge, \vee\}$ *es un conjunto adecuado de operadores, basta con escribir los elementos* $\wedge$ *y* $\vee$ *como combinación de* $\{\neg, \to\}$. *Luego,*

$$\alpha \wedge \beta \equiv \neg(\alpha \to \neg\beta) \quad y \quad \alpha \vee \beta \equiv (\neg\alpha \to \beta)$$

c. *Para probar que un conjunto de operadores no es adecuado debemos encontrar una función booleana* f *(es decir,* $f : \{0,1\}^k \to \{0,1\}$*) tal que ninguna fórmula escrita únicamente con los operadores* $\{\neg\}$ *o* $\{\vee, \wedge\}$, $\{\vee, \to\}$ *y* $\{\wedge, \to\}$ *induce a* f.

$\{\neg\}$: *Veamos qué tipo de funciones puedo inducir con una sola variable y el conector* $\{\neg\}$.
Si f *está inducida por una fórmula de este estilo,* $f(0)$ *sólo puede ser* 1 *(análogamente,* $f(1)$ *sólo puede ser* 0*).*
Lo probaremos usando inducción estructural,

- *Caso Base: Si* $\alpha = p$ *con* $p \in$ **Prop**, *entonces es claro que* $f_\alpha(1) = 0$ *y* $f_\alpha(0) = 1$.
- *Paso Inductivo: Sea* $\alpha \in$ **Form** *tal que* $f_\alpha(1) = 0$ *y* $f_\alpha(0) = 1$, *luego*

$$f_{\neg\alpha}(x) = 1 - f_\alpha(x) \Rightarrow f_{\neg\alpha}(1) = 0 \text{ y } f_{\neg\alpha}(0) = 1$$

Entonces, nunca podremos inducir, por ejemplo,

$$f(1) = 1$$
$$f(0) = 0$$

$\{\vee, \wedge\}$: *Veamos qué tipo de funciones puedo inducir con una sola variable y los conectores* $\{\vee, \wedge\}$.
Si f *está inducida por una fórmula de este estilo,* $f(0)$ *sólo puede valer* 0 *(análogamente,* $f(1)$ *sólo puede ser* 1*).*
Usando inducción estructural probaremos que $f(1)$ *sólo puede ser* 1 *(para ver que* $f(0)$ *sólo puede valer* 0 *es análogo).*

- *Caso Base: Si* $\alpha = p$ *con* $p \in$ **Prop**. *Como* $p = 1$ *entonces existe* v_1 *tal que* $v_1 \models p$ *y, por lo tanto, es claro que* $f_\alpha(1) = 1$ *dado que* $v_1 \models p = \alpha$.
- *Paso Inductivo: Sea* $\varphi, \phi \in$ **Form** *tal que* $f_\varphi(1) = 1$ *y* $f_\phi(1) = 1$, *luego*
  - *Si* $\alpha = \varphi \vee \phi$, *entonces* $f_\alpha(1) = 1$ *si y sólo si* $f_\varphi(1) = 1$ *o* $f\phi(1) = 1$, *lo que es cierto por hipótesis inductiva.*
  - *Si* $\alpha = \varphi \wedge \phi$, *entonces* $f_\alpha(1) = 1$ *si y sólo si* $f_\varphi(1) = 1$ *y* $f\phi(1) = 1$, *lo que es cierto por hipótesis inductiva.*

Entonces, nunca podremos inducir, por ejemplo,

$$f(1) = 0$$
$$f(0) = 1$$

$\{\vee, \to\}$: *Veamos qué tipo de funciones puedo inducir con una sola variable y los conectores* $\{\vee, \to\}$.
Si f *está inducida por una fórmula de este estilo,* $f(0)$ *puede valer* 0 *o* 1 *y* $f(1)$ *sólo puede ser* 1.
Usando inducción estructural probaremos que $f(1)$ *sólo puede ser* 1.

- *Caso Base: Si* $\alpha = p$ *con* $p \in$ **Prop**. *Como* $p = 1$ *entonces existe* v_1 *tal que* $v_1 \models p$ *y, por lo tanto, es claro que* $f_\alpha(1) = 1$ *dado que* $v_1 \models p = \alpha$.
- *Paso Inductivo: Sea* $\varphi, \phi \in$ **Form** *tal que* $f_\varphi(1) = 1$ *y* $f_\phi(1) = 1$, *luego*
  - *Si* $\alpha = \varphi \vee \phi$, *entonces* $f_\alpha(1) = 1$ *si y sólo si* $f_\varphi(1) = 1$ *o* $f\phi(1) = 1$, *lo que es cierto por hipótesis inductiva.*
  - *Si* $\alpha = \varphi \to \phi$, *entonces* $f_\alpha(1) = 1$ *si y sólo si* $f_\varphi(1) = 0$ *o* $f_\phi(1) = 1$, *lo que es cierto por hipótesis inductiva.*

Entonces, nunca podremos inducir, por ejemplo,

$$f(1) = 0$$
$$f(0) = 0$$

$\{\wedge, \to\}$: *Veamos qué tipo de funciones puedo inducir con una sola variable y los conectores* $\{\wedge, \to\}$.
Si f *está inducida por una fórmula de este estilo,* $f(0)$ *puede valer* 0 *o* 1 *y* $f(1)$ *sólo puede ser* 1.
Usando inducción estructural probaremos que $f(1)$ *sólo puede ser* 1.

- *Caso Base: Si* $\alpha = p$ *con* $p \in$ **Prop**. *Como* $p = 1$ *entonces existe* v_1 *tal que* $v_1 \models p$ *y, por lo tanto, es claro que* $f_\alpha(1) = 1$ *dado que* $v_1 \models p = \alpha$.

- *Paso Inductivo: Sea $\varphi, \phi \in$ **Form** tal que $f_\varphi(1) = 1$ y $f_\phi(1) = 1$, luego*
  - *Si $\alpha = \varphi \wedge \phi$, entonces $f_\alpha(1) = 1$ si y sólo si $f_\varphi(1) = 1$ y $f\phi(1) = 1$, lo que es cierto por hipótesis inductiva.*
  - *Si $\alpha = \varphi \to \phi$, entonces $f_\alpha(1) = 1$ si y sólo si $f_\varphi(1) = 0$ o $f_\phi(1) = 1$, lo que es cierto por hipótesis inductiva.*

Entonces, nunca podremos inducir, por ejemplo,

$$f(0) = 0$$
$$f(1) = 0$$

Ejercicio 4.11.
*Dadas $\alpha, \beta \in$ **Form** puede escribirse $(\neg\alpha \vee \neg\beta)$ como $\alpha|\beta$ (llamada barra de* Sheffer*), y $(\neg\alpha \wedge \neg\beta)$ como $\alpha \downarrow \beta$ (llamada barra de* Nicod*).*

1. *Construir las tablas de verdad de $\alpha|\beta$ y $\alpha \downarrow \beta$.*
2. *Mostrar que $\{|\}$ y $\{\downarrow\}$ son adecuados.*
3. *Probar que si $*$ es un conectivo binario adecuado, entonces $*$ es $|$ ó $\downarrow$.*

Solución 4.11.

1.

α	β	$\neg\alpha$	$\neg\beta$	$\alpha\|\beta$	$\alpha \downarrow \beta$
1	1	0	0	0	0
1	0	0	1	1	0
0	1	1	0	1	0
0	0	1	1	1	1

2. - *$\{|\}$ es adecuado: Usando que el conjunto $\{\neg, \vee\}$ es adecuado, para ver que $\{|\}$ es adecuado basta ver que la negación y la disyunción puede expresarse usando solo este conectivo:*

$$\neg\alpha \equiv \neg\alpha \vee \neg\alpha \equiv \alpha|\alpha$$

$$\alpha \vee \beta \equiv \neg\neg\alpha \vee \neg\neg\beta \equiv \neg(\alpha|\alpha) \vee \neg(\beta|\beta) \equiv ((\alpha|\alpha)|(\beta|\beta))$$

   - *$\{\downarrow\}$ es adecuado: Usando que el conjunto $\{\neg, \wedge\}$ es adecuado, para ver que $\{\downarrow\}$ es adecuado basta ver que la negación y la conjunción puede expresarse usando solo este conectivo:*

$$\neg\alpha \equiv \neg\alpha \wedge \neg\alpha \equiv \alpha|\alpha$$

$$\alpha \wedge \beta \equiv \neg\neg\alpha \wedge \neg\neg\beta \equiv \neg(\alpha \downarrow \alpha) \vee \neg(\beta \downarrow \beta) \equiv ((\alpha \downarrow \alpha) \downarrow (\beta \downarrow \beta))$$

Ejercicio 4.12.
Sean $\top$ y $\bot$ dos conectivos de aridad cero (i.e., constantes booleanas), que cumplen $v \models \top$ y $v \not\models \bot$ para toda valuación v.

1. *Probar que $\{\to, \bot\}$ es un conjunto adecuado de conectivos.*
2. *Probar que $\{\to, \top\}$ no es un conjunto adecuado de conectivos.*

Solución 4.12.

1. *Sabemos que $\{\to, \neg\}$ es adecuado, por lo tanto, basta con observar que*

$$\neg p \equiv p \to \bot$$

   por lo tanto el conjunto $\{\to, \bot\}$ es adecuado puesto que es equivalente a $\{\to, \neg\}$.

2. *Basta con notar que no se puede inducir la función $f(1) = 0$, $f(0) = 1$.*

Ejercicio 4.13.
Sea $\mathcal{L} = \{|\}$. Podemos dar una codificación biyectiva $\# : \mathbf{Form}(\mathcal{L}) \to \mathbb{N}$ de la siguiente manera

$$\#\varphi = \begin{cases} 2 * (i-1) & \text{si } \varphi = p_i \text{ (con } i \geq 1\text{)} \\ 2 * \langle \#\alpha, \#\beta \rangle + 1 & \text{si } \varphi = \alpha|\beta \end{cases}$$

a. *Probar que existe un algoritmo primitivo recursivo que resuelve el problema de model checking proposicional. Es decir, dada una fórmula α y una valuación finita v tal que $\mathbf{Var}(\alpha) \subseteq Dom\ v$ decide si $v \models \alpha$.*

   Sugerencia: Dar una codificación biyectiva para las valuaciones y justificar la elección. Recordar que existen varios esquemas de recursión que resultan ser primitivos recursivos.

b. *Probar que existe un algoritmo primitivo recursivo que resuelve el problema de satisfacción proposicional. Es decir, dada una fórmula α decide si existe una valuación v tal que $v \models \alpha$.*

c. *Probar que existe un algoritmo primitivo recursivo que, dada una fórmula α, decide si α es una tautología.*

Solución 4.13.

Ejercicio 4.14.
Sea $\mathcal{L} = \{\wedge, \vee, \neg\}$ y α una fórmula proposicional del lenguaje $\mathcal{L}$. Sea $\alpha^\star$ la fórmula que resulta de reemplazar en $\alpha : \wedge \mapsto \vee$, $\vee \mapsto \wedge$ y para todo i, $p_i \mapsto \neg p_i$. Probar que para toda valuación v, $v \models \alpha^\star$ si y sólo si $v \not\models \alpha$.

Solución 4.14.
Usando inducción estructural

- *Casos Base: Sean $p, q \in$* **Prop**,

(i) *Si $\alpha = p$ entonces $\alpha^* = \neg p$ y vale que $v \models \alpha^*$ si y sólo si $v \models \neg p$ si y sólo si $v \not\models p$ si y sólo si $v \not\models \alpha$.*

- *Paso Inductivo: Sean $\psi, \phi \in$* **Form** *tales que para toda valuación v se tiene que $v \models \psi^*$ si y sólo si $v \not\models \psi$ y $v \models \phi^*$ si y sólo si $v \not\models \phi$.*

(i) *Si $\alpha = \neg\psi$ entonces $\alpha^* = \neg\psi^*$ y vale que $v \models \alpha^*$ si y sólo si $v \models \neg\psi^*$ si y sólo si $v \not\models \psi^*$ si y sólo si (usando paso inductivo) $v \models \psi$ si y sólo si $v \not\models \neg\psi$ si y sólo si $v \not\models \alpha$.*

(ii) *Si $\alpha = (\psi \wedge \phi)$ entonces $\alpha^* = \psi^* \vee \phi^*$ y vale que $v \models \alpha^*$ si y sólo si $v \models \psi^*$ o $v \models \phi^*$ si y sólo si (usando paso inductivo) $v \not\models \psi$ o $v \not\models \phi$ si y sólo si $v \not\models \psi \wedge \phi$ si y sólo si $v \not\models \alpha$.*

(iii) *Si $\alpha = (\psi \vee \phi)$ entonces $\alpha^* = \psi^* \wedge \phi^*$ y vale que $v \models \alpha^*$ si y sólo si $v \models \psi^*$ y $v \models \phi^*$ si y sólo si (usando paso inductivo) $v \not\models \psi$ y $v \not\models \phi$ si y sólo si $v \not\models \psi \vee \phi$ si y sólo si $v \not\models \alpha$.*

Ejercicio 4.15.
Dada una valuación v, sean p y q dos proposiciones tales que $v(p) = v(q)$. Demostrar que $v \models \varphi$ si y sólo si $v \models \varphi[p \mapsto q]$ para toda fórmula φ, donde $\varphi[p \mapsto q]$ denota la fórmula que resulta de reemplazar uniformemente la proposición p por q en φ.

Solución 4.15.
Usando inducción estructural

- *Casos Base: Sean $p_1, q_1 \in$* **Prop** *tales que $v(q_1) = v(p_1)$*

(i) *Si $\varphi = p_1$ entonces $v \models \varphi$ si y sólo si $v(p_1) = 1$ si y sólo si ($v(q_1) = v(p_1)$ por hipótesis) $v(q_1) = 1$ si y sólo si $v \models \varphi[p_1 \mapsto q_1]$.*

- *Paso Inductivo: Sean $\psi, \phi \in$* **Form** *tales que para toda valuación v se tiene que $v \models \psi$ si y sólo si $v \models \psi[p \mapsto q]$ y $v \models \phi$ si y sólo si $v \models \phi[p' \mapsto q']$.*

(i) *Si $\alpha = \neg\psi$ entonces $v \models \alpha$ si y sólo si $v \not\models \neg\alpha$ si y sólo si (usando paso inductivo) $v \not\models \psi[p \mapsto q]$ si y sólo si $v \models \neg\psi[p \mapsto q]$.*

(ii) *Si $\alpha = (\psi \wedge \phi)$ entonces $v \models \alpha$ si y sólo si $v \models \psi$ y $v \models \phi$ si y sólo si (usando paso inductivo) $v \models \psi[p \mapsto q]$ y $v \models \phi[p' \mapsto q']$ si y sólo si $v \models (\psi[p \mapsto q] \wedge \phi[p' \mapsto q'])$.*

(iii) *Si $\alpha = (\psi \vee \phi)$ entonces $v \models \alpha$ si y sólo si $v \models \psi$ o $v \models \phi$ si y sólo si (usando paso inductivo) $v \models \psi[p \mapsto q]$ o $v \models \phi[p' \mapsto q']$ si y sólo si $v \models (\psi[p \mapsto q] \vee \phi[p' \mapsto q'])$.*

Ejercicio 4.16.

1. *Decidir si los siguientes conjuntos son satisfacibles o no:*

   a. $\emptyset$.
   b. $\{p_9\}$.
   c. **Form**.
   d. *Sean Γ_1 y Γ_2 satisfacibles*
      - $\Gamma_1 \cup \Gamma_2$.
      - $\Gamma_1 \cap \Gamma_2$.

2. *Decidir si las siguientes afirmaciones son verdaderas o falsas:*

   a. $\{p_1\} \models \{p_2\}$.
   b. $\emptyset \models (p_3 \wedge p_7)$.
   c. $\emptyset \models (p_5 \vee \neg p_5)$.
   d. $\mathbf{Form} \models (p_1 \wedge \neg p_1)$.
   e. *Si* $\Gamma \subseteq \Delta$ *y* $\Gamma \models \varphi$ *entonces* $\Delta \models \varphi$.
   f. *Si* $\Gamma \models \varphi$ *y* $\Gamma \models \varphi \to \psi$ *entonces* $\Gamma \models \psi$.

Solución 4.16.

1. a. *Cualquier valuación satisface al conjunto vacío.*
   b. *Cualquier valuación v tal que $v \models p_9$ lo satisface.*
   c. **Form** *no es satisfacible, pues si existiera $v \models$* **Form** *en particular $v \models p_0$ y $v \models \neg p_0$, lo cual sería una contradicción.*
   d.
   - *No es satisfacible en general. Por ejemplo, si tomamos $\Gamma_1 = \{p_0\}$ y $\Gamma = \{\neg p_0\}$ entonces la unión no es satisfacible pero ambos conjuntos lo son.*
   - *Es satisfacible pues $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1$. Cualquier valuación que satisface a Γ_1 también satisface al conjunto más pequeño.*

2. a. ***Falso.***
   Pues cualquier valuación que cumpla $v \models (p_1 \wedge \neg p_2)$ satisface las hipótesis sin satisfacer la conclusión.
   b. ***Falso.***
   Pues las únicas consecuencias del vacío son las tautologías.
   c. ***Verdadero.***
   Pues esa fórmula es una tautología.
   d. ***Verdadero.***
   Porque (entre otros motivos) tanto p_1 como $\neg p_1$ pertenecen a **Form**.
   e. ***Verdadero.***
   Sea v valuación tal que $v \models \Delta$ entonces, como $\Gamma \subseteq \Delta$ se tiene que $v \models \Gamma$ y, por lo tanto, $v \models \varphi$. Es decir, $\Delta \models \varphi$.
   f. ***Verdadero.***
   Sea v valuación tal que $v \models \Gamma$, entonces $v \models \varphi$ y $v \models \varphi \to \psi$. Por lo tanto, la única opción posible que queda es que $v \models \psi$.

Ejercicio 4.17.
Dado un conjunto de fórmulas Γ, llamamos $\mathbf{Con}(\Gamma)$ *al conjunto de consecuencias semánticas de Γ definido como* $\mathbf{Con}(\Gamma) = \{\varphi : \Gamma \models \varphi\}$. *Sean $\Gamma, \Gamma_1, \Gamma_2, \Gamma_3$ conjuntos de fórmulas. Probar que:*

1. $\mathbf{Con}(\emptyset) = TAUT$.
2. *Si Γ es satisfacible, entonces* $\mathbf{Con}(\Gamma)$ *también.*
3. *Para cualquier* $\Gamma \subset \mathbf{Form}$ *se tiene que* $TAUT \subset \mathbf{Con}(\Gamma)$.
4. $\Gamma \subseteq \mathbf{Con}(\Gamma)$.
5. *si* $\Gamma_1 \subseteq \Gamma_2$, *entonces* $\mathbf{Con}(\Gamma_1) \subseteq \mathbf{Con}(\Gamma_2)$.
6. *si* $\Gamma_1 \subseteq \mathbf{Con}(\Gamma_2)$ *y* $\Gamma_2 \subseteq \mathbf{Con}(\Gamma_3)$ *entonces* $\Gamma_1 \subseteq \mathbf{Con}(\Gamma_3)$.
7. $\mathbf{Con}(\mathbf{Con}(\Gamma)) = \mathbf{Con}(\Gamma)$.
8. *Un conjunto de fórmulas Γ se dice independiente si para toda $\varphi \in \Gamma$ se tiene que $\varphi \notin \mathbf{Con}(\Gamma \setminus \{\varphi\})$.*

   Sea Γ independiente e infinito. Demostrar que existe $\Gamma_0 \subset \Gamma$ finito y no vacío tal que $\left\{ \bigwedge_{\varphi \in \Gamma_0} \varphi \right\}$ *es independiente.*

Solución 4.17.

1. *Para probarlo, es suficiente probar que $\varphi \in \mathbf{Con}(\emptyset)$ si y sólo si φ es tautología. Pero eso es cierto porque $\emptyset \models \varphi$ es equivalente a decir que toda valuación satisface a φ, pues toda valuación satisface a $\emptyset$.*
2. *Si Γ es satisfacible, existe $v \models \Gamma$. La misma valuación satisface a* $\mathbf{Con}(\Gamma)$.

3. *Sea* $\Gamma \subset \mathbf{Form}$ *y sea* v *una valuación tal que* $v \models \Gamma$. *Hay que ver que si* $\tau \in TAUT$ *entonces* $v \models \tau$ *pero esto es inmediato por la misma definición de tautología.*

4. *Sea* $\alpha \in \Gamma$ *y* v *valuación tal que* $v \models \Gamma$, *entonces como* $\alpha \in \Gamma$ *en particular* $v \models \alpha$ *y, por lo tanto,* $\alpha \in \mathbf{Con}(\Gamma)$.

5. *Sea* $\alpha \in \mathbf{Con}(\Gamma_1)$. *Si* v *satisface a* Γ_2, *entonces también satisface a* Γ_1 *(pues* $\Gamma_1 \subset \Gamma_2$*) y por lo tanto, satisface a* α. *Por lo tanto,* $\alpha \in \mathbf{Con}(\Gamma_2)$.

6. *Sea* $\alpha \in \Gamma_1$. *Queremos ver que* $\alpha \in \mathbf{Con}(\Gamma_3)$. *O lo que es lo mismo, que si* v *es una valución tal que* $v \models \Gamma_3$ *entonces* $v \models \alpha$.

   Como $\Gamma_2 \subseteq \mathbf{Con}(\Gamma_3)$, $v \models \beta$ *para toda fórmula* $\beta \in \Gamma_2$. *Por lo tanto,* $v \models \Gamma_2$.

   Análogamente, como $\Gamma_1 \subseteq \mathbf{Con}(\Gamma_2)$, $v \models \gamma$ *para toda fórmula* $\gamma \in \Gamma_1$.

   Finalmente, como $\alpha \in \Gamma_1$, $v \models \alpha$. *Es decir,* $\alpha \in \mathbf{Con}(\Gamma_3)$.

7. $\underline{\subseteq}$: *Sea* $\alpha \in \mathbf{Con}(\mathbf{Con}(\Gamma))$. *Si* v *satisface a* $\mathbf{Con}(\Gamma)$ *también satisface a* α. *Además, si* w *satisface a* Γ, *también satisface a* $\mathbf{Con}(\Gamma)$ *y, por lo tanto, también a* α.

   Luego, $\alpha \in \mathbf{Con}(\Gamma)$.

   $\underline{\supseteq}$: *Como* $\Gamma \subseteq \mathbf{Con}(\Gamma)$, *entonces* $\mathbf{Con}(\Gamma) \subseteq \mathbf{Con}(\mathbf{Con}(\Gamma))$.

8. *Sea* $\Gamma_0 \subset \Gamma$ *finito. Queremos que* $\bigwedge_{\varphi \in \Gamma_0} \varphi \notin \mathbf{Con}(\emptyset)$. *O sea, que* $\bigwedge_{\varphi \in \Gamma_0} \varphi$ *no es tautología.*

   Alcanza con que exista una valuación v *tal que* $v \not\models \varphi$ *para algún* $\varphi \in \Gamma_0$, *pues si existe entonces* $v \not\models \bigwedge_{\varphi \in \Gamma_0} \varphi$.

   Supongamos que no, es decir, que para toda v *valuación se tiene que* $v \models \varphi$ *para toda* $\varphi \in \Gamma_0$

   $$\Rightarrow \text{toda } \varphi \in \Gamma_0 \text{ es una tautología} \Rightarrow \varphi \in \mathbf{Con}(\Gamma \setminus \{\varphi\})$$

   lo que resulta ser una contradicción puesto que Γ *era independiente.*

Ejercicio 4.18.
Dado un conjunto de fórmulas proposicionales Γ *se define* $\mathcal{V}(\Gamma) = \{v \mid v \models \Gamma\}$. *Es decir,* $\mathcal{V}(\Gamma)$ *es el conjunto de valuaciones que satisfacen* Γ.

Demostrar que para todo par de conjuntos Γ_0 *y* Γ_1 *de fórmulas proposicionales,* $\mathbf{Con}(\Gamma_0) \subseteq \mathbf{Con}(\Gamma_1)$ *si y sólo si* $\mathcal{V}(\Gamma_1) \subseteq \mathcal{V}(\Gamma_0)$.

Solución 4.18.

$\underline{(\Rightarrow:)}$ *Supongamos que* $\mathbf{Con}(\Gamma_0) \subseteq \mathbf{Con}(\Gamma_1)$ *y sea* $v \in \mathcal{V}(\Gamma_1)$. *Entonces* $v \models \Gamma_1$ *y por definición de consecuencia semántica,* $v \models \mathbf{Con}(\Gamma_1)$.

Luego, por hipótesis, $v \models \mathbf{Con}(\Gamma_0)$ *y como* $\Gamma_0 \subseteq \mathbf{Con}(\Gamma_0)$ *se tiene que* $v \models \Gamma_0$ *y, por lo tanto,* $v \in \mathcal{V}(\Gamma_0)$.

$\underline{(\Leftarrow:)}$ *Sean* $\varphi \in \mathbf{Con}(\Gamma_0)$ *y* v *valuación con* $v \models \Gamma_1$. *Como* $v \models \Gamma_1$, *entonces* $v \in \mathcal{V}(\Gamma_1) \Rightarrow v \in \mathcal{V}(\Gamma_0) \Rightarrow v \models \Gamma_0 \Rightarrow v \models \varphi$ *(pues* $\varphi \in \mathbf{Con}(\Gamma_0)$*), entonces* $\varphi \in \mathbf{Con}(\Gamma_1)$.

Ejercicio 4.19.
Sean $\alpha, \beta \in \mathbf{Form}$.

1. *Probar que* $\mathbf{Con}(\{\beta\}) \subseteq \mathbf{Con}(\{\alpha\})$ *si y sólo si* $\alpha \to \beta$ *es tautología.*

2. *Analizar la validez de las siguientes afirmaciones:*

   a) $\mathbf{Con}(\{(\alpha \wedge \beta)\}) = \mathbf{Con}(\{\alpha\}) \cap \mathbf{Con}(\{\beta\})$.
   b) $\mathbf{Con}(\{(\alpha \vee \beta)\}) = \mathbf{Con}(\{\alpha\}) \cup \mathbf{Con}(\{\beta\})$.
   c) $\mathbf{Con}(\{(\alpha \to \beta)\}) \subseteq \mathbf{Con}(\{\beta\})$.

3. *Decidir si las siguientes afirmaciones son necesariamente verdaderas o no, para* $\Gamma_1, \Gamma_2 \subseteq \mathbf{Form}$.

   a. $\mathbf{Con}(\Gamma_1 \cap \Gamma_2) \subseteq \mathbf{Con}(\Gamma_1) \cap \mathbf{Con}(\Gamma_2)$.
   b. $\mathbf{Con}(\Gamma_1 \cap \Gamma_2) \supseteq \mathbf{Con}(\Gamma_1) \cap \mathbf{Con}(\Gamma_2)$.
   c. *Si* Γ *es un conjunto insatisfacible de fórmulas proposicionales, entonces* $\Gamma' = \{\neg\alpha \mid \alpha \in \Gamma\}$ *es satisfacible.*
   d. *Si* Γ *es un conjunto insatisfacible de fórmulas proposicionales tal que* $\Gamma' = \{\neg\alpha \mid \alpha \in \Gamma\}$ *es satisfacible, entonces en* Γ *hay al menos una contradicción.*
   e. *Si* Γ_1 *y* Γ_2 *son satisfacibles pero* $\Gamma_1 \cup \Gamma_2$ *es insatisfacible, entonces* Γ_1 *contiene necesariamente una contingencia*

f. $\Gamma_1 \cup \Gamma_2$ *insatisfacible pero* Γ_1 *insatisfacible y* Γ_2 *satisfacibles, entonces* Γ_1 *contiene necesariamente una contingencia.*

g. $\Gamma_1 \cup \Gamma_2$ *insatisfacible pero* Γ_2 *insatisfacible y* Γ_1 *satisfacibles, entonces* Γ_1 *contiene necesariamente una contingencia.*

Solución 4.19.

1. *Supongamos que* $\mathbf{Con}(\{\beta\}) \subseteq \mathbf{Con}(\{\alpha\})$ *y* $v(\alpha \to \beta) = 0$*, entonces existe una valuación* v *tal que* $v(\alpha \to \beta) = 0$.
   Entonces existe v *tal que* $v(\alpha) = 1$ *y* $v(\beta) = 0$*, pero como* $\mathbf{Con}(\{\beta\}) \subseteq \mathbf{Con}(\{\alpha\})$ *si* $v(\beta) = 0$ *entonces* $v(\alpha) = 0$*. Lo que resulta ser una contradicción.*
   Para ver la recíproca, sea $\beta' \in \mathbf{Con}(\{\beta\})$ *y sea* v *valuación tal que* $v(\alpha) = 1$ *(ie,* $v \models \{\alpha\}$*). Como* $\alpha \to \beta$ *es tautología entonces si* $v(\alpha) = 1$ *necesariamiente* $v(\beta) = 1$*. Entonces* $v \models \{\beta\}$ *y* $\{\beta\} \models \beta'$*, es decir,* $v \models \beta'$*. Por lo tanto,* $\mathbf{Con}(\{\beta\}) \subseteq \mathbf{Con}(\{\alpha\})$.

2. *a) Dado que* $(\alpha \wedge \beta) \to \alpha$ *es una tautología (análogamente para* $(\alpha \wedge \beta) \to \beta$*), entonces* $\mathbf{Con}(\{\alpha\}) \subseteq \mathbf{Con}(\{\alpha \wedge \beta\})$ *y puesto que* $\mathbf{Con}(\{\alpha\}) \cap \mathbf{Con}(\{\beta\}) \subseteq \mathbf{Con}(\{\alpha\})$ *se tiene que* $\mathbf{Con}(\{\alpha\}) \cap \mathbf{Con}(\{\beta\}) \subseteq \mathbf{Con}(\{\alpha \wedge \beta\})$.
   Por otra parte, la otra inclusión no es válida puesto que si $\alpha = p$ *y* $\beta = \neg p$ *se tiene que* $\mathbf{Con}(\{\alpha\}) \cap \mathbf{Con}(\{\beta\}) = \emptyset$ *pero* $\mathbf{Con}(\{\alpha \wedge \beta\}) = \mathbf{Con}(\emptyset) = TAUT$.
   b) Dado que $\alpha \to (\alpha \vee \beta)$ *es una tautologías (análogamente para* $\beta \to (\alpha \vee \beta)$*), entonces* $\mathbf{Con}(\{\alpha \vee \beta\}) \subseteq \mathbf{Con}(\{\alpha\})$ *y* $\mathbf{Con}(\{\alpha\}) \subseteq \mathbf{Con}(\{\alpha\}) \cup \mathbf{Con}(\{\beta\})$*, entonces* $\mathbf{Con}(\{\alpha \vee \beta\}) \subseteq \mathbf{Con}(\{\alpha\}) \cup \mathbf{Con}(\{\beta\})$.
   Por otra parte, la otra inclusión no es válida puesto que si α *es insatisfacible y* $\beta = p_1$ *entonces* $\mathbf{Con}(\{\alpha\})$ *es* **Form** *y, por lo tanto,* $\mathbf{Con}(\{\alpha\}) \cup \mathbf{Con}(\{p_1\})$ *es* **Form** *pero* $\neg p_1 \in \mathbf{Con}(\{\alpha\}) \cup \mathbf{Con}(\{p_1\})$ *pero* $\neg p_1 \notin \mathbf{Con}(\{\alpha \vee \beta\})$.
   c) Basta con notar que $(\beta \to (\alpha \to \beta))$ *es una tautología. Luego,* $\mathbf{Con}(\{(\alpha \to \beta)\}) \subseteq \mathbf{Con}(\{\beta\})$.

3. *a.* ***Verdadero.***
   Sea $\alpha \in \mathbf{Con}(\Gamma_1 \cap \Gamma_2)$*. Por definición de* **Con***, para toda valuación* v*,* $v \models \Gamma_1 \cap \Gamma_2$ *implica* $v \models \alpha$.
   Supongamos $w \models \Gamma_1$*. En particular* $w \models \Gamma_1 \cap \Gamma_2$*, porque* $\Gamma_1 \cap \Gamma_2 \subseteq \Gamma_1$.
   Por lo tanto, $w \models \alpha$*. Esto prueba que* $\alpha \in \mathbf{Con}(\Gamma_1)$ *(por simetría,* $\alpha \in \mathbf{Con}(\Gamma_2)$*) y, por lo tanto* $\alpha \in \mathbf{Con}(\Gamma_1) \cap \mathbf{Con}(\Gamma_2)$.
   b. ***Falso.***
   Sea $\Gamma_1 = \{p\}$ *y* $\Gamma_2 = \{q\}$*. Por definición,* $p \vee q \in \mathbf{Con}(\Gamma_1), \mathbf{Con}(\Gamma_2)$*. Sin embargo, claramente* $\Gamma_1 \cap \Gamma_2 = \emptyset$*, por lo tanto* $\mathbf{Con}(\Gamma_1 \cap \Gamma_2) = \mathbf{Con}(\emptyset)$ *es el conjunto de las tautologías y* $p \vee q$ *no es una tautología.*
   c. ***Falsa.***
   Por ejemplo $\Gamma = \{p, \neg p\}$ *es insatisfacible y es su propio* Γ' *por lo cual también este es insatisfacible.*
   d. ***Falsa.***
   Por ejemplo $\Gamma = \{p, q, p \wedge \neg q\}$ *es insatisfacible, no tiene contradicciones y* $\Gamma' = \{\neg p, \neg q, \neg(p \wedge \neg q)\}$ *es satisfacible por cualquier valuación* v *tal que* $v(p) = v(q) = 0$.
   e. ***Verdadero.***
   Supongamos que Γ_1 *no contiene ninguna contingencia. Como es satisfacible, no contiene ninguna contradicción y, por lo tanto,* Γ_1 *sólo contiene tautologías.*
   Como Γ_2 *es satisfacible, sea* v *tal que* $v \models \Gamma_2$*. Como* Γ_1 *sólo contiene tautologías,* $v \models \Gamma_1$.
   Por lo tanto, $v \models \Gamma_1 \cup \Gamma_2$*. Absurdo.*
   f. ***Falso.***
   Sean $\Gamma_1 = \{p \wedge \neg p\}$ *y* $\Gamma_2 = \emptyset$*. Claramente* Γ_1 *es insatisfacible y* $\Gamma_1 \cup \Gamma_2 = \Gamma_1$ *también.*
   $\Gamma_2 = \emptyset$ *es satisfacible. A su vez,* $p \wedge \neg p$ *es una contradicción y el único elemento de* Γ_1*, con lo cual* Γ_1 *no contiene contingencias.*
   g. ***Falso.***
   Sean $\Gamma_2 = \{p \wedge \neg p\}$ *y* $\Gamma_1 = \emptyset$*, que se cumplen las condiciones de satisfacibilidad. Como* Γ_1 *es vacío, no contiene contingencias.*

Ejercicio 4.20.
Sea $\Gamma \subseteq$ **Form**.

1. *Probar que si* Γ *es satisfacible y* $\Gamma' \subseteq \Gamma$*, entonces* Γ' *es satisfacible. Mostrar que la recíproca no es cierta.*
2. *Probar que* Γ *es satisfacible si y sólo si* $\mathbf{Con}(\Gamma)$ *es satisfacible.*
3. *>Es cierto que si* Γ *es satsifacible entonces para toda fórmula* α *sucede* $\Gamma \models \alpha$ *o* $\Gamma \models \neg\alpha$*?*

Solución 4.20.

1. *Recordar que un conjunto Γ es satisfactible si y sólo si existe una valuación v tal que $v \models \alpha$ para toda $\alpha \in \Gamma$.*

   Tomemos entonces un conjunto Γ que sea satisfactible y un subconjunto Γ' de Γ.

   Sea v una valuación tal que $v \models \alpha$ para toda $\alpha \in \Gamma$. Luego si $\beta \in \Gamma'$ se tiene que $\beta \in \Gamma$ y, por lo tanto, se sigue de la hipótesis que $v \models \beta$. Lo que prueba que la misma valuación que satisface a Γ entonces satisface también a Γ'.

   Para ver que la recíproca no es cierta tomemos por ejemplo el conjunto $\Gamma = \{p \vee \neg p\}$ y $\Gamma' = \{p, \neg p\}$. Es claro que Γ es satisfactible ya que la valuación que vale 1 en todas las variables proposicionales satisface a Γ. Sin embargo Γ' es claro que no es satisfactible ya que contiene a una fórmula y a su negación.

2. *(⇒): Supongamos primero que Γ es satisfactible y tomemos una valuación tal que $v \models \alpha$ para toda $\alpha \in \Gamma$.*
   Veamos que la misma valuación satisface a $\mathbf{Con}(\Gamma)$.
   Para ello sea $\beta \in \mathbf{Con}(\Gamma)$, o sea $\Gamma \models \beta$. Como v satisface a Γ entonces se sigue de la definición de consecuencia semántica que $v \models \beta$.
   Luego $\Gamma \subseteq \mathbf{Con}(\Gamma)$.

   (⇐): Como $\mathbf{Con}(\Gamma)$ es satisfacible y $\Gamma \subseteq \mathbf{Con}(\Gamma)$, entonces Γ es satisfacible.

3. *La afirmación es falsa, una manera de ver como funciona un contraejemplo es tratar de probarlo.*

   Tomemos una fórmula α arbitraria. Si $\Gamma \models \alpha$ no hay nada que probar. Si $\Gamma \not\models \alpha$ existiría una valuación v tal que v satisface a Γ y $v \not\models \alpha$. Luego $v \models \neg\alpha$.

   Si uno quisera probar que $\Gamma \models \neg\alpha$ deberíamos tomar una valuación que satisface a Γ y probar que la misma satisface a $\neg\alpha$.

   Si la valuación que tomamos es v esto se cumple por lo anterior. El problema es que Γ puede ser satisfecho por más de una valuación, con lo que si cambiamos de valuación no necesariamente la misma satisface a $\neg\alpha$.

   Un ejemplo directo es tomar como Γ el conjunto vacío. En este caso Γ es satisfecho por cualquier valuación. Por otro lado $\emptyset \models \alpha$ si y sólo si si α es una tautología. Luego si la afirmación fuese cierta para el conjunto vacío se tendría que para toda fórmula α, α es tautología o bien $\neg\alpha$ es tautología.

   Si tomamos $\alpha = p_1$ claramente no satisface ninguna de las dos condiciones.

Ejercicio 4.21.
Demostrar que son equivalentes:

1. $\neg(\alpha_1 \wedge \ldots \wedge \alpha_n) \in \mathbf{Con}(\emptyset)$.
2. *$\alpha_1, \ldots, \alpha_n$ no son simultáneamente válidas para ninguna valuación.*
3. *Existe una fórmula β tal que $\beta \in \mathbf{Con}(\{\alpha_1, \ldots, \alpha_n\})$ y $\neg\beta \in \mathbf{Con}(\{\alpha_1, \ldots, \alpha_n\})$.*
4. *$\beta \in \mathbf{Con}(\{\alpha_1, \ldots, \alpha_n\})$ para toda fórmula β.*

Solución 4.21.

(1) ⇒ (2): *Por hipótesis $\neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$ es consecuencia del conjunto vacío lo que implica que $\neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$ es una tautología lo que implica que $\neg\neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$ es una contradicción.*

Como $\neg\neg\delta$ es equivalente a δ se sigue que $(\alpha_1 \wedge \ldots \wedge \alpha_n)$ es una contradicción.

Luego no existe ninguna valuación que satisface a esta fórmula, lo que equivale a decir que no existe ninguna valuación que satisface simultáneamente a las fórmulas $\alpha_1, \ldots, \alpha_n$.

(2) ⇒ (3): *Tomemos $\beta = p_1$. Supongamos que $\beta \notin \mathbf{Con}(\{\alpha_1, \ldots, \alpha_n\})$.*

Luego existiría una valuación tal que v satisface al conjunto $\{\alpha_1, \ldots, \alpha_n\}$ y $v \not\models p_1$. Pero se sigue de la hipótesis que no existe tal valuación v. Por lo tanto se tiene que $p_1 \in \mathbf{Con}(\{\alpha_1, \ldots, \alpha_n\})$. El mismo argumento sirve para probar que $\neg p_1 \in \mathbf{Con}(\{\alpha_1, \ldots, \alpha_n\})$.

(3) ⇒ (4): *Sea α cualquier fórmula. Si existiría una valuación v tal que v satisface a $\{\alpha_1, \ldots, \alpha_n\}$ y $v \not\models \alpha$.*

Tomemos la fórmula β del punto (3). Luego como β y $\neg\beta$ pertenecen a $\mathbf{Con}(\{\alpha_1, \ldots, \alpha_n\})$ tendríamos que $v \models \beta$ y $v \models \neg\beta$ lo que es imposible.

(4) ⇒ (1): *Para probar (1) debemos ver que $\neg(\alpha_1 \wedge \ldots \wedge \alpha_n) \in \mathbf{Con}(\emptyset)$, lo que equivale a probar que $\neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$ es una tautología. O equivalentemente $(\alpha_1 \wedge \ldots \wedge \alpha_n)$ es una contradicción.*

Supongamos que no, entonces existiría una valuación v tal que $v(\alpha_i) = 1$ para todo $1 \leq i \leq n$. Luego el conjunto $\{\alpha_1, \ldots, \alpha_n\}$ sería satisfactible por la valuación v.

Por otro lado usando la hipótesis (4) tomamos como fórmula β cualquier contradicción. Por lo tanto como $\beta \in \mathbf{Con}(\{\alpha_1, \ldots, \alpha_n\})$ llegamos a que $v \models \beta$ lo que es imposible.

Ejercicio 4.22.
Sea $\leq$ la siguiente relación definida en

$$\mathbf{Form}/\equiv := [\alpha] \leq [\beta] \text{ si y sólo si } (\alpha \to \beta) \text{ es una tautología.}$$

Probar que $\leq$ es una relación de orden parcial bien definida en $\mathbf{Form}/\equiv$ con primer y último elemento.

Solución 4.22. *Para probar la buena definición de $\leq$ debemos ver que la definición no depende del representante elegido en cada clase de equivalencia, esto es si $\alpha \equiv \alpha_1$ y $\beta \equiv \beta_1$ entonces $(\alpha_1 \to \beta_1)$ es también una tautología.*

Para ver esto tomemos una valuación v tal que $v \models \alpha_1$. Como $\alpha \equiv \alpha_1$ se sigue que $v \models \alpha$. Al ser $(\alpha \to \beta)$ una tautología entonces $v \models \beta$ lo que implica $v \models \beta_1$ ya que $\beta \equiv \beta_1$.

Por lo tanto $(\alpha_1 \to \beta_1)$ es tautología lo que prueba la buena definición de $\leq$.

Para ver que $\leq$ es una relación de orden parcial debemos ver que es una relación reflexiva, antisimétrica y transitiva.

- *Como $(\alpha \to \alpha)$ es una tautología para toda $\alpha \in \mathbf{Form}$, se desprende que $\leq$ es reflexiva, es decir $[\alpha] \leq [\alpha]$.*
- *Supongamos que $\alpha, \beta \in \mathbf{Form}$ son tales que $[\alpha] \leq [\beta]$ y $[\beta] \leq [\alpha]$. Luego por definición tenemos que $(\alpha \to \beta)$ y $(\beta \to \alpha)$ son tautologías. Debemos ver que α y β son equivalentes, es decir $[\alpha] = [\beta]$.*

  Para ellp tomemos una valuación v tal que $v \models \alpha$. Como $(\alpha \to \beta)$ es una tautología se sigue que $v \models \beta$. Del mismo modo se prueba que si $v \models \beta$ se prueba que $v \models \alpha$ usando que la otra implicación es una tautología.

  Luego $v \models \alpha$ si y sólo si $v \models \beta$ para toda valuación v lo que prueba que $[\alpha] = [\beta]$. Luego $\leq$ es antisimétrica.
- *Para probar la transitividad sean α, β y δ tres fórmulas tales que $[\alpha] \leq [\beta]$ y $[\beta] \leq [\delta]$.*

  Luego $(\alpha \to \beta)$ y $(\beta \to \delta)$ son tautologías.

  Tomemos una valuación v tal que $v \models \alpha$. Como $(\alpha \to \beta)$ es una tautología entonces $v \models \beta$ y como $(\beta \to \delta)$ es una tautología inferimos que $v \models \delta$ lo que prueba que $(\alpha \to \delta)$ es una tautología. Por lo tanto $[\alpha] \leq [\delta]$ lo que prueba que $\leq$ es transitiva.

Para ver que esta relación tiene primer elemento, al que llamamos $\bot$, basta ver que $\bot$ no es otra cosa que la clase de equivalencia de cualquier contradicción, es decir el conjunto de las contradicciones.

En efecto es inmediato ver que si α es una contradicción entonces $(\alpha \to \beta)$ es siempre una tautología para toda fórmula

lo que implica que $\bot = [\alpha] \leq [\beta]$ para toda $[\beta] \in \mathbf{Form}/\equiv$ probando de este modo que $\bot$ es el primer elemento de $\mathbf{Form}/\equiv$.

En forma análoga se prueba que $\top = [\delta]$ es el último elemento de $\mathbf{Form}/\equiv$ siendo δ cualquier tautología (por ejemplo tomar $\delta = (p_{\to} p_1)$).

Práctica 5

Sistemas Deductivos para la Lógica Proposicional y Aplicaciones de Compacidad

Salvo que se indique lo contrario, no asumir que SP es correcto ni completo.

Ejercicio 5.1.
Demostrar que las siguientes afirmaciones son verdaderas

a. $\{\neg\alpha\} \vdash (\alpha \to \beta)$.

b. $\{\alpha \to \beta, \beta \to \gamma\} \vdash \alpha \to \gamma$.

c. $\vdash (\alpha \to (\beta \to (\beta \to \alpha)))$.

d. $\vdash (\neg\alpha \to \neg\beta) \to (\beta \to \alpha)$.

e. $\forall\alpha, \{\neg\neg\alpha\} \vdash (\neg\alpha \to \beta)\ \forall\beta$.

Sugerencia: recordar que en SP vale el teorema de la deducción.

Solución 5.1.

a. Podemos observar intuitivamente que podemos derivar la verdad de una implicación a partir del hecho de que no vale su antecedente.

Este razonamiento surge de la interpretación de los símbolos, que, en términos de la consecuencia sintáctica, nos es indiferente. Lo que importa es la forma en la que está definido nuestro sistema deductivo.

Damos explícitamente una derivación:

$1.\neg\alpha$	
$2.\neg\alpha \to (\neg\beta \to \neg\alpha)$	$(SP1)$
$3.\neg\beta \to \neg\alpha$	$(MD\ 1\ y\ 2)$
$4.(\neg\beta \to \neg\alpha) \to (\alpha \to \beta)$	$(SP3)$
$5.(\alpha \to \beta)$	$(MD\ 3\ y\ 4)$

b.

$1.\beta \to \gamma$	
$2.(\beta \to \gamma) \to (\alpha \to (\beta \to \gamma))$	$(SP1)$
$3.\alpha \to (\beta \to \gamma)$	$(MD\ 1\ y\ 2)$
$4.(\alpha \to (\beta \to \gamma)) \to ((\alpha \to \beta) \to (\alpha \to \gamma))$	$(SP2)$
$5.(\alpha \to \beta) \to (\alpha \to \gamma)$	$(MD\ 3\ y\ 4)$
$6.\alpha \to \beta$	
$7.\alpha \to \gamma$	$(MD\ 5\ y\ 6)$

c.

$1.(\alpha \to ((\beta \to \alpha) \to (\beta \to (\beta \to \alpha)))) \to ((\alpha \to (\beta \to \alpha)) \to (\alpha \to (\beta \to (\beta \to))))$	$(SP2)$
$2.((((\beta \to \alpha) \to (\beta \to (\beta \to \alpha))) \to (\alpha \to (((\beta \to \alpha) \to (\beta \to (\beta \to \alpha)))))$	$(SP1)$
$3.((\beta \to \alpha) \to (\beta \to (\beta \to \alpha)))$	$(SP1)$
$4.(\alpha \to ((\beta \to \alpha) \to (\beta \to (\beta\alpha))))$	$(MD\ 2\ y\ 3)$
$5.((\alpha \to (\beta \to \alpha)) \to (\alpha \to (\beta \to (\beta \to \alpha))))$	$(MD\ 1\ y\ 3)$
$6.(\alpha \to (\beta \to \alpha))$	$(SP1)$
$7.(\alpha \to (\beta \to (\beta \to \alpha)))$	$(MD\ 5\ y\ 6)$

d. Basta con notar que al reemplazar representa una instanciación del axioma $SP3$ usando los reemplazos $\varphi = \alpha$ y $\psi = \beta$.

e.

$1.(\neg\neg\alpha \to (\neg\beta \to \neg\neg\alpha))$	$(SP1)$
$2.\neg\neg\alpha$	
$3.(\neg\beta \to \neg\neg\alpha)$	$(MD\ 1\ y\ 2)$
$4.((\neg\beta \to \neg\neg\alpha) \to (\neg\alpha \to \beta))$	$(SP3)$
$5.(\neg\alpha \to \beta)$	$(MD\ 3\ y\ 4)$

Ejercicio 5.2.

a. Probar que si α es una fórmula, entonces $(\neg\neg\alpha \to \alpha)$ es demostrable.

b. Sea α una fórmula, entonces $(\alpha \to \neg\neg\alpha)$ es demostrable.

c. Sean α y β fórmulas, entonces $((\alpha \to \beta) \to (\neg\neg\alpha \to \neg\neg\beta))$ es demostrable.

d. Sean α y β fórmulas, entonces $((\neg\neg\alpha \to \neg\neg\beta) \to (\alpha \to \beta))$ es demostrable.

e. Sean α y β fórmulas, entonces $((\alpha \to \beta) \to (\neg\beta \to \neg\alpha))$ es demostrable.

Solución 5.2.

a. Para ello usaremos el teorema de la deducción. Luego, basta probar que $\{\neg\neg\alpha\} \vdash \alpha$.

Sea β cualquier fórmula demostrable. Por e1 ejercicio anterior reemplazando β por $\neg\beta$ tenemos que se tiene que $\{\neg\neg\alpha\} \vdash (\neg\alpha \to \neg\beta)$.

Por otro lado podemos aplicar de nuevo el axioma $SP3$ para la fórmula $((\neg\alpha \to \neg\beta) \to (\beta \to \alpha))$. Aplicando modus ponens obtenemos $(\beta \to \alpha)$ y como β es demostrable entonces α es demostrable a partir de $\{\neg\neg\alpha\}$.

La secuencia de pasos para llegar a la prueba de α a partir de $\neg\neg\alpha$ se resume como sigue, donde elegimos como fórmula β cualquier instancia del axioma $SP1$

$1.(\neg\neg\alpha \to (\beta \to \neg\neg\alpha))$	$(SP1)$
$2.\neg\neg\alpha$	
$3.(\beta \to \neg\neg\alpha)$	$(MD\ 1\ y\ 2)$
$4.((\beta \to \neg\neg\alpha) \to (\neg\alpha \to \neg\beta))$	$(SP3)$
$5.(\neg\alpha \to \neg\beta)$	$(MD\ 3\ y\ 4)$
$6.((\neg\alpha \to \neg\beta) \to (\beta \to \alpha))$	$(SP3)$
$7.(\beta \to \alpha)$	$(MD\ 5\ y\ 6)$
$8.\beta$	$(SP1)$
$9.\alpha$	$(MD\ 7\ y\ 8)$

b. Al igual que el inciso anterior usaremos el teorema de la deducción. Luego debemos probar que $\{\alpha\} \vdash \neg\neg\alpha$.

La siguiente es una secuencia que genera una prueba de $\neg\neg\alpha$ a partir de α.

$1.(\neg\alpha \to \neg\neg\neg\alpha)$	*(usando el inciso anterior)*
$2.((\neg\alpha \to \neg\neg\neg\alpha) \to (\neg\neg\alpha \to \alpha))$	$(SP3)$
$3.(\neg\neg\alpha \to \alpha)$	$(MD\ 1\ y\ 2)$
$4.\neg\neg\alpha$	
$5.\alpha$	$(SP3)$

c. Para ello usaremos el teorema de la deducción para probar que $\{(\alpha \to \beta), \neg\neg\alpha\} \vdash \neg\neg\beta$.
La siguiente es una prueba de $\neg\neg\beta$ a partir de $\{(\alpha \to \beta), \neg\neg\alpha\}$

$1.\neg\neg\alpha$	
$2.(\neg\neg\alpha \to \alpha)$	*(usando el inciso anterior)*
$3.\alpha$	$(MD\ 1\ y\ 2)$
$4.(\alpha \to \beta)$	
$5.\beta$	$(MD\ 3\ y\ 4)$
$6.(\beta \to \neg\neg\beta)$	*(usando el inciso anterior))*
$7.\neg\neg\beta$	

d. Es similar al ejercicio anterior.

e. Esta fórmula es lo que se llama el contra recíproco.
Usando de nuevo el teorema de la deducción basta ver que $\{(\alpha \to \beta), \neg\beta\} \vdash \neg\alpha$. La siguiente es una prueba de esta deducción:

$1.(\alpha \to \beta)$	
$2.((\alpha \to \beta) \to (\neg\neg\alpha \to \neg\neg\beta))$	*(usando el inciso anterior)*
$3.(\neg\neg\alpha \to \neg\neg\beta)$	$(MD\ 1\ y\ 2)$
$4.((\neg\neg\alpha \to \neg\neg\beta) \to (\neg\beta \to \neg\alpha))$	$(SP3)$
$5.(\neg\beta \to \neg\alpha)$	$(MD\ 3\ y\ 4)$
$6.\neg\beta$	
$7.\neg\alpha$	$(MD\ 5\ y\ 6)$

Ejercicio 5.3.

a. Demostrar que (toda instanciación de) SP1 es una tautología.

b. Demostrar que (toda instanciación de) SP3 es una tautología.

c. Demostrar que si las premisas de la regla MP son tautologías, el resultado es una tautología.

d. Suponiendo además que (todas las instanciaciones de) SP1 y SP2 también son tautologías, demostrar que SP es correcto.

Solución 5.3.

a. Tenemos que probar que todas las valuaciones hacen verdadera a todas las instancias de $SP1$.
Supongamos que no es el caso. Entonces, deben existir α, β y v tal que $v \not\models \alpha \to (\beta \to \alpha)$. Luego,

$v \not\models \alpha \to (\beta \to \alpha)$	*entonces*	$v \models \alpha$ *y* $v \not\models \beta \to \alpha$	*(por definición de* $\models$*)*
	entonces	$v \models \alpha, v \models \beta$ *y* $v \models \neg\alpha$	*(por definición de* $\models$*)*

lo cual es absurdo, al que arribamos por suponer que existía una instancia de $SP1$ no era una tautología.

b. Tenemos que probar que todas las valuaciones hacen verdadera a todas las instancias de $SP3$.
Supongamos que no es el caso. Entonces, deben existir α, β y v tal que $v \not\models (\neg\alpha \to \neg\beta) \to (\beta \to \alpha)$. Luego,

$v \not\models (\neg\alpha \to \neg\beta) \to (\beta \to \alpha)$	*entonces*	$v \not\models (\beta \to \alpha)$ *y* $v \models (\neg\alpha \to \neg\beta)$	*(por definición de* $\models$*)*
	entonces	$v \not\models \alpha, v \models \beta$ *y* $v \not\models \neg\alpha$ *o* $v \models \neg\beta$	*(por definición de* $\models$*)*

lo cual es absurdo, al que arribamos por suponer que existía una instancia de $SP3$ no era una tautología.

c. Supongamos que existe v valuación tal que $v \not\models \beta$. Como las premisas de la regla MP son tautologías, entonces para la misma v se tiene que $v \models \alpha$ y $v \models (\alpha \to \beta)$.

Luego, como $v \models (\alpha \to \beta)$ se sabe que $v \models \beta$ o $v \not\models \alpha$ pero como $v \models \alpha$ por hipótesis, se sigue que $v \models \beta$.

Entonces $v \not\models \beta$ y $v \models \beta$, lo que resulta ser una contradicción.

Luego, si las premisas de la regla MP son tautologías, el resultado es una tautología.

d. Visto en la teórica.

Ejercicio 5.4.
Sea α una fórmula y sea v una valuación tal que $v \models \alpha$. Si $p_1, \ldots, p_n$ son las variables proposicionales que aparecen en α entonces

$$\{\varepsilon_1 p_1, \ldots, \varepsilon_n p_n\} \vdash \alpha$$

donde para cada $i \in \{1, \ldots, n\}$ se tiene que $\varepsilon_i p_i$ coincide con p_i si $v(p_i) = 1$ y $\varepsilon_i p_i = \neg p_i$ si $v(p_i) = 0$.

Solución 5.4.
Los que nos dice este resultado es que si una valuación satisface a una fórmula, la misma es deducible del conjunto formado por la variables o negación de las variables, donde la negación que afecta a cada variable va a aparecer si y sólo si la valuación en dicha variable es 0.

Veamos con un ejemplo el mecanismo de la prueba y luego daremos la prueba general.

Supongamos que $\alpha = (p_1 \to p_2)$. Sea v una valuación que satisface a α. Tenemos dos posibilidades:

(i) $v(p_1) = 0$.

(ii) $v(p_1) = v(p_2) = 1$.

Notar que en el caso (i) no importa el valor que se le asigna a p_2.

Veamos como funciona la prueba en el caso (ii). Luego debemos ver que $\{p_1, p_2\} \vdash (p_1 \to p_2)$. Esto es claro ya que por el teorema de la deducción si pasamos la variable p_1 al conjunto de hipótesis debemos ver que $\{p_1, p_2\} \vdash p_2$ (hecho que es obvio ya que p_2 forma parte del conjunto de hipótesis).

Veamos ahora la prueba general. Para la prueba asumiremos que el conjunto de los conectivos que aparecen en α están incluídos en $\{\neg, \to\}$.

Para la prueba haremos inducción en la complejidad binaria $cb(\alpha)$ de α. Supongamos que $cb(\alpha) = 0$. Luego $\alpha = \neg\neg \ldots \neg p_i$ con p_i una variable proposicional. Tenemos dos casos:

(i) la cantidad de veces que aparece $\neg$ en α es par, digamos $2k$. En este caso la valuación que satisface a α toma el valor 1 en p_i, luego debemos probar que $\{p_i\} \vdash \neg\neg \ldots \neg p_i$. Esto se prueba fácilmente por inducción en k y usando el ejercicio 2.

(ii) la cantidad de veces que aparece la negación es impar. En este caso la valuación que satisface a α toma el valor 0 en p_i, luego debemos probar que $\{p_i\} \vdash \neg\neg \ldots \neg p_i$. El argumento es el mismo que en el caso anterior.

Supongamos ahora que $cb(\alpha) > 0$. Tenemos los siguientes casos:

(i) $\alpha = (\alpha_1 \to \alpha_2)$ con α_1, α_2 fórmulas. Sea v una valuación que satisface a α.

Si $v(\alpha_1) = 0$ entonces $v(\neg\alpha_1) = 1$. Por hipótesis inductiva se tiene que $\{\varepsilon_1 p_1, \ldots, \varepsilon_n p_n\} \vdash \neg\alpha_1$ donde $p_1, \ldots, p_n$ son las variables proposicionales que aparecen en α_1.

Como $(\neg\alpha_1 \to (\alpha_1 \to \alpha_2))$ es demostrable se sigue, por Modus ponens, que $\{\varepsilon_1 p_1, \ldots, \varepsilon_n p_n\} \vdash \neg\alpha$. Como las variables de α contiene a las variables que aparecen en α_1 se sigue el resultado.

Si $v(\alpha_1) = v(\alpha_2) = 1$, se tiene por hipótesis inductiva que $\{\varepsilon_1 p_1, \ldots, \varepsilon_n p_n\} \vdash \neg\alpha_1$ y $\{\varepsilon_1 p_1, \ldots, \varepsilon_n p_n\} \vdash \neg\alpha_2$, donde $p_1, \ldots, p_n$ son las variables que aparecen en α.

Notar que en el argumento inductivo las variables que se usan en la hipótesis inductiva son las que aparecen en α_1 y en α_2, pero si agregamos más variables el argumento se mantiene ya que si agrandamos el conjunto de hipótesis la deducción sigue siendo válida.

Como $(\alpha_2 \to (\alpha_1 \to \alpha_2))$ es una instancia del axioma $SP1$ aplicando Modus ponens se sigue el resultado para α.

(ii) $\alpha = \neg \ldots \neg(\alpha_1 \to \alpha_2)$, donde $\neg$ aparece un número finito de veces y α_1, α_2 son fórmulas.

Este caso se resuelve de manera similar al caso anterior.

Ejercicio 5.5.
Demostrar o refutar las siguientes afirmaciones:

a. Si $\Gamma \vdash \alpha$ y $\Gamma \vdash \beta$, entonces $\Gamma \vdash (\alpha \wedge \beta)$.

b. Existe un conjunto consistente Γ y una fórmula α tal que $\vdash \alpha$ y $\Gamma \cup \{\neg\alpha\}$ es consistente.

Solución 5.5.

a. ***Verdadero.***

Para poder trabajar con nuestro sistema deductivo, recordamos que $\alpha \wedge \beta$ *es equivalente a* $\neg(\alpha \to \neg\beta)$.

Consideremos el conjunto Γ *tal que* $\Gamma \vdash \alpha$ *y* $\Gamma \vdash \beta$ *y veamos que* $\Gamma \vdash \neg(\alpha \to \neg\beta)$.

Por la proposición 1, vemos que lo que queremos probar es equivalente a probar que $\Gamma \cup \{\alpha \to \neg\beta\} = \Gamma'$ *es inconsistente. Buscamos una fórmula* δ *tal que* $\Gamma' \vdash \delta$ *y* $\Gamma' \vdash \neg\delta$.

Como $\Gamma \subseteq \Gamma'$ *y* $\Gamma \vdash \beta$, *entonces* $\Gamma' \vdash \beta$. *Veamos que también existe una derivación de* $\neg\beta$ *a partir de* Γ', *y así llegamos a que es incosistente.*

Sea $\alpha_1, \ldots, \alpha_n$ $(\alpha_n = \alpha)$ *una derivación de* α *a partir de* $\Gamma \subseteq \Gamma'$. *Como* $(\alpha \to \neg\beta) \in \Gamma'$, *podemos construir la siguiente cadena de fórmulas:*

$$
\begin{array}{ll}
1.\alpha_1 & \\
2.\alpha_2 & \\
\vdots & \\
n-1.\alpha_{n-1} & \\
n.\alpha & \\
n+1.\alpha \to \neg\beta & ((\alpha \to \neg\beta) \in \Gamma') \\
n+2.\neg\beta & (MD\ n\ y\ n+1)
\end{array}
$$

Notar que se trata de una derivación válida de $\neg\beta$ *a partir de* Γ'. *Luego,* $\Gamma' \vdash \beta$ *y* $\Gamma' \vdash \neg\beta$ *y, por lo tanto,* Γ' *es inconsistente.*

b. ***Falso.***

Si α *es teorema, entonces es consecuenta sintáctica de cualquier conjunto, en particular de* $\Gamma \cup \{\neg\alpha\}$. *Además, todas las fórmulas de un conjunto son consecuencias sintácticas del mismo, por lo que, por un lado tenemos que* $\Gamma \cup \{\neg\alpha\} \vdash \alpha$ *y, por esto último, que* $\Gamma \cup \{\neg\alpha\} \vdash \neg\alpha$. *Es decir,* $\Gamma \cup \{\neg\alpha\}$ *inconsistente por definición.*

Luego, $\Gamma \cup \{\neg\alpha\}$ *es inconsistente para cualquier conjunto de fórmulas* Γ *y cualquier teorema.*

Ejercicio 5.6.
Sea A un conjunto consistente, si definimos el conjunto $B = A \cup \{q\}$ *con q una proposición fresca, >será B extensible a uno maximal consistente?*

Solución 5.6.
Primero tenemos que ver que el conjunto B sea consistente ya que si no lo es nunca vamos a poder extenderlo a uno maximal consistente. Supongamos que $A \cup \{q\}$ *es inconsistente entonces debe suceder que para algún* ϕ

$$A \cup \{q\} \vdash \phi \quad y \quad [A \cup \{q\} \vdash \neg\phi,\ es\ decir\ A \cup \{q\} \vdash \phi \wedge \neg\phi$$

Como la demostración es finita, entonces usa finitos elementos de $A \cup \{q\}$. *Entonces, existen* $\alpha_1, \ldots, \alpha_n \in A$ *tal que* $\{\alpha_1, \ldots, \alpha_n, q\} \vdash \phi \wedge \neg\phi$.

Observemos que

- *La demostración debe usar la variable proposicional q en algún momento puesto que si no fuera así entonces* $\{\alpha_1, \ldots, \alpha_n\} \vdash \phi \wedge \neg\phi$ *lo que resultaría ser absurdo ya que el conjunto original A era consistente.*
- *La variable* $q \notin \mathbf{Var}(\alpha_i)$ *para todo i. Sabemos esto ya que el conjunto A se formó a partir del lenguaje* $\mathcal{L}$ *donde la variable q no existía.*

Continuando con el ejercicio sabemos que $\{\alpha_1, \ldots, \alpha_n, q\} \vdash \phi \wedge \neg\phi$ *y por el Teorema de la Deducción vale que* $\{\alpha_1, \ldots, \alpha_n\} \vdash q \to (\phi \wedge \neg\phi)$.

Ahora tenemos que ver lo siguiente, como tenemos una demostración para $q \to (\phi \wedge \neg\phi)$, *en cada paso de la demostración que se use q podemos reemplazar la q por una fórmula* φ *arbitraria y la demostración seguirá siendo válida.*

Estrictamente deberíamos reemplazar uniformemente en todos los α_i, *en q y en* φ *pero no es necesario reemplazar en los* α_i *ya que por las observaciones los mismos no cambiarán. Por lo tanto podemos asegurar que vale*

$$\{\alpha_1, \ldots, \alpha_n\} \vdash \varphi \to (\phi' \wedge \neg\phi')$$

donde ϕ' *es el resultado de reemplazar uniformemente q por* φ *en* ϕ. *Ahora debemos notar que lo siguiente siempre es cierto*

$$\{\alpha_1, \ldots, \alpha_n\} \vdash \alpha_1$$

ya que α_1 pertenece al conjunto.

Finalmente si instanciamos $\varphi = \alpha_1$ juntando ambas observaciones se tiene que

$$\left.\begin{array}{r}\{\alpha_1,\ldots,\alpha_n\}\vdash\alpha_1\\ \{\alpha_1,\ldots,\alpha_n\}\vdash\alpha_1\to(\phi'\wedge\neg\phi')\end{array}\right\}\underbrace{\Rightarrow}_{MP}\{\alpha_1,\ldots,\alpha_n\}\vdash(\phi'\wedge\neg\phi')$$

y llegamos a que el conjunto original A es inconsistente lo cual es un absurdo.

Ahora que ya probamos que $B = A\cup\{q\}$ es consistente podemos usar el Lema de Lindenbaum para extender a B a un conjunto B' maximal consistente.

Ejercicio 5.7.
Sea Γ un conjunto de fórmulas del lenguaje $\{\neg,\to\}$. Demostrar que Γ es inconsistente (i.e. existe β tal que $\Gamma\vdash\beta$ y $\Gamma\vdash\neg\beta$) si y sólo si $\Gamma\vdash\alpha$ para todo α.

Solución 5.7.

$\underline{(\Rightarrow)}$: *Como Γ es inconsistente, sea β tal que $\Gamma\vdash\beta$ y $\Gamma\vdash\neg\beta$. Luego, si $\{\neg\beta,\beta\}\vdash\alpha$ entonces $\Gamma\vdash\alpha$ para todo α.*

Sabemos que $\Gamma\cup\{\neg\alpha\}$ es inconsistente si y sólo si $\Gamma\vdash\alpha$. Por lo tanto,

$$\{\neg\beta,\beta\}\vdash\alpha\Leftrightarrow\{\neg\beta,\beta,\neg\alpha\}\text{ es inconsistente}$$

lo cual es cierto puesto que el conjunto $\{\neg\beta,\beta,\alpha\}$ es inconsistente ($\neg\beta$ y β están en ese conjunto).

Otra manera de ver que $\{\neg\beta,\beta\}\vdash\alpha$ es usando los axiomas:

$1.\neg\beta$	
$2.\neg\beta\to(\neg\alpha\to\neg\beta)$	$(SP1)$
$3.\neg\alpha\to\neg\beta$	$(MD\ 1\ y\ 2)$
$4.(\neg\alpha\to\neg\beta)\to(\beta\to\alpha)$	$(SP3)$
$4.\beta\to\alpha$	$(MD\ 3\ y\ 4)$
$5.\beta$	
$6.\alpha$	$(MD\ 4\ y\ 5)$

$\underline{(\Leftarrow)}$: *Como $\Gamma\vdash\alpha$ para todo α, en particular para $\alpha_1=\varphi$ y $\alpha_2=\neg\varphi$ donde $\varphi\in$ **Form**. Luego, $\Gamma\vdash\alpha_1$ y $\Gamma\vdash\alpha_2$ y, por lo tanto, Γ es inconsistente.*

Ejercicio 5.8.
Decidir si las siguientes afirmaciones son verdaderas o falsas.

a. *$\{p\}$ es consistente.*

b. *$\{p\}$ es maximal consistente.*

c. *Si Γ es maximal consistente, entonces es infinito.*

d. **Prop** *es maximal consistente.*

Solución 5.8.

a. ***Verdadero.***

El conjunto $\{p\}$ es consistente porque es satisfacible.

b. ***Falso.***

No es es maximal consistente porque $\{p,q\}$ es un conjunto consistente que lo contiene estrictamente.

c. ***Verdadero.***

Sea Γ un conjunto maximal consistente.

Si Γ fuera vacío, claramente no sería maximal consistente, así que podemos suponer que existe una $\varphi\in\Gamma$.

De la consistencia de Γ podemos deducir que el conjunto $\Gamma\cup\{(\varphi\wedge\varphi)\}$ es consistente, así que la fórmula $(\varphi\wedge\varphi)$ pertenece a Γ.

De la misma manera, la fórmula $(\varphi\wedge(\varphi\wedge\varphi))$ pertenece a Γ.

Inductivamente, si definimos

$$\begin{cases}\varphi_0=\varphi\\ \varphi_{n+1}=(\varphi_n\wedge\varphi)\end{cases}$$

entonces $\varphi_n\in\Gamma$ para todo número natural n. En particular, Γ es infinito.

d. ***Falso.***

$\mathbf{Prop} \cup \{(p_1 \wedge p_2)\}$ *es un conjunto consistente estrictamente más grande que* $\mathbf{Prop}$.

Ejercicio 5.9.
Demostrar o refutar las siguientes afirmaciones:

a. Una fórmula es un teorema si y sólo si pertenece a todo conjunto maximal consistente.

b. Si Γ_1 *y* Γ_2 *son conjuntos inconsistentes, entonces* $\Gamma_1 \cap \Gamma_2$ *no es maximal consistente.*

c. Si Γ *es un conjunto consistente y definimos* $\Gamma' = \{\neg\alpha \to \beta : \alpha \in \Gamma, \beta \in \mathbf{Form}\}$ *entonces existe un conjunto* Δ *maximal consistente tal que* $\Gamma' \subseteq \Delta$.

Solución 5.9.

a. ***Verdadero.***

$\underline{(\Rightarrow)}$: *Si consideramos un teorema* α *y un conjunto* Γ *maximal consistente cualquiera tal que* $\Gamma \vdash \alpha$.
Ccomo Γ *es maximal consistente, se tiene que* $\Gamma \vdash \alpha$ *si y sólo si* $\alpha \in \Gamma$. *Es decir, que un teorema pertenece a todo conjunto maximal consistente.*

$\underline{(\Leftarrow)}$: *Consideremos una fórmula* α *que pertence a todo conjunto maximal consistente y analicemos si se trata de un teorema.*
Si Γ *maximal consistente, como* $\alpha \in \Gamma$, *entonces sabemos que* $\neg\alpha \notin \Gamma$. *Es decir,* $\neg\alpha$ *no perteneces a ningún conjunto maximal consistente.*
Sea Δ *cualquier conjunto tal que* $\neg\alpha \in \Delta$. *Si* Δ *fuera consistente, por lema de Lindenbaum se podría extender a un conjunto maximal consistente, al cual pertenecería* $\neg\alpha$, *cosa que no puede pasar. Por lo que* Δ *es inconsistente. En particular,* $\{\neg\alpha\}$ *es inconsistente.*
Reescribimos $\{\neg\alpha\} = \emptyset \cup \{\neg\alpha\}$ *y decimos que* $\{\neg\alpha\}$ *es inconsistente si y sólo si* $\emptyset \vdash \alpha$.
Como las únicas consecuencias sintácticas del conjunto vacío son los teoremas, entonces α *es necesariamente un teorema.*

b. ***Falso.***

Armamos un contraejemplo partiendo de un conjunto maximal consistente.

A partir de ahí es fácil obtener un conjunto inconsistente: agregandole cualquier fórmula que no perteneciera.

Sea Δ *un conjunto maximal consistente. Como todo conjunto maximal consistente es infinito, podemos tomar* $\alpha, \beta \in \Delta$ *tales que* $\alpha \neq \beta$. *Definimos.*

- $\Gamma_1 = \Delta \cup \{\neg\alpha\}$. *Como* Δ *es maximal consistente y* $\alpha \in \Delta$, *entonces* $\neg\alpha \notin \Delta$. *Entonces, por definición de maximal consistente,* Γ_1 *es inconsistente.*
- $\Gamma_2 = \Delta \cup \{\neg\beta\}$. *Como* Δ *es maximal consistente y* $\beta \in \Delta$, *entonces* $\neg\beta \notin \Delta$. *Entonces, por definición de maximal consistente,* Γ_2 *es inconsistente.*

Si hacemos $\Gamma_1 \cap \Gamma_2 = \Delta$ *que es maximal consistente pero* Γ_1 *y* Γ_2 *no lo son.*

c. ***Verdadero.***

Probamos que Γ' *es consistente, ya que por lema de Lindenbaum, de serlo, se puede extender a un conjunto maximal consistente.*

Si observamos Γ', *se puede notar que todas sus fórmulas pueden derivarse a partir de* Γ *pues son implicaciones donde el antecedente es la negación de una fórmula de* Γ.

Intuitivamente, podemos decir que Γ' *debe ser consistente ya que si permite derivar una contradicción, esta podría derivarse también desde* Γ *que es consistnte. Entonces, demostramos por el absurdo:*

Supongamos que Γ' *no es consistnte. Entonces existe una fórmula* γ *tal que* $\Gamma' \vdash \gamma$ *y* $\Gamma' \vdash \neg\gamma$. *Sea* $\gamma_1, \ldots, \gamma_n$ $(\gamma_n = \gamma)$ *una derivación de* γ *a partir de* Γ'.

A partir de ella construiremos una derivación γ *a partir de* Γ. *Para cada* γ*:*

- *Si* γ_i *es un axioma de SP o se obtiene por MP a partir de fórmulas anteriores, la dejamos como está.*
- *Si* γ_i *es una fórmula de* Γ', *es de la forma* $(\neg\alpha \to \beta)$ *con* $\alpha \in \Gamma$. *Sabemos que* $\{\alpha\} \vdash (\neg\alpha \to \beta)$ *y como* $\{\alpha\} \subseteq \Gamma$ *vale que* $\Gamma \vdash \gamma$ *y, por definición de construcción sintáctica, existe una derivación de* γ $(\tilde{\gamma}_1, \ldots, \tilde{\gamma}_m)$ *a partir* Γ. *Si en la derivación a partir de* Γ' *agregamos antes de* γ *las fórmulas* $\tilde{\gamma}_1, \ldots, \tilde{\gamma}_m$ $(\tilde{\gamma}_m = \gamma_i)$ *seguimos teniendo una derivación válida, reemplazando las fórmulas de* Γ' *por las de* Γ.

De esto, sacamos que $\Gamma \vdash \gamma$, pero de manera análoga podemos construir una derivación de $\neg\gamma$ a partir Γ, obteniendo $\Gamma \vdash \neg\gamma$. Lo cual es absurdo pues Γ es consistente, por lo que Γ' también lo es.

Aplicando el lema de Lindenbaum, Γ' puede extenderse a Δ maximal consistente.

Ejercicio 5.10.
Sea Γ un conjunto de fórmulas del lenguaje $\{\neg, \to\}$. Demostrar los siguientes puntos:

a. Si Γ es un conjunto maximal consistente, entonces $\Gamma \vdash \alpha$ si y sólo si $\alpha \in \Gamma$.

b. Γ es un conjunto maximal consistente si y sólo si sucede simulatáneamente:

1. *Para toda α, o bien $\alpha \in \Gamma$ o bien ('o' exclusivo) $\neg\alpha \in \Gamma$.*
2. *Todos los axiomas de SP están en Γ.*
3. *Γ está cerrado por MP, es decir: si $(\alpha \to \beta) \in \Gamma$ y $\alpha \in \Gamma$ entonces $\beta \in \Gamma$.*

c. Si Γ es maximal consistente y $(\neg\alpha \to \beta) \in \Gamma$, entonces $\alpha \in \Gamma$ ó $\beta \in \Gamma$.

d. Si Γ es maximal consistente entonces para todo par de fórmulas α, β se tiene que $(\alpha \to \beta) \in \Gamma$ o $(\beta \to \alpha) \in \Gamma$.

Solución 5.10.

a. Visto en la teórica.

b. La idea de la demostración será ver ambas implicaciones por separado, es decir, por un lado que dado Γ, un conjunto maximal consistente, cada una de las propiedades se cumplen. Y por otro lado que, todo conjunto Γ que cumple las 3 propiedades, necesariamente es maximal consistente.

$(\Rightarrow)$: *Empecemos con el primer caso y sea Γ,un conjunto maximal consistente.*

(1) Si no se cumple que para toda α, o bien $\alpha \in \Gamma$ o bien $\neg\alpha \in \Gamma$, luego puede pasar:

- *$\alpha \in \Gamma$ y $\neg\alpha \in \Gamma$.*
- *$\alpha \notin \Gamma$ y $\neg\alpha \notin \Gamma$.*

Supongamos que ocurre lo primero. Luego $\Gamma \vdash \alpha$ y $\Gamma \vdash \neg\alpha$. Por lo tanto Γ es inconsistente, lo cual es un absurdo. Por lo tanto, no puede ocurrir que $\alpha \in \Gamma$ y a la vez $\neg\alpha \in \Gamma$.

Supongamos que ocurre lo segundo. Luego, por ser Γ maximal consistente, si $\alpha \notin \Gamma$ y $\neg\alpha \notin \Gamma$ entonces $\Gamma \cup \{\alpha\}$ y $\Gamma \cup \{\neg\alpha\}$ son inconsistentes y, por lo tanto, $\Gamma \vdash \alpha$ y $\Gamma \vdash \neg\alpha$. Luego Γ es inconsistente, lo cual es un absurdo. Por lo tanto, no puede ocurrir que $\alpha \notin \Gamma$ y $\neg\alpha \notin \Gamma$.

Habiendo agotado las opciones, se debe cumplir que, o bien $\alpha \in \Gamma$, o bien $\neg\alpha \in \Gamma$, pero no ambos.

(2) Tomemos α, una instanciación arbitraria de los axiomas de SP. Por ser α un axioma, la siguiente es una derivación de α a partir de Γ de longitud 1:

$1.\alpha$	*(por ser axioma de SP)*

Luego, $\Gamma \vdash \alpha$ y, por el inciso anterior, $\alpha \in \Gamma$.

(3) Dados α y β tales que $(\alpha \to \beta) \in \Gamma$ y $\alpha \in \Gamma$, la siguiente es una derivación de β a partir de Γ.

$1.(\alpha \to \beta)$	*(pues $(\alpha \to \beta) \in \Gamma$)*
$2.\alpha$	*(pues $\alpha \in \Gamma$)*
$3.\beta$	$(MP\ 1\ y\ 2)$

Luego, $\Gamma \vdash \beta$ y, por inciso anterior, $\beta \in \Gamma$.

$(\Leftarrow)$: *Supongamos Γ no maximal consistente.*

Esto ocurre si y sólo si Γ es no consistente o existe una fórmula $\varphi \notin \Gamma$ tal que $\Gamma \cup \{\varphi\}$ es consistente.

Supongamos lo primero: Γ no consistente. Luego, para alguna fórmula α, $\Gamma \vdash \alpha$ y $\Gamma \vdash \neg\alpha$.

Por definición de consecuencia sintáctica, existe una derivación $\alpha_1, \ldots, \alpha_n$ de α donde $\alpha_i \in \Gamma$, o es un axioma de SP, o se obtiene a partir de MP de $1 \leq j, k < i$ para todo $1 \leq i \leq n$.

Entonces existe una derivación $\alpha_1, \ldots, \alpha_n$ de α donde $\alpha_i \in \Gamma$, o se obtiene a partir de MP de $1 \leq j, k < i$ para todo $1 \leq i \leq n$. Ya que si α_i es un axioma, también $\alpha_i \in \Gamma$, por (2).

Entonces existe una derivación $\alpha_1, \ldots, \alpha_n$ de α donde $\alpha_i \in \Gamma$, para todo $1 \leq i \leq n$. Ya que si α_i se obtiene aplicando MP, también $\alpha_i \in \Gamma$ por (3).

Notemos que $\alpha \in \Gamma$ ya que $\alpha_n \in \Gamma$ y $\alpha_n = \alpha$ por ser $\alpha_1, \ldots, \alpha_n$ una derivación de α. El mismo razonamiento se aplica para $\neg\alpha$ y se concluye que $\neg\alpha \in \Gamma$.

Por (1), no puede pasar que $\alpha \in \Gamma$ y $\neg\alpha \in \Gamma$, por lo que llegamos a un absurdo.

Por lo tanto, no puede ser Γ inconsistente. Supongamos ahora que Γ es consistente pero no maximal.

Luego, existe $\alpha \notin \Gamma$ tal que $\Gamma \cup \{\alpha\}$ es consistente y, entonces, $\Gamma \vdash \neg\alpha$. Pero por (1) y como $\alpha \notin \Gamma$ se tiene que $\neg\alpha \in \Gamma$. Es decir, $\Gamma \vdash \neg\alpha$, lo que resulta ser una contradicción.

Por lo tanto, Γ debe ser maximal consistente.

c. *Si $\alpha \in \Gamma$ entonces ya no hay nada que probar. Luego, supongamos que $\alpha \notin \Gamma$ entonces $\neg\alpha \in \Gamma$ puesto que Γ es un conjunto maximal consistente.*

Entonces, como $\neg\alpha \in \Gamma$ y, por hipótesis, $(\neg\alpha \to \beta) \in \Gamma$ usando MP se tiene que necesariamente $\beta \in \Gamma$.

d. *Supongamos que $(\alpha \to \beta) \in \Gamma$, entonces como Γ es maximal consistente*

$$(\alpha \to \beta) \in \Gamma \Leftrightarrow \Gamma \vdash (\alpha \to \beta) \Leftrightarrow \Gamma \cup \{\alpha\} \vdash \beta \Leftrightarrow \Gamma \cup \{\alpha\} \cup \{\neg\beta\} \text{ es inconsistente}$$

Luego, si $\alpha \notin \Gamma$ entonces (sigue valiendo lo anterior) y se deduce que $(\alpha \to \beta) \in \Gamma$.

Usando el mismo argumento,

$$(\beta \to \alpha) \in \Gamma \Leftrightarrow \Gamma \cup \{\beta\} \cup \{\neg\alpha\} \text{ es inconsistente}$$

Luego, si $\alpha \in \Gamma$ entonces (vale lo anterior) y se deduce que $(\beta \to \alpha) \in \Gamma$.

Ejercicio 5.11.
El siguiente procedimiento fue diseñado por Lindenbaum para obtener un conjunto maximal consistente a partir de un conjunto consistente Γ.

1) Enumeramos las fórmulas de nuestro lenguaje $\alpha_1, \alpha_2, \ldots$

2) Definimos la secuencia de conjuntos:

$$\begin{array}{lcl} \Gamma_0 & = & \Gamma \\ \Gamma_{n+1} & = & \begin{cases} \Gamma_n \cup \{\alpha_n\} & \text{si el conjunto es consistente} \\ \Gamma_n \cup \{\neg\alpha_n\} & \text{en otro caso} \end{cases} \\ \Gamma^+ & = & \bigcup_{n \geq 0} \Gamma_n \end{array}$$

Demostrar los siguientes puntos:

a. *Cada Γ_i es consistente.*

b. *Exactamente una de las fórmulas α y $\neg\alpha$ está en Γ^+ para cada fórmula α.*

c. *Todos los teoremas están en Γ^+.*

d. *Γ^+ es un conjunto maximal consistente.*

Solución 5.11.

a. *Usaremos inducción en i.*

C.B. : *Es Trivial puesto que por construcción $\Gamma_0 = \Gamma$ es consistente ya que Γ lo es.*

P.I. : *Supongamos que Γ_i es consistente para todo $i < n$ y supongamos que Γ_n es inconsistente.*
*Como Γ_n es inconsistente entonces existe $\beta \in$ **Form** tal que $\beta \in \Gamma_n$ y $\neg\beta \in \Gamma_n$. Luego, $\beta \in \Gamma_{n-1} \cup \{\alpha\}$ y $\neg\beta \in \Gamma_{n-1} \cup \{\alpha\}$ o bien $\beta \in \Gamma_{n-1} \cup \{\neg\alpha\}$ y $\neg\beta \in \Gamma_{n-1} \cup \{\neg\alpha\}$.*
Puesto que Γ_{n-1} es consistente por hipótesis inductiva, entonces $\beta \in \{\alpha\}$ y $\neg\beta \in \{\alpha\}$ lo que resulta ser una contradicción.
El mismo argumento se usa para ver que si $\beta \in \Gamma_{n-1} \cup \{\neg\alpha\}$ y $\neg\beta \in \Gamma_{n-1} \cup \{\neg\alpha\}$ entonces también se llega a una contradicción.

b. *Sea α una fórmula entonces existe un $i \in \mathbb{N}$ tal que $\alpha_i = \alpha$ y como $\alpha_i \in \Gamma_{i+1}$ con Γ_{i+1} consistente se sigue que para cada fórmula α, exactamente una de las fórmulas α y $\neg\alpha$ está en Γ^+.*

c. *Sea α un teorema, entonces $\alpha \in \Gamma^+$ si y sólo si existe $i \in \mathbb{N}_0$ tal que $\alpha \in \Gamma_i$.*

*d. Es claro que resulta consistente puesto que para $\alpha \in$ **Form** se tiene que $\alpha \in \Gamma^+$ o bien $\neg\alpha \in \Gamma^+$ (por construcción con cada Γ_i consistente).*

Supongamos ahora que no es maximal, por lo tanto, debe existir Δ maximal consistente tal que $\Gamma^+ \subseteq \Delta$ y, por lo tanto, existiría una fórmula α que está en Δ pero no en Γ^+.

Luego, como $\alpha \notin \Gamma^+$ por el inciso (b) se tiene que $\neg\alpha \in \Gamma^+$ y, por lo tanto, como $\Gamma^+ \subseteq \Delta$ se tiene que $\neg\alpha \in \Delta$. Lo que resulta ser una contradicción puesto que $\alpha \in \Delta$ y Δ es consistente.

Luego, Γ^+ es un conjunto maximal consistente.

Ejercicio 5.12.
Demostrar que las siguientes definiciones de compacidad son equivalentes:

1. *Si $\Gamma \models \alpha$ entonces para algún subconjunto finito $\Gamma_0 \subseteq \Gamma$, $\Gamma_0 \models \alpha$.*
2. *Si todo subconjunto finito $\Gamma_0 \subseteq \Gamma$ es satisfacible, Γ es satisfacible.*
3. *Si Γ es insatisfacible, algún subconjunto finito de Γ es insatisfacible.*

Solución 5.12.

(1) $\Rightarrow$ (2) : *Sea $\varphi \in \Gamma$ entonces existe $\Gamma_0 \subseteq \Gamma$ finito satisfacible tal que $\varphi \in \Gamma_0$. Luego existe v valuación tal que $v \models \varphi$ y, por lo tanto, Γ es satisfacible.*

(2) $\Rightarrow$ (1) : *Sea $\alpha \in$ **Form** tal que $\Gamma \models \alpha$. Como todo subconjunto finito $\Gamma_0 \subseteq \Gamma$ es satisfacible, se tiene que existe $\Gamma_0 \subseteq \Gamma$ finito tal que $\Gamma_0 \models \alpha$.*

(2) $\Rightarrow$ (3) : *Sea Γ insatisfacible y supongamos que todo suconjunto finito de Γ es satisfacible. Luego por (2) se tendría que Γ es satisfacible, lo que resultaría ser una contradicción.*

(3) $\Rightarrow$ (2) : *Supongamos que Γ es insatisfacible, luego algún subconjunto finito de Γ es insatisfacible pero por hipótesis todos sus subconjuntos finitos son satisfacibles lo que resultaría ser una contradicción.*

En los siguientes ejercicios se puede asumir que SP es correcto y completo.

Ejercicio 5.13.
*Demostrar que $\Gamma \subseteq$ **Form** es un conjunto maximal consistente si y sólo si para alguna valuación v*

$$\Gamma = \{\alpha \mid v \models \alpha\}$$

Solución 5.13.

($\Rightarrow$) : *Si Γ es maximal consistente, en particular, es consistente. Por lo tanto, es satisfacible, es decir, existe una valuación v tal que $v \models \alpha$ para todo $\alpha \in \Gamma$. Para esta v, se cumple que*

$$\Gamma \subseteq \{\alpha \mid v \models \alpha\}$$

Por otra parte, supongamos que existe una fórmula β tal que $\beta \in \{\alpha \mid v \models \alpha\}$, pero $\beta \notin \Gamma$. Por definición de conjunto maximal consistente, el conjunto $\Gamma \cup \{\beta\}$ debe ser inconsistente. Pero, a su vez,

$$\Gamma \cup \{\beta\} \subseteq \{\alpha \mid v \models \alpha\}$$

que es un conjunto satisfacible (por la valuación v), y por lo tanto, también debe ser consistente.

Esto es absurdo, porque un conjunto consistente no puede tener un subconjunto inconsistente.

Entonces, debe ser que todo $\beta \in \{\alpha \mid v \models \alpha\}$ cumple $\beta \in \Gamma$, y por lo tanto,

$$\Gamma = \{\alpha \mid v \models \alpha\}$$

($\Leftarrow$) : *Sea v cualquier valuación. Consideremos el conjunto*

$$\Gamma = \{\alpha \mid v \models \alpha\}$$

y veamos que es maximal consistente.

Para empezar, Γ es claramente satisfacible ($v \models \Gamma$), así que debe ser también consistente. Para que sea maximal consistente, debe pasar que para cualquier fórmula $\beta \notin \Gamma, \Gamma \cup \{\beta\}$ es inconsistente, o lo que es lo mismo, $\Gamma \vdash \neg\beta$.

Veamos que esto efectivamente es así. Si tomamos β tal que $\beta \notin \Gamma$, por definición de Γ, sabemos que $v \not\models \beta$. Por lo tanto, $v \models \neg\beta$. Esto quiere decir que $\neg\beta \in \Gamma$, y entonces, $\Gamma \vdash \neg\beta$.

Concluimos entonces que Γ debe ser maximal consistente, como queríamos demostrar.

Ejercicio 5.14.
Sea α una fórmula que no es una tautología, y sea Γ el conjunto de todas las instanciaciones de α (por instancia de α nos referimos a reemplazar uniformemente las variables proposicionales de α por fórmulas arbitrarias). Demostrar que Γ es inconsistente.

Solución 5.14.
Como α no es una tautología existe v valuación tal que $v \not\models \alpha$. En particular, habrá valores para los cuales $v(p_i) = 0$ y $v(p_j) = 1$ para $p_i, p_j \in \mathbf{Var}(\alpha)$.

Sean entonces $\varphi, \beta \in \mathbf{Form}$ tales que φ es una contradicción y β una tautología.

Por lo tanto, al realizar la instancia de manera tal que para los $p_i \in \mathbf{Var}(\alpha)$ donde $v(p_i) = 0$ se cambian por φ y para los $p_j \in \mathbf{Var}(\alpha)$ donde $v(p_j) = 1$ se cambian por β se tiene que Γ es inconsistente.

- *Otra manera de resolverlo es plantear inducción estructural usando como caso base que toda $p \in \mathbf{Prop}$ se puede reemplazar por una β que sea una contradicción.*

Ejercicio 5.15.
Sea β una fórmula fija y Γ un conjunto consistente, mostrar que si $\Gamma \nvdash \beta$ y $\Gamma \nvdash \neg\beta$, entonces existen Γ_1 y Γ_2 maximales consistentes tales que $\mathbf{Con}(\Gamma) \subseteq \mathbf{Con}(\Gamma_1)$, $\mathbf{Con}(\Gamma) \subseteq \mathbf{Con}(\Gamma_2)$, y $\Gamma_1 \vdash \beta$ y $\Gamma_2 \vdash \neg\beta$.

Solución 5.15.
Como $\Gamma \nvdash \beta$ y $\Gamma \nvdash \neg\beta$, entonces $\Gamma \cup \{\beta\}$ y $\Gamma \cup \{\neg\beta\}$ son consistentes. Luego, por el lema de Lindenbaum existen Γ_1 y Γ_2 maximales consistentes tales que $\Gamma \subseteq \Gamma \cup \{\beta\} \subseteq \Gamma_1$ y $\Gamma \subseteq \Gamma \cup \{\neg\beta\} \subseteq \Gamma_2$.

Por lo tanto, $\mathbf{Con}(\Gamma) \subseteq \mathbf{Con}(\Gamma_1)$, $\mathbf{Con}(\Gamma) \subseteq \mathbf{Con}(\Gamma_2)$.

Como Γ_1 es maximal consistente entonces $\beta \in \Gamma_1$ o $\neg\beta \in \Gamma_1$, pero por construcción $\beta \in \Gamma_1$ (puesto que si $\neg\beta \in \Gamma_1$ entonces $\beta \in \Gamma_1$ y $\neg\beta \in \Gamma_1$, lo que resultaría ser una contradicción) y, por lo tanto, $\Gamma_1 \vdash \beta$.

Análogamente, como Γ_2 es maximal consistente entonces $\beta \in \Gamma_2$ o $\neg\beta \in \Gamma_2$, pero por construcción $\neg\beta \in \Gamma_2$ (puesto que si $\beta \in \Gamma_2$ entonces $\beta \in \Gamma_2$ y $\neg\beta \in \Gamma_2$, lo que resultaría ser una contradicción) y, por lo tanto, $\Gamma_2 \vdash \neg\beta$.

Ejercicio 5.16.
Sea Σ un conjunto de fórmulas proposicionales. Demostrar que si existe un conjunto finito de fórmulas Γ tal que $\mathbf{Con}(\Sigma) = \mathbf{Con}(\Gamma)$, entonces existe un subconjunto finito $\Sigma_0 \subseteq \Sigma$ tal que $\mathbf{Con}(\Sigma) = \mathbf{Con}(\Sigma_0)$.

Solución 5.16.
Sea $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$. Sabemos que, para todo $1 \leq i \leq n$, $\gamma_i \in \mathbf{Con}(\Gamma) = \mathbf{Con}(\Sigma)$. Es decir, $\Sigma \models \gamma_i$ para todo $i \in \{1, \ldots, n\}$, luego por el Teorema de Compacidad, para cada $i \in \{1, \ldots, n\}$ existe Σ_i finito tal que $\Sigma_i \models \gamma_i$.

Por lo tanto, sea $\Sigma_0 = \bigcup_{i=1}^{n} \Sigma_i$ finito (es unión finita de conjuntos finitos). Además, para todo $1 \leq i \leq n$

$$\gamma_i \in \mathbf{Con}(\Sigma_i) \subseteq \mathbf{Con}(\Sigma_0) \Rightarrow \Gamma \subseteq \mathbf{Con}(\Sigma_0)$$

Luego, como $\Gamma \subseteq \mathbf{Con}(\Sigma_0)$

$$\mathbf{Con}(\Sigma) = \mathbf{Con}(\Gamma) \subseteq \mathbf{Con}(\mathbf{Con}(\Sigma_0)) = \mathbf{Con}(\Sigma_0)$$

y, además,

$$\mathbf{Con}(\Sigma_0) \subseteq \mathbf{Con}(\Sigma) = \mathbf{Con}(\Gamma)$$

Es decir, $\mathbf{Con}(\Sigma) = \mathbf{Con}(\Sigma_0)$.

Ejercicio 5.17.
Demostrar que si Γ es un conjunto maximal consistente entonces $\Gamma = \mathbf{Con}(\Gamma)$.

Solución 5.17.
Sabemos que $\Gamma \subseteq \mathbf{Con}(\Gamma)$. Luego, sea $\alpha \in \mathbf{Con}(\Gamma) \Rightarrow \Gamma \models \alpha \Rightarrow \Gamma \vdash \alpha \underbrace{\Rightarrow}_{\Gamma \text{ m.c.}} \alpha \in \Gamma$.

Ejercicio 5.18.
Dados $\{\Gamma_i\}_{i \in \mathbb{N}}$ tal que $\Gamma_i \subseteq \Gamma_{i+1}$.

a. Si Γ_i es satisfacible. >Es $\Gamma^\infty = \bigcup_{i \in \mathbb{N}} \Gamma_i$ satisfacible?

b. tal que $\Gamma^\infty = \bigcup_{i \in \mathbb{N}} \Gamma_i$ es insatisfacible, entonces existe un $k \in \mathbb{N}$ tal que Γ_k es insatisfacible.

Solución 5.18.

a. *Vamos a probar que todo subconjunto finito es satisfacible entonces por compacidad Γ^∞ será satisfacible.*

Tomemos un conjunto finito $\Delta \subseteq \Gamma^\infty$, como es finito y está incluido en Γ^∞ entonces existe un Γ_k que contiene a todo Δ.

Este Γ_k es satisfacible y como $\Delta \subseteq \Gamma_k$ entonces Δ es satisfacible.

Al haber tomado un Δ genérico probamos que todo subconjunto finito de Γ^∞ es satisfacible entonces por compacidad Γ^∞ es satisfacible.

b. *Usamos la segunda equivalencia del teorema de compacidad. Como Γ^∞ es insatisfacible, existe un conjunto finito $\Gamma_0 = \{\psi_1, \ldots, \psi_n\}$ insatisfacible.*

Como $\Gamma_0 \subseteq \Gamma^\infty$, para cada ψ_i existe un índice $f(i)$ tal que $\psi_i \in \Gamma_{f(i)}$.

Llamo k al máximo entre todos los $f(i)$. Como la sucesión de conjuntos es creciente y $\psi_i \in \Gamma_{f(i)}$ para todo i, se cumple que $\Gamma_0 \subseteq \Gamma_k$.

Entonces Γ_k es un conjunto insatisfacible, porque contiene a un conjunto insatisfacible.

Ejercicio 5.19.
Sean Γ_1 y Γ_2 conjuntos satisfacibles de fórmulas tales que $\Gamma_1 \cup \Gamma_2$ es insatisfacible.

a. *Mostrar que existen fórmulas $\alpha \in \mathbf{Con}(\Gamma_1)$, $\beta \in \mathbf{Con}(\Gamma_2)$ tales que $\alpha \to \neg\beta$ es una tautología.*

b. *Mostrar que existe un φ tal que $\Gamma_1 \models \varphi$ y $\Gamma_2 \models \neg\varphi$.*

Sugerencia: usar el Teorema de Compacidad.

Solución 5.19.

a. *Sabemos que $\Gamma_1 \cup \Gamma_2$ es insatisfacible, entonces por compacidad debe existir un $\Gamma_0 \subseteq \Gamma_1 \cup \Gamma_2$ insatisfacible finito.*

Observemos que Γ_0 no está completamente incluido en Γ_1 ni Γ_2. Si así fuera entonces Γ_1 y Γ_2 serían insatisfacibles ya que contienen un conjunto insatisfacible.

Entonces la relación entre los conjuntos debe ser la siguiente: Γ_0 tiene elementos tanto de Γ_1, como de Γ_2 inclusive algunos que no están en su intersección.

Podemos dividir a Γ_0 como $\Gamma_0 = \{\underbrace{\alpha_1, \ldots, \alpha_n}_{\in\Gamma_1}, \underbrace{\beta_1, \ldots, \beta_m}_{\in\Gamma_2}\}$.

Veamos que $\Gamma_1 \models \alpha_1 \wedge \ldots \wedge \alpha_n$:

Por un lado sabemos que Γ_1 es satisfacible, entonces existe una valuación v tal que $v \models \Gamma_1$. Como para todo $1 \leq i \leq n, \alpha_i \in \Gamma_1$ entonces para toda valuación v tal que $v \models \Gamma_1$ entonces $v \models \alpha_i$ y, por lo tanto, $v \models \alpha_1 \wedge \ldots \wedge \alpha_n$.

En consecuencia, como toda valuación v tal que $v \models \Gamma_1$ es tal que $v \models \alpha_1 \wedge \ldots \wedge \alpha_n$, se tiene que $\Gamma_1 \models \alpha_1 \wedge \ldots \wedge \alpha_n$.

Sea, entonces, $\alpha = \alpha_1 \wedge \ldots \wedge \alpha_n$. Por lo anterior, es claro que $\alpha \in \mathbf{Con}(\Gamma_1)$.

Veamos que $\Gamma_2 \models \beta_1 \wedge \ldots \wedge \beta_n$:

Por un lado sabemos que Γ_2 es satisfacible, entonces existe una valuación w tal que $w \models \Gamma_2$. Como para todo $1 \leq j \leq m, \beta_j \in \Gamma_2$ entonces para toda valuación w tal que $w \models \Gamma_2$ entonces $w \models \beta_j$ y, por lo tanto, $w \models \beta_1 \wedge \ldots \wedge \beta_n$.

En consecuencia, como toda valuación w tal que $w \models \Gamma_2$ es tal que $w \models \beta_1 \wedge \ldots \wedge \beta_n$, se sigue que $\Gamma_2 \models \beta_1 \wedge \ldots \wedge \beta_n$.

Sea, entonces, $\beta = \beta_1 \wedge \ldots \wedge \beta_n$. Por lo anterior, es claro que $\beta \in \mathbf{Con}(\Gamma_2)$.

Como Γ_0 es insatisfacible la fórmula $\alpha \wedge \beta$ es una contradicción pero, esto es equivalente a que

$$\neg(\alpha \wedge \beta) \equiv \neg\alpha \vee \neg\beta \equiv \alpha \to \neg\beta$$

sea una tautología.

b. *Sabemos que $\Gamma_1 \cup \Gamma_2$ es insatisfacible, entonces por compacidad debe existir un $\Gamma_0 \subseteq \Gamma_1 \cup \Gamma_2$ insatisfacible finito.*

Observemos que Γ_0 no está completamente incluido en Γ_1 ni Γ_2. Si así fuera entonces Γ_1 y Γ_2 serían insatisfacibles ya que contienen un conjunto insatisfacible.

Entonces la relación entre los conjuntos debe ser la siguiente, Γ_0 tiene elementos tanto de Γ_1, Γ_2 inclusive algunos que no están en su intersección.

Podemos dividir a Γ_0 como $\Gamma_0 = \{\underbrace{\alpha_1, \ldots, \alpha_n}_{\in\Gamma_1}, \underbrace{\beta_1, \ldots, \beta_m}_{\in\Gamma_2}\}$.

Veamos primero que $\Gamma_1 \models \alpha_1 \wedge \ldots \wedge \alpha_n$:

Por un lado sabemos que Γ_1 es satisfacible, entonces existe una valuación v tal que $v \models \Gamma_1$. Como para todo $1 \leq i \leq n, \alpha_i \in \Gamma_1$ entonces para toda valuación v tal que $v \models \Gamma_1$ entonces $v \models \alpha_i$ y, por lo tanto, $v \models \alpha_1 \wedge \ldots \wedge \alpha_n$.

En consecuencia, como toda valuación v tal que $v \models \Gamma_1$ es tal que $v \models \alpha_1 \wedge \ldots \wedge \alpha_n$, se tiene que $\Gamma_1 \models \alpha_1 \wedge \ldots \wedge \alpha_n$.

Veamos ahora que $\Gamma_2 \models \neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$:

Por definición sabemos que si v es una valuación que satisface Γ_2 entonces $v \models \beta$ para todo $1 \leq j \leq m$ (ya que $\beta_j \in \Gamma_2$). Sabemos también que al menos existe una $v \models \Gamma_2$ porque Γ_2 es satisfacible.

Sea v una valuación que cumple que $v \models \Gamma_2$ luego, como Γ_0 es insatisfacible, debe existir algún α_i tal que $v \not\models \alpha_i$ y, por lo tanto, $v \models \neg\alpha_i$. Luego $v \models \neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$.

En consecuencia, como toda valuación v que $v \models \Gamma_2$ es tal que $v \models \neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$, se tiene que $\Gamma_2 \models \neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$.

Podemos concluir entonces que

$$\Gamma_1 \models \alpha_1 \wedge \ldots \wedge \alpha_n$$
$$\Gamma_2 \models \neg(\alpha_1 \wedge \ldots \wedge \alpha_n)$$

Por lo tanto si tomamos $\varphi = \alpha_1 \wedge \ldots \wedge \alpha_n$ tenemos que $\Gamma_1 \models \varphi$ y $\Gamma_2 \models \neg\varphi$ que era lo pedido.

Ejercicio 5.20.

a. *Sea Γ un conjunto de contingencias tal que para todo par de fórmulas α, β se cumple que $\mathbf{Var}(\alpha) \cap \mathbf{Var}(\beta) = \emptyset$. Probar que Γ es satisfacible.*

b. *Sean $\Gamma \subseteq \mathbf{Form}$ y $\beta \in \mathbf{Form}$ tales que para toda valuación v, existe un $\alpha \in \Gamma$ tal que $v \not\models \alpha \wedge \beta$.*

   Demostrar que existe una cantidad finita de fórmulas $\alpha_1, \ldots, \alpha_n \in \Gamma$ tales que

$$(\beta \to \neg\alpha_1) \vee (\beta \to \neg\alpha_2) \vee \ldots \vee (\beta \to \neg\alpha_n)$$

   es una tautología.

Solución 5.20.

a. *El objetivo es probar que Γ, un conjunto no necesariamente finito, es satisfacible, demostrando que todo subconjunto finito $\Gamma_0 \subseteq \Gamma$ lo es.*

   Dado que tenemos que probar la satisfacibilidad de cualquier subconjunto finito, el problema puede resultar difícil de abordar.

   Lo haremos usaremos una técnica muy útil, que es hacer inducción en el cardinal de los subconjuntos. Es muy importante tener presente que solo podemos hacer esto cuando estamos considerando subconjuntos finitos, cuyos cardinales son números naturales.

   Probaremos, entonces, que para todo $n \in \mathbb{N}$, todo subconjunto $\Gamma_0 \subseteq \Gamma$ con n elementos es satisfacible.

   - *C.B. Si $n = 0$, entonces $\Gamma_0 = \emptyset$ que es trivialmente satisfacible por cualquier valuación.*
   - *P.I. Si Γ_0 tiene $n+1$ elementos, podemos escribir*

$$\Gamma_0 = \Gamma_0' \cup \{\alpha\}$$

     donde Γ_0' tiene n elementos. Como α es una contingencia, existe una valuación v_1 tal que $v_1 \models \alpha$. Además, por hipótesis inductiva, Γ_0' es satisfacible, es decir, existe una valuación v_2 tal que $v_2 \models \Gamma_0'$. Definimos la siguiente valuación:

$$v(p) = \begin{cases} v_1(p) & \text{si } p \in \mathbf{Var}(\alpha) \\ v_2(p) & \text{en otro caso} \end{cases}$$

     Podemos ver que v está bien definida, ya que, por hipótesis, si $p \in \mathbf{Var}(\alpha)$, entonces $p \notin \mathbf{Var}(\beta)$ para cualquier otra fórmula $\beta \in \Gamma$.
     Además, $v(p) = v_1(p)$ para cualquier variable $p \in \mathbf{Var}(\alpha)$, por lo que $v \models \alpha$ si y sólo si $v_1 \models \alpha$.
     Análogamente, para cualquier fórmula $\beta \in \Gamma_0'$, $v \models \beta$ si y sólo si $v_2 \models \beta$.
     Esto nos permite ver que v satisface todas las fórmulas en Γ_0, es decir, $v \vdash \Gamma_0$.
     Por lo tanto, Γ_0 es satisfacible, como queríamos demostrar.

   Esto prueba que todo subconjunto finito de Γ es satisfacible, de donde, aplicando compacidad, se sigue que Γ también es satisfacible.

b. En este caso, nos interesa usar el teorema de compacidad para demostrar la existencia de algún subconjunto finito dentro de otro conjunto. Para estas ocasiones, viene bien tener presente su formulación contrarrecíproca, es decir:

Si un conjunto Δ es insatisfacible, existe algún subconjunto finito $\Delta_0 \subseteq \Delta$ insatisfacible.

Para poder usar el teorema, en primer lugar, debemos tener un conjunto insatisfacible. Intentemos construir, a partir de Γ, un conjunto Δ que sea insatisfacible. Esto quiere decir que, para toda valuación v, existe alguna fórmula $\delta \in \Delta$ tal que $v \not\models \delta$. Esto es exactamente lo que cumplen las fórmulas del tipo $\alpha \wedge \beta$, donde α es alguna de las fórmulas de Γ.

Si definimos, entonces,

$$\Delta = \{\alpha \wedge \beta : \alpha \in \Gamma\}$$

podemos estar seguros de que, para cada valuación v, habrá una fórmula en Δ que no ser á verdadera. Es decir, Δ es un conjunto insatisfacible.

Aplicando ahora el teorema de compacidad, sabemos que existe un subconjunto finito $\Delta_0 \subseteq \Delta$ finito e insatisfacible. Este conjunto es de la forma

$$\Delta_0 = \{(\alpha_1 \wedge \beta), (\alpha_2 \wedge \beta), \ldots, (\alpha_n \wedge \beta)\}$$

y, como es insatisfacible, es falso que todas sus fórmulas sean verdaderas al mismo tiempo. Es decir, la fórmula

$$\neg((\alpha_1 \wedge \beta) \wedge (\alpha_2 \wedge \beta) \wedge \ldots \wedge (\alpha_n \wedge \beta)),$$

que equivale a

$$(\neg(\alpha_1 \wedge \beta) \vee (\neg(\alpha_2 \wedge \beta)) \vee \ldots \vee (\neg(\alpha_n \wedge \beta)),$$

es una tautología.

Solo queda recordar que $(\alpha \wedge \beta)$ es una abreviatura para $\neg(\alpha \to \neg\beta)$, y que a su vez, esta fórmula equivale a $\neg(\beta \to \neg\alpha)$. Haciendo los reemplazos correspondientes en la tautología anterior, vemos que equivale a

$$(\beta \to \neg\alpha_1) \vee (\beta \to \neg\alpha_2) \vee \ldots \vee (\beta \to \neg\alpha_n)$$

Es decir, esta última forma es una tautología, como queríamos demostrar.

Ejercicio 5.21.
Sea $\Gamma = \{\gamma_n\}_{n\in\mathbb{N}}$ un conjunto infinito satisfacible de fórmulas proposicionales, tal que

$$\Delta = \{\gamma_1 \wedge p_1, \gamma_2 \wedge p_2, \ldots\} = \{\gamma_n \wedge p_n\}_{n\in\mathbb{N}}, \text{ es insatisfacible.}$$

Demostrar que existe $k \in \mathbb{N}$ tal que

$$((p_1 \wedge p_2 \ldots \wedge p_k) \to (\neg\gamma_1 \vee \neg\gamma_2 \vee \ldots \neg\gamma_k))$$

es una tautología.

Solución 5.21.
Supongamos que, para todo $k \in \mathbb{N}$, $((p_1 \wedge p_2 \ldots \wedge p_k) \to (\neg\gamma_1 \vee \neg\gamma_2 \vee \ldots \neg\gamma_k))$ no es una tautología. Entonces existe una valuación v_k tal que

$$v_k((p_1 \wedge p_2 \ldots \wedge p_k) \to (\neg\gamma_1 \vee \neg\gamma_2 \vee \ldots \neg\gamma_k)) = 0$$

por lo tanto,

$$v_k(p_1 \wedge p_2 \ldots \wedge p_k) = 1 \text{ y } v_k(\neg\gamma_1 \vee \neg\gamma_2 \vee \ldots \neg\gamma_k) = 0$$

entonces $v_k(p_i) = 1$ y $v_k(\neg\gamma_i) = 0$ para todo $i \in \{1, \ldots, k\}$. Es decir, $v_k(p_i) = v_k(\gamma_i) = 1$ para todo $i \in \{1, \ldots, k\}$. Luego, para todo $k \in \mathbb{N}$ el conjunto

$$\{\gamma_1 \wedge p_1, \gamma_2 \wedge p_2, \ldots, \gamma_k \wedge p_k\} \text{ es satisfacible (por } v_k)$$

Entonces, por el Teorema de Compacidad, el conjunto Δ resultaría satisfacible. Lo que resulta ser una contradicción.

Ejercicio 5.22.
Sea Γ un conjunto insatisfacible de fórmulas proposicionales tal que para todo para todo par de fórmulas $\alpha, \beta \in \Gamma$ vale que si $\{\alpha, \beta\} \vdash \delta$, entonces $\delta \in \Gamma$.

Demostrar que existe una fórmula $\gamma \in \Gamma$ que es una contradicción.

Solución 5.22.
Como Γ es insatisfacible, por el Teorema de Compacidad, existe un subconjunto $\{\gamma_1, \gamma_2, \ldots, \gamma_n\} = \Gamma_0 \subseteq \Gamma$ finito insatisfacible.

Luego, $\gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n$ es una contradicción. Veamos que esta fórmula pertenece a Γ.

$$\{\gamma_1, \gamma_2\} \models \gamma_1 \wedge \gamma_2 \underbrace{\Rightarrow}_{\text{por hipótesis}} \gamma_1 \wedge \gamma_2 \in \Gamma$$

$$\{\gamma_1 \wedge \gamma_2, \gamma_3\} \models \gamma_1 \wedge \gamma_2 \wedge \gamma_3 \underbrace{\Rightarrow}_{\text{por hipótesis}} \gamma_1 \wedge \gamma_2 \wedge \gamma_3 \in \Gamma$$

$$\vdots$$

de manera inductiva

$$\vdots$$

$$\{\gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_{n-1}, \gamma_n\} \models \gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n \underbrace{\Rightarrow}_{\text{por hipótesis}} \gamma_1 \wedge \gamma_2 \wedge \ldots \wedge \gamma_n \in \Gamma$$

Ejercicio 5.23.

1. *Sea Γ un conjunto de fórmulas tal que cada valuación satisface al menos una fórmula de Γ. Probar que existe un número finito de fórmulas $\alpha_1 \ldots \alpha_n \in \Gamma$ tales que $\alpha_1 \vee \ldots \vee \alpha_n$ es tautología.*

2. *Sean $\Gamma_1, \Gamma_2 \subseteq$ **Form** conjuntos de fórmulas de la lógica proposicional con la siguiente propiedad:*

$$\text{para cada valuación } v \text{ existen fórmulas } \alpha \in \Gamma_1 \text{ y } \beta \in \Gamma_2 \text{ tales que } v \models \alpha \vee \beta$$

Demostrar que existe un número finito de fórmulas $\gamma_1, \gamma_2, \ldots, \gamma_n \in \Gamma_1 \cup \Gamma_2$ tales que $\gamma_1 \vee \gamma_2 \vee \ldots \vee \gamma_n$ es una tautología.

Solución 5.23.

1. *Sea $\Delta = \{\neg\alpha : \alpha \in \Gamma\}$.*

   Sea v una valuación y $\alpha \in \Gamma$ la fórmula existente por hipótesis tal que $v \models \alpha$.

   Por semántica del $\neg$, se tiene que $v \not\models \neg\alpha$ y, por lo tanto, $v \not\models \Delta$.

   Como esto vale para cualquier valuación v, Δ es insatisfacible. Por compacidad, hay un subconjunto insatisfacible finito

$$\{\neg\alpha_1, \neg\alpha_2, \ldots, \neg\alpha_n\} \subseteq \Delta$$

   Entonces, para cualquier valuación w,

$$w \not\models \{\neg\alpha_1, \neg\alpha_2, \ldots, \neg\alpha_n\} \Rightarrow w \not\models \neg\alpha_1 \wedge \neg\alpha_2 \wedge \ldots \wedge \neg\alpha_n$$

   y, luego,

$$w \models \neg(\neg\alpha_1 \wedge \neg\alpha_2 \wedge \ldots \wedge \neg\alpha_n)$$

   Como esto vale para cualquier valuación w,

$$\neg(\neg\alpha_1 \wedge \neg\alpha_2 \wedge \ldots \wedge \neg\alpha_n)$$

   es una tautología, y por leyes de de Morgan es equivalente a

$$\alpha_1 \vee \ldots \vee \alpha_n$$

   que es una disyunción de fórmulas contenidas en Γ, como pide el enunciado.

2. *Sea $\Delta = \{\neg(\alpha \vee \beta) : \alpha \in \Gamma_1 \text{ y } \beta \in \Gamma_2\}$.*

   Sea v una valuación y $\alpha \in \Gamma_1$ y $\beta \in \Gamma_2$ las fórmulas existentes por hipótesis tal que $v \models \alpha \vee \beta$.

   Por semántica del $\neg$, se tiene que $v \not\models \neg(\alpha \vee \beta)$ y, por lo tanto, $v \not\models \Delta$.

   Como esto vale para cualquier valuación v, Δ es insatisfacible. Por compacidad, hay un subconjunto insatisfacible finito

$$\{\neg(\alpha_1 \vee \beta_1), \neg(\alpha_2 \vee \beta_2), \ldots, \neg(\alpha_k \vee \beta_k)\} \subseteq \Delta$$

Entonces, para cualquier valuación w,

$$w \not\models \{\neg(\alpha_1 \vee \beta_1), \neg(\alpha_2 \vee \beta_2), \ldots, \neg(\alpha_k \vee \beta_k)\}$$

$$w \not\models \neg(\alpha_1 \vee \beta_1) \wedge \neg(\alpha_2 \vee \beta_2) \wedge \ldots \wedge \neg(\alpha_k \vee \beta_k)$$

y, luego,

$$w \models \neg(\neg(\alpha_1 \vee \beta_1) \wedge \neg(\alpha_2 \vee \beta_2) \wedge \ldots \wedge \neg(\alpha_k \vee \beta_k))$$

Como esto vale para cualquier valuación w,

$$\neg(\neg(\alpha_1 \vee \beta_1) \wedge \neg(\alpha_2 \vee \beta_2) \wedge \ldots \wedge \neg(\alpha_k \vee \beta_k))$$

es una tautología, y por leyes de de Morgan es equivalente a

$$\alpha_1 \vee \beta_1 \vee \alpha_2 \vee \beta_2 \vee \ldots \vee \alpha_k \vee \beta_k$$

que es una disyunción de fórmulas contenidas en $\Gamma_1 \cup \Gamma_1$, como pide el enunciado.

Ejercicio 5.24.
Sea Γ un conjunto de fórmulas que verifica la siguiente propiedad: si $\alpha, \beta \in \Gamma$, entonces $\alpha \to \beta$ es tautología ó $\beta \to \alpha$ es tautología. Probar que si $\Gamma \models \gamma$, entonces existe $\delta \in \Gamma$ tal que $\{\delta\} \models \gamma$.

Solución 5.24.
Por el Teorema de Compacidad, sabemos que si una fórmula es consecuencia semántica de un conjunto de fórmulas entonces es consecuencia de un conjunto finito (que en este caso llamaremos $\Gamma_0 = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$).

Si Γ_0 tiene sólo un elemento, ya no hay nada que probar. Sea $\Gamma_0 = \{\gamma_1, \gamma_2\}$ tal que $\Gamma \vdash \gamma_1$ y $\gamma_1 \to \gamma_2$ es tautología, entonces $\{\gamma_1\} \vdash \gamma$ (si suponemos que $\Gamma \vdash \gamma_2$ y $\gamma_2 \to \gamma_1$ es tautología, entonces $\{\gamma_2\} \vdash \gamma$)

Siguiendo este razonamiento si $\Gamma_0 = \{\gamma_1, \gamma_2, \gamma_3\}$ suponiendo que $\gamma_i, \gamma_j \in \Gamma_0$ entonces $\gamma_i \to \gamma_j$ es tautología ó $\gamma_j \to \gamma_i$ es tautología, al separar en los tres casos posibles se llega a que a partir de alguna de esas tres se puede deducir γ (la idea de fondo es que como las tres fórmulas son comparables entre sí se puede encontrar una que representará a "la más chica" que servirá para deducir las otras).

Haciendo inducción en n, se tiene que el caso base ya está probado.

Supongamos ahora que para todo conjunto $\Gamma_0' = \{\gamma_1, \gamma_2, \ldots, \gamma_j\}$ con $j < n$ vale que si $\Gamma_0' \models \gamma$, entonces existe $\delta \in \Gamma_0'$ tal que $\{\delta\} \models \gamma$ y sea Γ_0 un conjunto de n elementos.

Luego, $\Gamma_0 = \Delta \cup \{\varphi\}$ y sean $\alpha, \beta \in \Gamma_0$ tal que $\Gamma_0 \vdash \delta$. Si $\alpha, \beta \in \Delta$ entonces por hipótesis inductiva sabemos que existe $\gamma \in \Delta \subset \Gamma_0$ tal que $\{\gamma\} \vdash \delta$ como queríamos probar. Por otra parte si $\alpha \in \Delta$ y $\beta \in \{\varphi\}$ como $\alpha \to \beta$ es una tautología entonces como $\Gamma \vdash \alpha$ (es análogo si suponemos que $\beta \to \alpha$ es una tautología pero en ese caso se usaría que $\Gamma \vdash \beta$) entonces $\{\alpha\} \vdash \delta$.

Ejercicio 5.25.
Decidir si la siguiente afirmación es verdadera o falsa y justificar: Si Γ_1 y Γ_2 son consistentes, entonces o bien a partir de $\Gamma_1 \cup \Gamma_2$ se demuestra una contradicción o bien existe Δ maximal consistente tal que $\Gamma_1 \subseteq \Delta$ y $\Gamma_2 \subseteq \Delta$.

Solución 5.25.
Sabemos que $\Gamma_1 \cup \Gamma_2$ puede ser consistente o bien inconsistente.

Si a partir de $\Gamma_1 \cup \Gamma_2$ de deduce una contradicción es porque resultó ser inconsistente.

Si no fue el caso anterior, entonces $\Gamma_1 \cup \Gamma_2$ es consistente y por el lema de Lindenbaum existe Δ maximal consistente tal que $\Gamma_1, \Gamma_2 \subseteq \Gamma_1 \cup \Gamma_2 \subseteq \Delta$.

Práctica 6

Lógica de Primer Orden

Ejercicio 6.1.
Decidir si las siguientes interpretaciones son apropiadas para los siguientes lenguajes, en donde f es un símbolo unario y g es binario:

1. $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{P, =\}, U_A = \mathbb{Q}, c_A = \dfrac{2}{3}, f_A(a) = \dfrac{a}{a^2+1}, g_A(a,b) = a + b$

   $$a\ P_A\ b \text{ si y sólo si } a \leq b.$$

2. $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{P, =\}, U_B = \text{conjunto de todas las listas de enteros}, c_B = [1,2,3,4,5,6,7]$,
   $f_B(I) = \text{quedarse únicamente con las posiciones pares de } I, g_B(I_1, I_2) = \text{concatenar } I_1 \text{ con } I_2$

   $$I_1\ P_B\ I_2 \text{ si y sólo si la long. de } I_1 \text{ es menor o igual que la long. de } I_2.$$

3. $\mathcal{C} = \emptyset, \mathcal{F} = \{f\}, \mathcal{P} = \{=\}, U_C = \text{conjunto de todas las listas de enteros}, h_C(I) = \text{longitud de } I.$

4. $\mathcal{C} = \emptyset, \mathcal{F} = \{f, g\}, \mathcal{P} = \{=\}, U_I = \mathbb{N}, f_I(n) = \sqrt{n}$ y $g_I(n,m) = n + m.$

5. $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{=\}, U_I = \mathbb{N}, f_I(n) = n^2, g_I(n,m) = n + m$ y $c_I = 2.$

6. $\mathcal{C} = \{c, d\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{=\}, U_I = \mathbb{N}, g_I(n,n) = n^2 - n, c_I = d_I = 0$ y $f_I(n) = \begin{cases} 1 & \text{si } n \text{ es primo} \\ 2 & \text{si } n \text{ no es primo} \end{cases}.$

Solución 6.1.
Debemos chequear:

- *El universo U es no vacío.*
- *la interpretación de cada constante pertenece al universo U elegido.*
- *la interpretación de cada función es total y tiene como imagen a elementos del universo U.*
- *la interpretación de cada predicado es total.*

Las únicas interpretaciones que no son válidas son:

4. *La imagen de la función f_I puede no estar en $\mathbb{N}$.*
6. *La función g no es total.*

Ejercicio 6.2.
Decidir si las siguientes fórmulas son

(i) universalmente válidas

(ii) válidas en alguna interpretación del ejercicio anterior

(iii) satisfacibles

(iv) insatisfacibles

1. $\exists x(f(x) = x).$

2. $\forall x(P(x, f(x)))$.

3. $P(x, f(y))$.

Solución 6.2.

1. *Veamos que no es universalmente válida. Sea I una interpretación donde el universo son los naturales y $f_I(n) = n + 1$.*

   Sea v una valuación cualquiera.

   $I, v \models \exists x(f(x) = x)$
   si y sólo si hay algún valor n en U_I tal que $I, v[x \mapsto n] \models f(x) = x$
   si y sólo si hay algún valor n en U_I tal que $v[x \mapsto n](f(x)) \models_I v[x \mapsto n](x)$
   si y sólo si hay algún valor n en U_I tal que $f_I([x \mapsto n](x)) \models_I v[x \mapsto n](x)$
   si y sólo si hay algún valor n en U_I tal que $f_I(n) =_I n$

   Veamos ahora si la fórmula vale en la estructura $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{P, =\}, U_A = \mathbb{Q}, c_A = \frac{2}{3}$, $f_A(a) = \frac{a}{a^2+1}, g_A(a, b) = a + b$

   $a\ P_A\ b$ *si y sólo si* $a \leq b$.

   Sea v una valuación cualquiera: $A, v \models \exists x(f(x) = x)$

   si y sólo si hay algún valor a en U_A tal que $A, v[x \mapsto a] \models f(x) = x$
   si y sólo si hay algún valor a en U_A tal que $v[x \mapsto a](f(x)) =_A v[x \mapsto a](x)$
   si y sólo si hay algún valor a en U_A tal que $f_A(v[x \mapsto a](x)) =_A v[x \mapsto a](x)$
   si y sólo si hay algún valor a en U_A tal que $f_A(a) =_A a$
   si y sólo si hay algún valor a en U_A tal que $\frac{a}{a^2+1} =_A a$

   Veamos ahora si la fórmula vale en la estructura $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{P, =\}$,
   $U_B =$ *conjunto de todas las listas de enteros*, $c_B = [1, 2, 3, 4, 5, 6, 7]$,
   $f_B(I) =$ *quedarse únicamente con las posiciones partes de* $I, g_B(I_1, I_2) =$ *concatenar* I_1 *con* I_2

   $I_1\ P_B\ I_2$ *si y sólo si la long. de* I_1 *es menor o igual que la long. de* I_2.

   $B, v \models \exists x(f(x) = x)$
   si y sólo si hay alúb valor I en U_B tal que $B, v[x \mapsto I] \models f(x) = x$
   si y sólo si hay alúb valor I en U_B tal que $v[x \mapsto I](f(x)) =_B v[x \mapsto I](x)$
   si y sólo si hay alúb valor I en U_B tal que $f_B(v[x \mapsto I](x)) =_B v[x \mapsto I](x)$
   si y sólo si hay alúb valor I en U_B tal que $f_B(I) =_B I$

2. $\forall x(P(x, f(x)))$, $>$ *es universalmente válida?*

   Veamos ahora si la fórmula vale en la estructura $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{P, =\}, U_A = \mathbb{Q}, c_A = \frac{2}{3}$, $f_A(a) = \frac{a}{a^2+1}, g_A(a, b) = a + b$

   $a\ P_A\ b$ *si y sólo si* $a \leq b$.

   Sea v una valuación cualquiera: $A, v \models \forall x(P(x, f(x)))$

   si y sólo si todo valor a en U_A cumple $A, v[x \mapsto a] \models (P(x, f(x)))$
   si y sólo si todo valor valor a en U_A cumple $v[x \mapsto a]P_A v[x \mapsto a](f(x))$
   si y sólo si todo valor valor a en U_A cumple $aP_A f_A(v[x \mapsto a](x))$
   si y sólo si todo valor valor a en U_A cumple $aP_A f_A(a)$
   si y sólo si todo valor valor a en U_A cumple $a \leq \frac{a}{a^2+1}$

Veamos ahora si la fórmula vale en la estructura $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{P, =\}$,
$U_B =$ *conjunto de todas las listas de enteros,* $c_B = [1, 2, 3, 4, 5, 6, 7]$,
$f_B(I) =$ *quedarse únicamente con las posiciones partes de* $I, g_B(I_1, I_2) =$ *concatenar* I_1 *con* I_2

$$I_1 \ P_B \ I_2 \text{ si y sólo si la long. de } I_1 \text{ es menor o igual que la long. de } I_2.$$

$$\begin{aligned} B, v &\models \forall x(P(x, f(x))) \\ &\text{si y sólo si todo valor valor } I \text{ en } U_B \text{ cumple } B, v[x \mapsto I] \models (P(x, f(x))) \\ &\text{si y sólo si todo valor valor } I \text{ en } U_B \text{ cumple } v[x \mapsto I] P_B v[x \mapsto I](f(x)) \\ &\text{si y sólo si todo valor valor } I \text{ en } U_B \text{ cumple } I P_B f_B(v[x \mapsto I](x)) \\ &\text{si y sólo si todo valor valor } I \text{ en } U_B \text{ cumple } I P_B f_B(I) \\ &\text{si y sólo si todo valor valor } I \text{ en } U_B \text{ cumple } |I| \leq |\text{posiciones pares de } I| \end{aligned}$$

Veamos qué sucede en una interpretación I *tal que su universo son los naturales,* P_A *es la operación de menor o igual y* f_A *es la identidad.*

$$\begin{aligned} I, v &\models \forall x(P(x, f(x))) \\ &\text{si y sólo si todo valor valor } n \text{ en } U_I \text{ cumple } I, v[x \mapsto n] \models (P(x, f(x))) \\ &\text{si y sólo si todo valor valor } n \text{ en } U_I \text{ cumple } v[x \mapsto n] P_I v[x \mapsto n](f(x)) \\ &\text{si y sólo si todo valor valor } n \text{ en } U_I \text{ cumple } n P_I f_I(v[x \mapsto n](x)) \\ &\text{si y sólo si todo valor valor } n \text{ en } U_I \text{ cumple } n P_I f_I(n) \\ &\text{si y sólo si todo valor valor } n \text{ en } U_I \text{ cumple } n \leq n \end{aligned}$$

3. *Veamos ahora si la fórmula vale en la estructura* $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{P, =\}, U_A = \mathbb{Q}, c_A = \dfrac{2}{3}$,
$f_A(a) = \dfrac{a}{a^2 + 1}, g_A(a, b) = a + b$

$$a \ P_A \ b \text{ si y sólo si } a \leq b.$$

Sea v_1 *una valuación tal que* $v_1(x) = 10$ *y* $v_1(y) = 0$.

$$\begin{aligned} A, v_1 &\models (P(x, f(y))) \\ &\text{si y sólo si } v_1(x) P_A v_1(f(y)) \\ &\text{si y sólo si } 10 P_A f_A(v_1(y)) \\ &\text{si y sólo si } 10 P_A \frac{0}{0^2 + 1} \\ &\text{si y sólo si } 10 \leq 0 \end{aligned}$$

>*Es satisfacible en esta interpretación? Sea* v_2 *una valuación tal que* $v_2(x) = 0$ *y* $v_2(y) = 1$.

$$\begin{aligned} A, v_2 &\models (P(x, f(y))) \\ &\text{si y sólo si } v_2(x) P_A v_2(f(y)) \\ &\text{si y sólo si } 0 P_A f_A(v_2(y)) \\ &\text{si y sólo si } 0 P_A \frac{1}{1^2 + 1} \\ &\text{si y sólo si } 0 \leq \frac{1}{2} \end{aligned}$$

Análogamente con la interpretación $\mathcal{C} = \{c\}, \mathcal{F} = \{f, g\}, \mathcal{P} = \{P, =\}$,
$U_B =$ *conjunto de todas las listas de enteros,* $c_B = [1, 2, 3, 4, 5, 6, 7]$,
$f_B(I) =$ *quedarse únicamente con las posiciones partes de* $I, g_B(I_1, I_2) =$ *concatenar* I_1 *con* I_2

$$I_1 \ P_B \ I_2 \text{ si y sólo si la long. de } I_1 \text{ es menor o igual que la long. de } I_2.$$

Sea v_3 una valuación tal que $v_3(x) = [3,1]$ y $v_3(y) = []$

$$\begin{aligned} B, v_3 &\models (P(x, f(y))) \\ &\text{si y sólo si } v_(x) P_B v_3(f(y)) \\ &\text{si y sólo si } [3,1] P_B f_B(v_3(y)) \\ &\text{si y sólo si } [3,1] P_B f_B([]) \\ &\text{si y sólo si } |[3,1]| \leq |[]| \\ &\text{si y sólo si } 2 \leq 0 \end{aligned}$$

Ejercicio 6.3.
En cada uno de los siguientes ejemplos, describir la propiedad que determinan los siguientes enunciados.

1. $\forall x \forall y(P(x,y) \rightarrow \exists z((Q(z) \wedge P(x,z)) \vee P(z,y)))$, *donde P y Q son símbolos de predicados binario y unario respectivamente, el universo de la interpretación son los números reales, $P_I =<$, $Q_I(x)$ significa x es un número racional.*

2. $\forall x(Q(x) \rightarrow \exists y(R(y) \wedge P(y,x)))$, *donde P es un símbolo de predicado binario, Q y R son símbolos de predicados unarios, el universo de la interpretación es el conjunto de los días y las personas, $P_I(x,y)$ significa x nace en el día y, $Q_I(x)$ significa x es un día, y $R_I(x)$ significa x es un esclavo.*

3. $\forall x \forall y((Q(x) \wedge Q(y)) \rightarrow P(f(x,y)))$, *donde Q y P son símbolos de predicados unarios, f es un símbolo de función binario, el universo de la interpretación son los números enteros, $Q_I(x)$ significa x es par, $P_I(x)$ significa x es impar, y $f_I(x,y) = x + y$.*

Solución 6.3.

1. *para todo par de números x, y reales existe z racional tal que $x < z < q$.*

2. *todos los días nace un esclavo.*

3. *la suma de dos números pares enteros, es impar.*

Ejercicio 6.4.
Usando como lenguaje el que contiene únicamente la igualdad, escribir enunciados que expresen:

a. *Existen al menos dos elementos.*

b. *Existen exactamente dos elementos.*

c. *Existen a lo sumo dos elementos.*

Agregando al lenguaje un símbolo de predicado unario P, escribir:

d. *Existen a lo sumo dos elementos y al menos uno que cumplen la propiedad P.*

e. *Si existe un elemento que cumple la propiedad P, es único.*

f. *Existe un elemento que cumple la propiedad P y es único.*

Solución 6.4.

a. $(\exists x, y)(x \neq y)$.

b. $(\exists x, y)((x \neq y) \wedge (\forall z)((z = x) \vee (z = y)))$.

c. *la idea será esribir que o no existe ninguno o existe sólo uno o existen exactamente dos,*

$$(\neg(\exists x)(x = x)) \vee ((\exists x)(\forall z)(z = x)) \vee ((\exists x, y)(x \neq y) \wedge (\forall z)((z = x) \vee (z = y)))$$

d. $((\neg(\exists x)(x = x)) \vee ((\exists x)(\forall z)(z = x)) \vee ((\exists x, y)(x \neq y) \wedge (\forall z)((z = x) \vee (z = y)))) \wedge (\exists t)(P(t))$.

e. $(\exists x)(P(x) \rightarrow (\forall y)(x = y))$.

f. $(\exists x)(P(x) \wedge (\forall y)(x = y))$.

Ejercicio 6.5.
Considerar un lenguaje con un símbolo de función f unario. Escribir una fórmula φ que cumpla $\mathcal{A} \models \varphi$ si y sólo si $f_{\mathcal{A}}$ es inyectiva pero no sobreyectiva. >Es φ satisfacible? >Es satisfacible por un modelo finito?

Solución 6.5.

$$\varphi = \underbrace{((\forall x, y)(f_A(x) = f_A(y) \to x = y)}_{\varphi_1} \wedge \underbrace{((\exists y)(\forall x)(f_A(x) \neq y))}_{\varphi_2}$$

Luego, φ expresa que f es inyectiva pero no sobreyectiva para cualquier modelo $\mathcal{M}$ con un símbolo de función f unario. Es decir,

$$\mathcal{M} \models \varphi \Leftrightarrow f_{\mathcal{M}} \quad (\varphi \text{ es satisfacible}).$$

No es satisfacible por un modelo finito puesto que en ese caso una función inyectiva debe ser también sobreyectiva.

Ejercicio 6.6.
Sea $\mathcal{L} = \{=, f, R\}$ un lenguaje con igualdad, una función binaria f y un predicado binario R.

a. *Demostrar que la relación ternaria $P = \{f(x, y, z) : x = y \cap z\}$ es expresable en la interpretación*

$$\mathcal{M} = \langle \mathcal{P}(\mathbb{N}), =, \cup, \subseteq \rangle.$$

Es decir, demostrar que existe una fórmula $\varphi(x, y, z)$ tal que

$$\mathcal{M} \models \varphi(x, y, z)[v] \text{ si y sólo si } v(x) = v(y) \cap v(z).$$

b. *Demostrar que es definible la clase de modelos $\mathcal{M}$ en los cuales los elementos relacionados con x por $R_{\mathcal{M}}$ son aquellos pertenecientes a la imagen de la función $y \mapsto f_{\mathcal{M}}(x, y)$. Es decir, demostrar que existe una sentencia φ tal que $M \models [v]$ si y sólo si $\mathcal{M}$ tiene la propiedad mencionada.*

c. *Demostrar que en la interpretación $\mathcal{M} = \langle \mathbb{Z}, =, \cdot, < \rangle$, todos los elementos del universo son distinguibles. Es decir, demostrar que existen fórmulas*

$$\ldots \varphi_{-2}(x), \varphi_{-1}(x), \varphi_0(x), \varphi_1(x), \varphi_2(x), \ldots$$

tal que $\mathcal{M} \models \varphi_z(x)[v]$ si y sólo si $v(x) = z$.

Solución 6.6.

a. *Un error típico es intentar $\varphi(x, y, z) = R(x, y) \wedge R(x, z)$. Es decir, decir que x es subconjunto de y y de z. Eso no alcanza, porque podría haber muchos, por ejemplo, $x = \emptyset$. Una opción que funciona es*

$$\varphi(x, y, z) = R(x, y) \wedge R(x, z) \wedge (\forall w)(R(w, y) \wedge R(w, z) \to R(w, x))$$

que es lo mismo, pero estamos diciendo que x es el más grande de todos los subconjuntos posibles de y y z. Dicha definición, sí es una definición correcta de la intersección.

b. $\varphi = ((\forall x, y)R(x, y) \to (\exists w)f(x, w) = y) \wedge ((\forall x, y)(\exists w)(f(x, w) = y \to R(x, y)))$.

Notar que funciona porque en la primer implicación estamos viendo que si un elemento está relacionado con x, está en la imagen de la función unaria resultante de fijar x como primer parámetro, y en la segunda estamos viendo que estar en la imagen implica estar relacionado.

Cada implicación significa una inclusión de conjuntos, por lo cual ambas significan la igualdad de los conjuntos involucrados (los elementos relacionados con x y los que estan en la imagen de $f(x, \cdot)$).

c. *Usando la igualdad y la relación menor podemos decir cosas como 'el mínimo' y 'el máximo'. Como en los enteros no hay mínimos ni máximos, precisamos un punto de partida, para lo cual podemos usar el producto.*

Aquí elegí el 0 como punto de partida para que todo sea simétrico.

$$\varphi_0(x) = \forall y,\ f(x, y) = x$$

Es claro que el 0 es el único entero tal que multiplicado por cualquier número da sí mismo.

Ahora, veamos el tema de los mínimos. Podemos decir '1 es el mínimo mayor que 0':

$$\varphi_1(x) = \exists y, \varphi_0(y) \wedge y < x\ \forall z, y < z \to x < z \vee x = z$$

Aquí y toma necesariamente el valor 0 por definición de φ_0. Además, $y < x$ asegura que estamos hablando de un positivo. La última condición asegura que cualquier positivo es igual o mayor a x, cosa que sólo vale para $x = 1$.

Ahora podemos repetir la fórmula para el 2 cambiando 0 por 1, es decir '2 es el mínimo mayor que 1':

$$\varphi_2(x) = \exists y, \varphi_1(y) \wedge y < x\ \forall z, y < z \to x < z \vee x = z$$

Y en general funciona para $i > 0$, 'i es el mínimo mayor que $i - 1$'.

$$\varphi_i(x) = \exists y, \varphi_{i-1}(y) \wedge y < x\ \forall z, y < z \to x < z \vee x = z$$

Así mostramos que 0 y todos los positivos son distinguibles. Ver que los negativos son distinguibles se hace de fórma análoga. Para cada $i < 0$, 'i es el máximo menor que $i + 1$'.

Ejercicio 6.7.
Sea P un símbolo de relación unario y sea f un símbolo de función binario. Para cada una de las fórmulas

$$\forall x \forall y f(x,y) = x, \exists x \forall y f(x,y) = y \text{ y } \exists x (P(x) \wedge \forall y P(f(x,y)))$$

hallar una interpretación que la satisfaga y otra que no la satisfaga.

Solución 6.7.

- *Para $\forall x \forall y f(x,y) = x$, por ejemplo*
  1. *para que la satisfaga, se puede definir una estructuva $\mathcal{A} = (\mathbb{N}, f_A, =)$ donde $f_A(x,y) = x$ (proyección en la primer coordenada).*
  2. *para que la satisfaga, se puede definir una estructuva $\mathcal{A} = (\mathbb{N}, f_A, =)$ donde $f_A(x,y) = y$ (proyección en la segunda coordenada).*
- *Para $\exists x \forall y f(x,y) = y$, por ejemplo*
  1. *para que no la satisfaga, se puede definir una estructuva $\mathcal{A} = (\mathbb{N}, f_A, =)$ donde $f_A(x,y) = y$ (proyección en la segunda coordenada).*
  2. *para que no la satisfaga, se puede definir una estructuva $\mathcal{A} = (\mathbb{N}, f_A, =, P)$ donde $f_A(x,y) = y + 1$.*
- *Para $\exists x (P(x) \wedge \forall y P(f(x,y)))$, por ejemplo*
  1. *para que la satisfaga, se puede definir una estructuva $\mathcal{A} = (\mathbb{N}, f_A, =, P)$ donde $f_A(x,y) = x$ (proyección en la primer coordenada) y $P(x) =$ 'x es par'.*
  2. *para que no la satisfaga, se puede definir una estructuva $\mathcal{A} = (\mathbb{N}, f_A, =, P)$ donde $f_A(x,y) = x + 1$ y $P(x) =$ 'x es par'.*

Ejercicio 6.8.
Sea $\mathcal{L} = \{E, R\}$ y considere las fórmulas

$$\varphi_1 : (\forall xy)(E(x,y) \to R(x))$$
$$\varphi_2 : (\forall x)(((\forall y)E(x,y)) \to R(x))$$

a. Hallar $\mathcal{L}$-estructuras $\mathcal{A}_1$ y $\mathcal{A}_2$ tales que

- $\mathcal{A}_1 \not\models \varphi_1$ y $\mathcal{A}_1 \models \varphi_2$.
- $\mathcal{A}_2 \models \varphi_1$ y $\mathcal{A}_2 \models \varphi_2$.

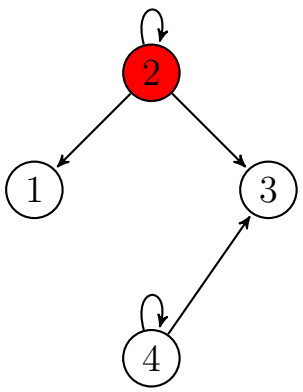
b. >Se podrá construir un modelo que satisfaga φ_1 pero no a φ_2?

Solución 6.8.
Notemos que las interpretaciones bajo $\mathcal{L} = \{E, R\}$ suelen ser grafos con ciertos vértices coloreados donde, por ejemplo, $E(x,y)$ se puede interpretar como "del nodo x sale una flecha hacia el nodo y" y $R(x)$ como "el nodo x es rojo".

a. Luego, es posible describir las fórmulas de la siguiente manera

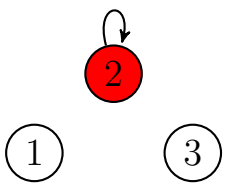
φ_1 :*"para todo par de nodos x e y, si hay una flecha de x a y entonces x es un nodo rojo"*
φ_2 :*"para todo x, si de x sale una flecha a cualquier otro nodo y entonces x es un nodo rojo"*

Sea, por ejemplo, $\mathcal{A}_1$ el siguiente modelo



Es claro que $\mathcal{A}_1$ no satisface φ_1 porque hay un nodo representado por (4) del cual salen flechas pero no es rojo. Y, por otro lado, satisface φ_2 puesto que no hay nodos que cumplan con la premisa de la implicación.

Sea, por ejemplo, $\mathcal{A}_2$ el siguiente modelo



$\mathcal{A}_2$ satisface φ_1 y φ_2 porque hay un solo nodo del que salen flechas y resulta ser rojo.

b. No es posible construir un modelo que satisfaga φ_1 pero no a φ_2.

Supongamos que existe un modelo $\mathcal{A}$ tal que $\mathcal{A} \not\models \varphi_2$ entonces existe $a \in \mathcal{A}$ tal que $E_{\mathcal{A}}(a, b)$ para todo $b \in \mathcal{A}$ con a distinto de rojo pero como $\mathcal{A} \models \varphi_1$ se sigue que como $E_{\mathcal{A}}(a, b)$ se cumple a debe ser rojo. Lo que resulta ser una contradicción.

Ejercicio 6.9.
Decimos que un elemento e del universo de una interpretación $\mathcal{I}$ es distinguible *con el lenguaje $\mathcal{L}$ si existe una $\mathcal{L}$-fórmula $\varphi(x)$ con una sola variable libre x tal que $\mathcal{I} \models \varphi(x)[v]$ si y sólo si $v(x) = e$.*

Dar un ejemplo de un lenguaje y una interpretación de dicho lenguaje con universo infinito tal que todo elemento del universo de la interpretación dada sea distinguible.

Solución 6.9.
Se podría tomar el universo $\mathbb{N}$ (sin el cero), junto con el predicado $<$ y la interpretación usual. Puesto que, por ejemplo,

- *El elemento 1 es distinguible,*
$$\varphi_1(x) : \nexists x_1(x_1 < x)$$
- *el elemento 2 es distinguible (hay un número menor que 2 y ninguno menor a ese número)*
$$\varphi_2(x) : \exists x_1(x_1 < x \wedge (\nexists x_1(x_2 < x_1)))$$
- *y de manera inductiva, se puede deducir que el elemento k es distinguible*
$$\varphi_k(x) : (\exists x_1, \ldots, x_{k-1})((x_1 < x) \wedge (x_2 < x) \wedge (x_3 < x) \wedge \ldots \wedge (x_{k-2} < x_{k-1}) \wedge (\nexists x_k(x_k < x)))$$

Ejercicio 6.10.
Sea $\mathcal{L}$ el lenguaje con igualdad que consiste únicamente de un predicado binario R. Sea $\mathcal{M}$ una $\mathcal{L}$-interpretación que consiste en un árbol ($R_{\mathcal{M}}$ es la relación de accesibilidad: $xR_{\mathcal{M}}y$ si y sólo si y es hijo de x). Demostrar que la raíz de M es un elemento distinguible.

Solución 6.10.
En efecto, se puede distinguir a la raíz con la fórmula:

$$\varphi : \forall y(\neg(yRx)).$$

Ejercicio 6.11.
Sea $\mathcal{L}$ un lenguaje con igualdad y un símbolo de función binario, y sean $\mathcal{I}_1$ e $\mathcal{I}_2$ las siguientes interpretaciones:

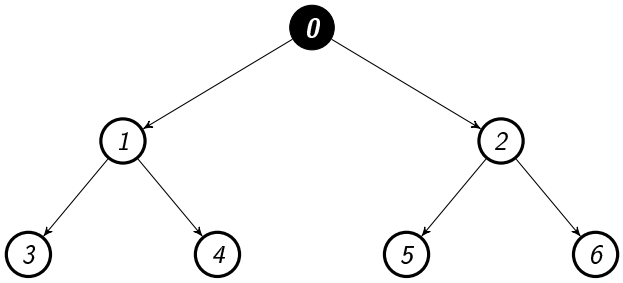
$$\mathcal{I}_1 = (\mathbb{N}, +) \qquad \mathcal{I}_2 = (\mathbb{N}, \cdot)$$

donde $\mathbb{N}$ denota el conjunto de los números naturales. Probar que 1 es un elemento distinguido en ambas interpretaciones.

Solución 6.11.

1. *para $\mathcal{I}_1 = (\mathbb{N}, +)$, la idea es decir 'para todo número natural z distinto de 0 existe un número natural y distinto de z tal que $x + y = z$'*
$$\varphi(x) : \forall z(\underbrace{\neg(\forall y(+(x, y) = y))}_{z \text{ distinto de } 0} \rightarrow (\exists y((y \neq z) \wedge +(x, y) = z)))$$
2. *para $\mathcal{I}_2 = (\mathbb{N}, \cdot)$, $\varphi(x) : \forall y(\cdot(x, y) = y)$*

Ejercicio 6.12.
Sea $\mathcal{L} = \{c, f, R, =\}$ donde c es un símbolo constante, f es un símbolo de función unario y R un símbolo de relación binaria y consideremos la estructura $\mathcal{T}$ definida por:



Donde en este caso $f_{\mathcal{T}} = id : \{0, \dots, 6\} \to \{0, \dots, 6\}$, $c_{\mathcal{T}} = 0$ *y* $R_{\mathcal{T}}$ *el conjunto de flechas del árbol.*

Verificar si las fórmulas

$$\varphi_1 : \forall x(\exists y(f(y) = x))$$
$$\varphi_2 : \forall x R(x, f(x))$$
$$\varphi_3 : \forall x(x = c) \vee \forall x(f(x) \neq x \wedge f(f(x)) = x)$$

son válidas para esta estructura.

Solución 6.12.

1. *Se puede interpretar a* φ_1 *como la propiedad sobreyectiva de una función. En este modelo, como* f *es la función identidad (y siempre resulta sobreyectiva) es claro que* $\mathcal{T} \models \varphi_1$.

2. *Sea* v *valuación de* $\mathcal{T}$ *y sea* $v(x) = a$ *en* $\{0, \dots, 6\}$. *Como* f *se interpreta como la función identidad, entonces*

$$v(f(x)) = f_{\mathcal{T}}(v(x)) = id(a) = a$$

   Falta ver que se cumple $aR_{\mathcal{T}}a$ *pero, para este modelo,* a *no está relacionado con* a.

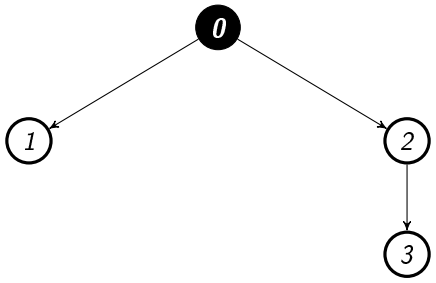
   Luego, $\mathcal{T} \not\models \varphi_2$.

3. *Veamos que* $\mathcal{T} \not\models \varphi_3$.

   En primer lugar, $\mathcal{T} \not\models (\forall x(x = c))$ *puesto que los elementos que están en el dominio no son el elemento distinguido. Y, por último, como* f *se interpreta como la función identidad es claro que* $\mathcal{T} \not\models (\forall x(f(x) \neq x))$.

$$\underbrace{\forall x(x = c)}_{\mathcal{T} \not\models} \vee \underbrace{\forall x(\underbrace{f(x) \neq x}_{\mathcal{T} \not\models} \wedge f(f(x)) = x)}_{\substack{\mathcal{T} \not\models \\ \text{por ser una conjunción}}} \Rightarrow \mathcal{T} \not\models \varphi_3$$

Ejercicio 6.13.
Sea $\mathcal{L} = \{c, f, R, =\}$ *donde* c *es un símbolo constante,* f *es un símbolo de función unario y* R *un símbolo de relación binaria y consideremos la estructura* $\mathcal{T}$ *definida por:*



Donde en este caso $f_{\mathcal{T}} = id : \{0, \dots, 3\} \to \{0, \dots, 3\}$, $c_{\mathcal{T}} = 0$ *y* $R_{\mathcal{T}}$ *el conjunto de flechas del árbol.*

Demostrar que todos elementos de $\mathcal{T}$ *son distinguibles.*

Solución 6.13.
Para el nodo 0, *basta con notar que se puede definir*

$$\varphi_0(x) : x = c$$

de manera tal que resulta ser una fórmula con una única variable libre donde sólo es veradera cuando $x = 0$.

Otra manera de distinguir el nodo 0 *es observando que 'no le entran flechas'. Es decir,*

$$\varphi_0(x) : \forall y(\neg R(y, x))$$

Para el nodo 1*, se puede notar que de él entran pero no salen flechas*

$$\varphi_1 : R(c,x) \wedge \neg\exists y(R(x,y))$$

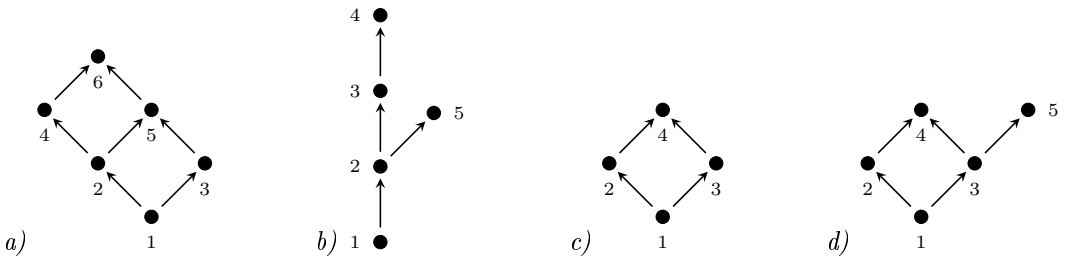
Para el nodo 2*, se puede notar que de él entran pero y salen flechas*

$$\varphi_2 : R(c,x) \wedge \exists y(R(x,y))$$

Por último, para el nodo 3 *se puede notar, por ejemplo, que no hay una flecha del nodo* 0 *a él*

$$\varphi_3 : \neg R(c,x) \wedge \neg\exists y(R(x,y))$$

Ejercicio 6.14.
Sea $\mathcal{L}$ *un lenguaje de primer orden con igualdad y con un símbolo de predicado binario* $\leq$*. Probar que todos los elementos del universo de la siguientes interpretaciones son distinguibles.*



Observación: Estos esquemas se conocen como 'Diagramas de Hasse' y la relación que describen es la menor relación reflexiva y transitiva que contiene a los pares explicitados en el diagrama.
Por ejemplo, en a), se tienen los pares $(1,1),(2,2),(2,6)$ *entre otros aunque no estén explícitamente en el esquema.*

Solución 6.14.
Para mayor comodidad definimos la igualdad como una reescritura:

$$x = y \equiv x \leq y \wedge y \leq x$$

De forma similar definimos el menor estricto como:

$$x < y \equiv x \leq y \wedge \neg(y = x)$$

a. Los siguientes seis predicados se verifican en un solo elemento del diagrama, y cada elemento verifica uno solo de ellos.

- *El mínimo*
$$\forall y(x \leq y)$$
- *El que está a la derecha del mínimo:*
$$\exists y\forall z(z \leq x \to y = z) \wedge \exists z\exists y(z \neq y \wedge z \neq x \wedge y \neq x \wedge \forall w(x \leq w \to (w = y \vee w = z \vee w = x)))$$
(tiene por lo menos uno abajo y exactamente dos arriba).
- *El que está a la izquierda del mínimo:*

$$\exists y\forall z(z \leq x \to z = y) \wedge$$
$$\exists y\exists z\exists w(x \leq w \wedge x \leq z \wedge x \leq y \wedge z \neq y \wedge z \neq w \wedge z \neq x \wedge y \neq w \wedge y \neq x \wedge w \neq x \wedge \forall v(x \leq v \to (v = w \vee v = y \vee v = z)))$$

(hay uno abajo y exactamente tres arriba).
- *El que está a la izquierda del máximo:*
$$\exists y\forall z(x \leq z \to y = z) \wedge \exists z\exists y(z \neq y \wedge z \neq x \wedge y \neq x \wedge \forall w(w \leq x \to (w = y \vee w = z \vee w = x)))$$
- *El que está a la derecha del máximo:*

$$\exists y\forall z(x \leq z \to z = y) \wedge$$
$$\exists y\exists z\exists w(w \leq x \wedge z \leq x \wedge y \leq x \wedge y \neq z \wedge w \neq z \wedge z \neq x \wedge y \neq w \wedge y \neq x \wedge w \neq x \wedge \forall v(v \leq x \to (v = w \vee v = y \vee v = z)))$$

- *El máximo:*

$$\forall y(y \leq x)$$

b. Los siguientes cinco predicados se verifican en un sólo elemento del diagrama, y cada elemento verifica uno sólo de ellos.

- *El mínimo:*

$$\forall y(x \leq y)$$

- *El que está arriba del mínimo:*

$$\exists y(y \neq x \land y \leq x \land \forall z(z \leq x \to z = y)) \land \exists y \exists z(z \neq y \land z \neq x \land y \neq x \land x \leq z \land x \leq y)$$

(Tiene exactamente uno abajo y dos arriba).

- *El que sobresale:*

$$\exists y \exists z(y \neq x \land z \neq y \land z \neq y \land y \leq x \land z \leq x \land \forall w(w \leq x \to (w = y \lor w = z \lor w = x)) \land \neg \exists y(y \neq x \land x \leq y))$$

(Tiene exactamente dos abajo y ninguno arriba).

- *El de abajo del 'máximo':*

$$\exists y(y \neq x \land x \leq y \land \forall z(x \leq z \to (z = y \lor x = z)))$$

(Tiene exactamente uno arriba).

- *El máximo:*

Tomo la conjunción de las negaciones de todos los predicados anteriores.
Hay un solo elemento que la cumple, y es éste.

c. Veamos si el elemento 4 es distinguible del resto de los nodos.
Sea $\phi_4 = \forall y(y \leq x)$

$$\begin{aligned}
&I, v \models \forall y(y \leq x) \\
&\quad \textit{si y sólo si } \textit{ todo valor } a \textit{ en } U_I \textit{ cumple } I, v[y \mapsto a] \models y \leq x \\
&\quad \textit{si y sólo si } \textit{ todo valor } a \textit{ en } U_I \textit{ cumple } v[y \mapsto a](y) \leq v[y \mapsto a](x) \\
&\quad \textit{si y sólo si } \textit{ todo valor } a \textit{ en } U_I \textit{ cumple } a \leq_I v(x) \\
&\quad \textit{si y sólo si } \textit{ todas las siguientes se cumplen} \begin{cases} 1 \leq_I v(x) \\ 2 \leq_I v(x) \\ 3 \leq_I v(x) \\ 4 \leq_I v(x) \end{cases} \\
&\quad \textit{si y sólo si } x = 4
\end{aligned}$$

d. Notemos que el 1 se distingue de la misma forma que el elemento 4. Veamos cómo distinguir el 4.

El 4 ya no es mayor o igual a todos pues no es más grande que el 5. Es el único elemento que tiene 3 elementos distintos que son menores que él.

$$\phi_4 = \exists y, z, w(y \neq z \land z \neq w \land w \neq y \land y < x \land z < x \land w < x)$$

Análogamente, el 5 es el único elemento que tiene exactamente 2 elementos distintos menores que él.

$$\phi_5 = \exists y, z(y \neq z \land y < x \land z < x \land \forall w(w < x \to (w = y \lor w = z)))$$

Por último, veamos cómo distinguir al 2 y al 3.

$$\phi_2 = \exists y(x < y \land \forall z(x < z \to z = y))$$

$$\phi_3 = \exists y, z(y \neq z \land x < y \land x < z \land \forall w(x < w \to (w = y \lor w = z)))$$

En todos los casos falta probar que efectivamente las ϕ cumplen lo pedido en la interpretación dada, es decir, como se hizo para ϕ_4 en el inciso anterior.

Ejercicio 6.15.
Sea $\mathcal{L}$ un lenguaje de primer orden con un símbolo de predicado binario $\leq$. Considerar una interpretación I tal que U_I es un universo finito y totalmente ordenado.

Probar que todos los elementos de U_I son distinguibles.

Solución 6.15.
Como U_I es un universo finito y totalmente ordenado podemos reescribirlo como $\{e_1, \ldots, e_n\}$ donde $e_i \leq e_j$ si y sólo si $i \leq j$. Además sabemos que $e_i < e_j$ pues $e_i \neq e_j$.

Debemos definir n predicados $\phi_i, \ldots, \phi_n$ tales que cada uno tiene una única variable libre x y cada ϕ_i es válido únicamente en valuaciones v tales que $v(x) = e_i$.

Distingamos primero a e_1. Entonces e_1 cumple que es menor o igual a todos los demás.

$$\phi_1(x) = \forall y(x \leq y)$$

Para distinguir a 2, hay que notar que e_2 es el único elemento estrictamente mayor a e_1 que es más cercano a e_1.

$$\phi_2(x) = \exists z(\phi_1(z) \wedge z < x \wedge \forall w((z \leq w \wedge w \leq x) \to (w = z \vee w = x)))$$

En general, podemos definir ϕ_i recursivamente:

$$\begin{aligned}\phi_1(x) &= \forall y(x \leq y)\\ \phi_{i+1}(x) &= \exists z(\phi_i(z) \wedge z < x \wedge \forall w((z \leq w \wedge w \leq x) \to (w = z \vee w = x)))\end{aligned}$$

Sólo resta mostrar, vía inducción, que $\phi_i(x)$ es verdadera sólo si la valuación hace a $x = e_i$.

C.B. *La fórmula $\phi_1(x)$ es cierta sólo cuando x es un elemento menor o igual a todos los demás. Tal elemento es e_1.*

P.I. *Sabemos que la fórmula $\phi_i(z)$ es únicamente verdadera en valuaciones v tales que $v(z) = e_i$.*

Partiendo de esto, queremos ver que $\phi_{i+1}(x)$ sólo es verdadera cuando $v(x) = e_{i+1}$.

Por hipótesis inductiva sabemos que la valuación v es tal que $v(z) = e_i$.

Por lo tanto, para que la valuación haga verdadera a $z < x$ debe suceder que $v(x)$ sea un elemento mayor estricto que $v(z)$.

Además, como $\forall w((z \leq w \wedge w \leq x) \to (w = z \vee w = x))$ sabemos que todo elemento entre z y x es necesariamente z o necesariamente x, es decir, no hay elementos entre z y x.

Finalmente, teniendo en cuenta que x es un elemento estrictamente mayor a z y tal que está a distancia mínima de z, y $v(z) = e_i$, luego $v(x) = e_{i+1}$.

Ejercicio 6.16.
Sea $\mathcal{L}$ el lenguaje de primer orden con igualdad, con una relación binaria $<$. Considerar la siguiente $\mathcal{L}$-interpretación $\mathcal{M}$ con universo $\omega^2 = \{(x, y) | x, y \in \mathbb{N}\}$ y

$$(x, y) <_{\mathcal{M}} (x', y') \text{ si y sólo si } (x < x' \vee (x = x' \wedge y < y'))$$

1. *Demostrar que son distinguibles todos los elementos de la forma $(i, 0)$ con $i \in \mathbb{N}_0$.*
2. *Un orden total finito es una $\mathcal{L}$-estructura $\mathcal{A}$ con dominio $A = \{a_1, \ldots, a_n\}$ y tal que $<_{\mathcal{A}}$ es la clausura transitiva del conjunto $\{(a_1, a_2), (a_2, a_3), \ldots, (a_{n-1}, a_n)\}$.*

   Demostrar que todo elemento de un orden total finito, es distinguible.

Solución 6.16.

1. - *Vamos a construir inductivamente predicados φ_i, tales que cada uno tiene una única variable libre y y cada φ_i es válido únicamente en valuaciones v tales que $v(y) = (i, 0)$.*

   Para el caso base, definimos

   $$\varphi_0 : \neg(\exists x(x < y))$$

   Efectivamente tiene como única variable libre a y, y sólo se satisface en $\mathcal{M}$ cuando $v(y) = (0, 0)$.

   Ahora, supongamos que tenemos a φ_i construida para todo $i \in \mathbb{N}$. Veamos que podemos construir a φ_{n+1}. Para eso, primero construimos otra fórmula con variable libre z que dice que z es de la forma $(i, 0)$ para algún i. Notar que estos son exactamente aquellos elementos que no tienen un antecesor inmediato. Entonces, como paso inicial, construyamos primero una fórmula que determina que un elemento no tiene antecesor inmediato.

   $$\psi : \forall x(x < z \to \exists x_2(x < x_2 \wedge x_2 < z))$$

   Ahora definimos

   $$\varphi_{n+1} : \psi(y) \wedge \neg(\varphi_0(y)) \wedge \ldots \wedge \neg(\varphi_n(y)) \wedge \forall x((\psi(x) \wedge x < y) \to (\varphi_0(x) \vee \varphi_1(x) \vee \ldots \vee \varphi_n(x)))$$

   Observar que φ_{n+1} distingue a $(n + 1, 0)$, como queríamos.
   Entonces, probamos que todos los elementos de $\mathcal{M}$ de la forma $(i, 0)$ son distinguibles.

- *Misma forma pero planteada desde la práctica:*
  Sean $a, b \in \mathbb{R}^2$, para distinguir el par $(0,0)$ basta con definir

$$\psi_0(a) : \forall b(a <_{\mathcal{M}} b)$$

Para distinguir los siguientes $(i,0)$ con $i \in \mathbb{N}$ definimos para todo $a, b, z \in \mathbb{R}^2$

$$\psi(a) : \forall b((y <_{\mathcal{M}} a) \to (\exists z(b <_{\mathcal{M}} z \wedge z <_{\mathcal{M}} a))$$

(la fórmula expresa que no hay un predecesor inmediato)
Notar que el par $(1,0)$ el primer elemento (después del par $(0,0)$) que satisface ψ y que el par $(2,0)$ el primer elemento (después de los pares $(0,0)$ y $(1,0)$) que satisface ψ. Por lo tanto,

$$\psi_1(a) : \psi(a) \wedge \forall b((b <_{\mathcal{M}} a) \to (\neg\psi(b) \vee \varphi_0(b)))$$

$$\psi_2(a) : \psi(a) \wedge \forall b((b <_{\mathcal{M}} a) \to (\neg\psi(b) \vee \varphi_0(b) \vee \varphi_1(b)))$$

por lo tanto, de manera inductiva,

$$\psi_n(a) : \psi(a) \wedge \forall b((b <_{\mathcal{M}} a) \to (\neg\psi(b) \vee \varphi_0(b) \vee \varphi_1(b) \vee \ldots \vee \varphi_{n-1}(b))).$$

2. *Vamos a describir una fórmula φ_i para cada $i = 1, \ldots, n$.*

   Notar que a_1 es el único elemento de A tal que $a_1 <_A$ y para todo $y \in A$ distinto de a_1. Esto lo podemos traducir a primer orden como

$$\varphi_1(x) : \forall y(y \neq x \to x < y)$$

   Ahora podemos distinguir a_2 aprovechando la fórmula anterior, después de todo, a_2 es el sucesor inmediato de a_1.

$$\varphi_2(x) : \forall y(\forall z(\varphi_1(y) \wedge y < z \to (z = x \vee x < x))).$$

   Luego, para definir a_i con $2 \leq i \leq n$ sabiendo que a_{i-1} es distinguible mediante la fórmula φ_{i-1}, realizamos un proceso similar al caso $i = 2$, es decir, definimos φ_i simplemente como

$$\varphi_i : \forall yz(\varphi_{i-1}(y) \wedge y < z \to (z = x \vee x < z)).$$

   Luego, por inducción, tenemos que todos los elementos de $\mathcal{A}$ son distinguibles.

   Observación: *También podríamos haber definido un orden total como una estructura que satisface las siguientes fórmulas*

$$\begin{aligned}
&\phi_1 : \forall x\neg(x < x) && \textit{(Antirreflexividad)}\\
&\phi_2 : \forall x(\forall y(x < y \to \neg(y < x))) && \textit{(Asimetría)}\\
&\phi_3 : \forall x(\forall y(\forall z(x < y \wedge y < z \to x < z))) && \textit{(Transitividad)}\\
&\phi_4 : \forall x(\forall y(x < y \vee x = y \vee y < x)) && \textit{(Totalidad)}
\end{aligned}$$

   A la conjunción de todas estas fórmulas la denotamos por φ_{ot}.

Ejercicio 6.17.
Probar que si el universo de una interpretación es finito con $n+1$ elementos, y tiene la propiedad que n elementos del universo son distinguibles, entonces todos los elementos son distinguibles.

Solución 6.17.
Supogamos que el dominio de esta estructura es el conjunto $\{a_0, a_1, \ldots, a_n, a_{n+1}\}$. Como existe la propiedad de que n elementos del universo son distinguibles, existe la fórmula φ_1 que distingue a a_1, φ_2 que distingue a a_2; es decir, existe la fórmula φ_j que distingue a a_j para $1 \leq j \leq n$

Sólo faltaría poder distinguir el elemento a_{n+1}. Entonces usando una valuación que interpreta la variable libre de φ_1 como a_{n+1} es claro que la fórmula φ_1 no se podrá satisfacer. De la misma manera, se puede decir lo mismo para φ_2 y así para cada $j \in \{3, \ldots, n\}$.

Por lo tanto, una manera de distinguir el último elemento es

$$\varphi_{n+1} = \neg\varphi_1 \wedge \neg\varphi_2 \wedge \ldots \neg\varphi_n.$$

Ejercicio 6.18. *Dada una interpretación $\mathcal{I}$ con universo A, decimos que una relación $R \subseteq A^n$ es* expresable *con el lenguaje $\mathcal{L}$ si existe una $\mathcal{L}$-fórmula $\varphi(x_1, \ldots, x_n)$ con n variables libres tal que para toda valuación v cumpla $\mathcal{I} \models \varphi(x_1, \ldots, x_n)[v]$ si y sólo si $(v(x_1), \ldots, v(x_n)) \in R_{\mathcal{I}}$.*

Demostrar que las siguientes relaciones son expresables.

a. $\mathcal{I}_1 = \langle \mathbb{N}, \times, = \rangle$ *con* $\times$ *el producto de naturales.*
$R_1 = \{(n, m) : n \text{ divide a } m\}$.
$P_1 = \{n : n \text{ es primo}\}$.

b. $\mathcal{I}_2 = \langle \mathbb{N}, +, =, 0, 1 \rangle$ *con* $+$ *la suma de naturales.*
$R_2 = \{(n, m) : n < m\}$.

c. $\mathcal{I}_3 = \langle L, \circ, = \rangle$ *con* L *el conjunto de todas las listas,* $\circ$ *la concatenación de listas.*
$R_3 = \{(a, b) : a \text{ es sublista de } b\}$.

d. $\mathcal{I}_4 = \langle \mathbb{N}, +, = \rangle$ *con* $+$ *la suma de naturales.*
$R_4 = \{(x_1, x_2) | x_1 \leq x_2\}$

e. $\mathcal{I}_5 = \langle \mathbb{N}, +, = \rangle$ *con* $+$ *la suma de naturales.*
$R_5 = \{(x_1, x_2) | x_1 < x_2\}$

Solución 6.18.

a.

b.

c.

d. La siguiente fórmula, de variables libres x_1 *y* x_2 *expresa* R_4

$$\varphi_{\leq} : \exists x(+(x_1, x) = x_2)$$

e. La siguiente fórmula expresa R_5 *al poner la condición que el nuevo sumando no sea el* 0*:*

$$\varphi_{<} : \exists x(\neg(+(x, x) = x) \wedge +(x_1, x) = x_2)$$

Ejercicio 6.19. *Decimos que una clase de modelos* K *es* definible *con el lenguaje* $\mathcal{L}$ *si existe una sentencia* φ *tal que para toda interpretación* $\mathcal{I}$ *y valuación* v *cumpla* $\mathcal{I} \models \varphi[v]$ *si y sólo si* $\mathcal{I} \in \mathsf{K}$.

Demostrar que las siguientes clases de modelos son definibles en sus respectivos lenguajes.

1. $\mathcal{L}_0 = \{=\}$. $\mathsf{K}_0 = \emptyset$.
2. $\mathcal{L}_1 = \{=\}$. $\mathsf{K}_1 = \{\text{todas las interpretaciones}\}$.
3. $\mathcal{L}_2 = \{P, =\}$ *con* P *predicado binario.* $\mathsf{K}_2 = \{\mathcal{I} : P^{\mathcal{I}} \text{ es reflexivo y transitivo}\}$.
4. $\mathcal{L}_3 = \{f, g, =\}$ *con* f, g *funciones unarias.* $\mathsf{K}_3 = \{\mathcal{I} : Im\, f^{\mathcal{I}} \subseteq Im\, g^{\mathcal{I}}\}$.
5. $\mathcal{L}_4 = \langle R, = \rangle$ *donde* R *es un símbolo de relación binaria.*
$\mathsf{K}_4 = \{\mathcal{I} : R^{\mathcal{I}} \text{ es irreflexivo y transitiva y con al menos un mínimo}\}$.

Solución 6.19.

1.

2.

3.

4. *Una sentencia que sirve es:*

$$\varphi : \underbrace{\forall x(\neg(xRx))}_{irreflexividad} \wedge \underbrace{\forall x(\forall y(\forall z((xRy \wedge yRz) \to xRz)))}_{transitividad} \wedge \underbrace{\exists x(\forall y(x \neq y \to \neg(yRx)))}_{al\ menos\ un\ mínimo}$$

Si se quiere, se puede acortar a otra sentencia equivalente (usando irreflexividad al final):

$$\varphi' : \forall x(\forall y(\forall z(\neg(xRx) \wedge (xRy \wedge yRz \to xRz)))) \wedge \exists x(\forall y(\neg(yRx)))$$

Ejercicio 6.20.
Sea $\mathcal{L} = \{=, f\}$ *un lenguaje con igualdad y una función unaria. Demostrar que no es definible la clase de modelos en los cuales* $f_{\mathcal{M}}$ *es finitos-a-uno.*

Una función $f_{\mathcal{M}}$ *es finitos-a-uno si para cualquier elemento* y *de su imagen, la preimagen* $\{x : f_{\mathcal{M}}(x) = y\}$ *de* y *es finita. Es decir, demostrar que no existe una sentencia* φ *tal que* $\mathcal{M} \models \varphi[v]$ *si y sólo si* $f_{\mathcal{M}}$ *es finitos-a-uno.*

Solución 6.20.
Supongamos que existe tal φ. Tenemos que buscar las fórmulas de negación incremental de φ. Como la condición pedida es un para todo ('para cualquier elemento y'), es natural pensar en negaciones incrementales existenciales. Por ejemplo, que φ_i sea 'existe un y con preimagen de cardinal al menos i':

$$\begin{aligned}
&\varphi_1 = \exists y, x_1, f(x_1) = y \\
&\varphi_2 = \exists y, x_1, x_2, f(x_1) = y \wedge f(x_2) = y \wedge \mathit{todosDistintos}(x_1, x_2) \\
&\varphi_3 = \exists y, x_1, x_2, x_3, f(x_1) = y \wedge f(x_2) = y \wedge f(x_3) = y \wedge \mathit{todosDistintos}(x_1, x_2, x_3) \\
&\quad\vdots \\
&\varphi_i = \exists y, x_1, \ldots, x_i, f(x_1) = y \wedge f(x_2) = y \wedge \ldots \wedge f(x_i) = y \wedge \mathit{todosDistintos}(x_1, x_2, \ldots, x_i) \\
&\quad\vdots
\end{aligned}$$

El problema es que cada φ_i puede referirse a un contraejemplo y distinto, con lo cual, incluso todas juntas, no contradicen φ. Por ejemplo, el modelo $\mathcal{M}$ con universo $\mathbb{N}$ y $f_{\mathcal{M}}(x) = \lfloor \sqrt{x} \rfloor$ cumple φ y todos los φ_i.

Para resolver el problema tenemos que usar variables libres en las fórmulas. Si en lugar de poner el cuantificador de y, dejamos y libre, obligamos a que en todas las φ_i la y tome el mismo valor (el que le asigna la valuación). Como la satisfacibilidad está definida como 'existe un modelo y una valuación', esto podemos interpretarlo como que tenemos un existencial de y por fuera de las φ_i, y por lo tanto, es el mismo existencial el que liga las ocurrencias de y en cada una de las fórmulas.

Veámoslo concretamente. Ahora pensamos $\varphi_i(y)$ como 'y tiene una preimagen de cardinal al menos i'.

$$\begin{aligned}
&\varphi_1(y) = \exists y, x_1, f(x_1) = y \\
&\varphi_2(y) = \exists y, x_1, x_2, f(x_1) = y \wedge f(x_2) = y \wedge \mathit{todosDistintos}(x_1, x_2) \\
&\varphi_3(y) = \exists y, x_1, x_2, x_3, f(x_1) = y \wedge f(x_2) = y \wedge f(x_3) = y \wedge \mathit{todosDistintos}(x_1, x_2, x_3) \\
&\quad\vdots \\
&\varphi_i(y) = \exists y, x_1, \ldots, x_i, f(x_1) = y \wedge f(x_2) = y \wedge \ldots \wedge f(x_i) = y \wedge \mathit{todosDistintos}(x_1, x_2, \ldots, x_i) \\
&\quad\vdots
\end{aligned}$$

Ahora si, consideremos el conjunto $\Gamma = \{\varphi, \varphi_1, \varphi_2, \ldots\}$. Vemos que es Γ es insatisfacible: supongamos que es satisfacible, entonces existen $\mathcal{M}$ y v tal que $\mathcal{M} \models \Gamma[v]$.

Luego, por $\varphi \in \Gamma$ el elemento $v(y)$ tiene una preimagen finita. Llamemos $k \in \mathbb{N}$ al cardinal de su preimagen. Vemos ahora que $\mathcal{M} \not\models \varphi_{k+1}[v]$. Llegamos a un absurdo, por lo cual Γ es insatisfacible.

Ahora consideremos un subconjunto finito $\Gamma_0 \subseteq \Gamma$. Sea $M = \max\{i : \varphi_i \in \Gamma'\} \cup \{1\}$.

Consideremos el modelo $\mathcal{M}$ que propusimos antes con universo $\mathbb{N}$ y $f_{\mathcal{M}}$. Notemos que la preimagen de b es el intervalo $[(b-1)^2 + 1, b^2]$ y, por lo tanto, tiene $b^2 - (b-1)^2 = 2b - 1$ elementos.

Por un lado, la preimagen tiene una cantidad finita de elementos, por lo cual $\mathcal{M} \models \varphi[v]$ con cualquier v.

Por otro lado, si tomamos una valuación v tal que $v(y) = M$ notamos que la preimagen de $v(y)$ tiene $2M - 1 \geq M$ elementos, y por lo tanto $\mathcal{M} \models \varphi_i[v]$ para todo $i \leq M$, es decir, para todo $\varphi_i \in \Gamma_0$.

Por lo tanto,$\mathcal{M} \models \Gamma_0[v]$, es decir, Γ_0 es satisfacible. Como esto vale para cualquier subconjunt finito Γ_0, por compacidad, Γ es satisfacible.

Hemos llegado a un absurdo proveniente de suponer que existía una fórmula φ con las propiedades buscadas, por lo tanto, dicha fórmula no puede existir.

- *Otra manera de pensarlo:*

En la primer solución utilizamos las fórmulas φ_i de manera que contradigan exactamente la propiedad representada por φ. Es decir, para cada modelo $\mathcal{M}$ o vale $\mathcal{M} \models \varphi[v]$ para toda valuación v, o existe una valuación v tal que $\mathcal{M} \models \{\varphi_1, \varphi_2, \ldots\}[v]$.

Eso es lo más seguro a la hora de definir los φ_i. Sin embargo, a veces puede ser más cómodo contradecir 'de más'.

En particular, contradiciendo de más podemos llegar a requerir menos variables libres y/o puede ser más fácil demostrar la insatisfacibilidad de todas las fórmulas juntas.

Lo que seguro va a pasar es que, al ser las φ_i más restrictivas, encontrar el modelo para un subconjunto finito al demostrar la satisfacibilidad se puede dificultar. Si nos pasamos de la raya, puede ser que directamente no lo podamos encontrar.

Veamos un ejemplo de contradicción de más. Pensamos a φ_i como 'todos los elementos tienen una preimagen de

cardinal al menos i'.

$$\varphi_1 = \exists y, x_1, f(x_1) = y$$
$$\varphi_2 = \exists y, x_1, x_2, f(x_1) = y \wedge f(x_2) = y \wedge todosDistintos(x_1, x_2)$$
$$\varphi_3 = \exists y, x_1, x_2, x_3, f(x_1) = y \wedge f(x_2) = y \wedge f(x_3) = y \wedge todosDistintos(x_1, x_2, x_3)$$
$$\vdots$$
$$\varphi_i = \exists y, x_1, \ldots, x_i, f(x_1) = y \wedge f(x_2) = y \wedge \ldots \wedge f(x_i) = y \wedge todosDistintos(x_1, x_2, \ldots, x_i)$$
$$\vdots$$

Consideremos el conjunto $\Gamma = \{\varphi, \varphi_1, \varphi_2, \ldots\}$. Vemos que es Γ es insatisfacible: supongamos que es satisfacible, entonces existen $\mathcal{M}$ y v tal que $\mathcal{M} \models \Gamma[v]$.

Ahora, como nuestras fórmulas no tienen variables libres (son sentencias), no nos importa v. Por φ_i sabemos que todo elemento de $\mathcal{M}$ tiene una preimagen finita. Si tomamos uno cualquiera, sea k el tamaño de su preimagen. Ahora, φ_{k+1} se hace falsa porque hay un elemento con preimagen de tamaño estrictamente menor a $k+1$. Esta parte fue muy parecida a la resolución anterior en este caso, pero en lugar de elegir $v(y)$ pudimos elegir cualquier elemento del modelo. De paso, no tuvimos que andar lidiando con la v.

Ahora consideremos un subconjunto $\Gamma' \subseteq \Gamma$. Sea $M = \max\{i : \varphi_i \in \Gamma'\} \cup \{1\}$. El modelo que propusimos en la resolución anterior no funciona porque, para cualquier cantidad k, existen elementos de preimagen al menos k, en particular, para $k > M$. Esta es la parte fea de elegir φ_i mas restrictivas. Con un poco de suerte, podemos encontrar un modelo igual.

Necesitamos un modelo en el cual todos los elementos tengan una preimagen finita (para que cumpla φ) y a la vez de cardinal al menos M. Una opción es usar universo $\mathbb{N}$ y como función $f_{\mathcal{M}} = \left\lfloor \frac{x}{M} + 1 \right\rfloor$. De esta manera, la preimagen de todos los elementos tiene tamaño exactamente $M+1$ y cumplimos todos los requisitos. A partir de aquí se termina igual que en el otro caso, así que no vamos a repetir los detalles.

Una cosa interesante es que el modelo que usamos para esta segunda parte también servirá para la idea de resolución inicial: como las φ_i de esta segunda resolución son estrictamente más restrictivas, un modelo que sirve para ellas, también sirve para las versiones más laxas.

Como conclusión, está bueno recalcar que la elección de las φ_i supone un balance entre la simplicidad de una parte de la demostración vs. la simplicidad de la otra.

Ejercicio 6.21.
Sea $\mathcal{L}$ un lenguaje de primer orden con igualdad y un símbolo de predicado binario R. Decidir si es posible expresar en $\mathcal{L}$ la propiedad que afirma que existe $k \in \mathbb{N}$ tal que todo elemento se relaciona via R con a lo sumo con k elementos distintos.

Solución 6.21.
Veremos que la propiedad no es expresable. Esta clase de demostraciones de la no expresabilidad de una propiedad en primer orden se caracteriza por cinco pasos:

(1) Suponemos que sí es expresable. Sea φ_α la fórmula de primer orden que describe la propiedad del enunciado.

(2) Aquí el objetivo es crear un conjunto de fórmulas tales que para poder satisfacer a todas a la vez, φ_α debe ser falsa pero tales que subconjuntos finitos de esas fórmulas sí sean compatibles con φ_α.

Dicho de otra manera, debemos definir finitas fórmulas con las cuales vayamos negando incrementalmente a φ_α.

En este caso se podría definir:

$$\varphi_1 : \text{“Hay un elemento que se relaciona con al menos 1 elemento”}$$
$$\varphi_2 : \text{“Hay un elemento que se relaciona con al menos 2 elementos”}$$
$$\ldots$$
$$\varphi_i : \text{“Hay un elemento que se relaciona con al menos i elementos”}$$

De este modo, y uniendo todas las fórmulas, no podemos dar un k fijo para nuestra propiedad tal que satisfaga a todas las φ_i puesto que, en particular, no sería verdad para φ_{k+1}.

En primer orden, la definición de estas fórmulas quedaría expresada de la siguiente manera:

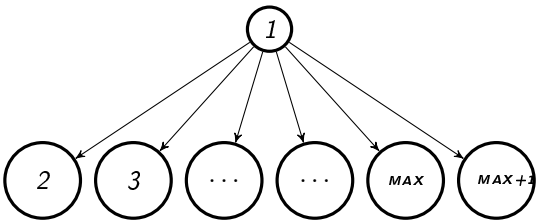
$$\varphi_1 : (\exists x)(\exists y_1) R(x, y_1)$$
$$\varphi_2 : (\exists x)(\exists y_1)(\exists y_2)(y_1 \neq y_2) R(x, y_1) \wedge R(x, y_2)$$
$$\ldots$$
$$\varphi_i : (\exists x)(\exists y_1)(\exists y_2) \ldots (\exists y_i) todosDistintos(y_1, \ldots, y_i) \wedge R(x, y_1) \wedge R(x, y_2) \wedge \ldots \wedge R(x, y_i)$$

donde $todosDistintos(y_1, \ldots, y_n)$ denota a la fórmula $y_1 \neq y_2 \wedge \ldots \wedge y_1 \neq y_n \wedge y_2 \neq y_3 \wedge \ldots \wedge y_2 \neq y_n \wedge \ldots \wedge y_{n-1} \neq y_n$

(3) Definimos el siguiente conjunto $\Gamma = \{\varphi_i : i \in \mathbb{N}\} \cup \{\varphi_\alpha\}$.

(4) Vamos a mostrar que el conjunto Γ *es tanto satisfacible como insatisfacible, llegando así al absurdo.*

- Γ *es insatisfacible:*
  Suponemos que es satisfacible. Es decir, existen $\mathcal{M}$ *y* v *tales que* $M, v \models \Gamma$.
  Luego, φ_α *es verdadera en el modelo* $\mathcal{M}$ *y la valuación* v *pues* $\varphi_\alpha \in \Gamma$.
  De aquí se desprende que existe k *fijo tal que todo elemento de* $\mathcal{M}$ *se relaciona a lo sumo con* k *elementos distintos de* $\mathcal{M}$.
  Por otro lado, como $\mathcal{M}, v \models \Gamma$ *y* $\{\varphi_i : i \in \mathbb{N}\} \subseteq \Gamma$, *entonces* φ_{k+1} *es verdadera en* $\mathcal{M}, v$.
  Por definición de φ_{k+1}, *existe un elemento en* $\mathcal{M}$ *tal que se relaciona con al menos* $k+1$ *elementos distintos, lo que representa un absurdo puesto que habíamos dicho que todos los elementos de* $\mathcal{M}$ *se relacionaban con a lo sumo* k *elementos.*
  Por ende, Γ *es insatisfacible.*
- Γ *es satisfacible:*
  Por compacidad vale que si todo $\Gamma_0 \subseteq \Gamma$ *es satisfacible, entonces* Γ *también lo es.*
  Tomemos $\Gamma_1 \subseteq \Gamma$ *finito y mostremos que dicho conjunto es satisfacible.*
  Como Γ_1 *es finito contiene una cantidad finita de* φ_i *(en particular, puede que no contenga ninguna* φ_i*). Lo que tenemos que ver es que existe un modelo* $\mathcal{M}$ *y una valuación* v *que hacen verdaderas a todas las fórmulas de* Γ_1.
  Tomemos $MAX = max\{i : \varphi_i \in \Gamma_1\} \cup \{1\}$. *La unión con el conjunto que sólo tiene al* 1 *se debe a que el primer conjunto podría ser vacío y queremos darle un valor a* MAX.
  Sólo resta construir el modelo $\mathcal{M}$ *y ver que se satisfacen todas las fórmulas de* Γ_1. *Sea* $\mathcal{M}$



  En primer lugar, notar que ningún valor de verdad de nuestras φ_i *depende de la valuación* v, *ya que se trata de sentencias, es decir, fórmulas con todas sus variables ligadas.*
  Usando al elemento 1 *como el testigo del existencial sobre* x *se satisfacen todas las* φ_i *en* Γ_1, *pues* 1 *se relaciona con al menos* MAX *elementos. También se satisface* φ_α, *en caso de que estuviera en* Γ_1, *pues todo elemento de* $\mathcal{M}$ *se relaciona con a lo sumo* MAX *elementos. Por ende,* Γ_1 *es satisfacible.*
  Como vimos que todo subconjunto finito de Γ *es satisfacible, concluimos por el Teorema de Compacidad que* Γ *es satisfacible.*

(5) Como probamos que Γ *es insatisfacible y, a la vez, satisfacible entonces llegamos a un absurdo, que provino de suponer que* φ_α *era expresable en primer orden.*

Ejercicio 6.22.
Sea $\mathcal{L}$ *un lenguaje de primer orden con igualdad y un símbolo de predicado binario* $\lhd$.
Demostrar que no es posible expresar en primer orden la proposición que afirma:

"para todo par de elementos x *e* y, *si* $x \lhd y$, *entonces hay una cantidad finita de elementos* x_i

que cumplen que $x \lhd x_i$ *y* $x_i \lhd y$."

Solución 6.22.
Veremos que la propiedad no es expresable. Esta clase de demostraciones de la no expresabilidad de una propiedad en primer orden se caracteriza por cinco pasos:

(a) Suponemos que sí es expresable. Sea φ_β *la fórmula de primer orden que describe la fórmula.*

(b) Definimos una serie de fórmulas que intenten contradecir incrementalmente a φ_β*:*

$$\varphi_0 : (\exists x)(\exists y)(x \lhd y)$$
$$\varphi_1 : (\exists x)(\exists y)(\exists z)(x \lhd y) \wedge (x \lhd z) \wedge (z \lhd y)$$
$$\dots$$
$$\varphi_i : (\exists x)(\exists y)(\exists z_1, \dots, z_i) todosDistintos(z_1, \dots, z_i) \wedge (x \lhd y) \wedge \bigwedge_{j=1}^{i} (x \lhd z_j) \wedge \bigwedge_{j=1}^{i} (z_j \lhd y)$$

donde $todosDistintos(z_1, \dots, z_n)$ *denota a la fórmula* $z_1 \neq z_2 \wedge \dots \wedge z_1 \neq z_n \wedge z_2 \neq z_3 \wedge \dots \wedge z_2 \neq z_n \wedge \dots \wedge z_{n-1} \neq z_n$

(c) Definimos el siguiente conjunto $\Gamma = \{\varphi_i : i \in \mathbb{N}_0\} \cup \{\varphi_\beta\}$.

(d) Vamos a mostrar que el conjunto Γ *es tanto satisfacible como insatisfacible, llegando así al absurdo.*

- Γ *es insatisfacible:*

  Suponemos que es satisfacible. Luego, hay un modelo y una valuación que hacen verdaderas a todas las fórmulas de Γ*, es decir:*

  $$\text{existen } \mathcal{M}, v \text{ tales que } \mathcal{M}, v \models \Gamma$$

  En particular, φ_β *es verdadera en dicho modelo y valuación, por lo que para todo par de elementos existe una cantidad finita de elementos 'entre' ambos (según la relación* $\lhd$*).*

  Por otro lado, las φ_i *nos dicen que hay un par de elementos del dominio para los cuales no existe una cantidad acotada de elementos entre ellos. Este razonamiento se desprende por cómo construimos las fórmulas en (b). Utilizamos dos variables libres para poder luego instanciarlas en un valor particular del dominio. Digamos que* $v(x) = a$ *y* $v(y) = b$*. Luego, las* φ_i *nos dicen que:*

  $$\begin{aligned}
  &\varphi_0(a,b) : \text{“entre } a \text{ y } b \text{ hay al menos } 0 \text{ elementos que se relacionan según } \lhd\text{”}\\
  &\varphi_1(a,b) : \text{“entre } a \text{ y } b \text{ hay al menos } 1 \text{ elementos que se relacionan según } \lhd\text{”}\\
  &\quad\dots\\
  &\varphi_i(a,b) : \text{“entre } a \text{ y } b \text{ hay al menos } i \text{ elementos que se relacionan según } \lhd\text{”}
  \end{aligned}$$

  De esta manera, para el par (a, b) *llegamos a una contradicción: existen a la vez una cantidad finita e infinita de elementos entre ellos.*

  Por ende, Γ *es insatisfacible.*

- Γ *es satisfacible:*

  Apelando al Teorema de Compacidad, basta con ver que todo subconjunto finito incluido en Γ *es satisfacible. Sea* $\Gamma_1 \subset \Gamma$ *finito y veamos que existe un modelo y una valuación que hacen verdaderas a todas las fórmulas en* Γ_1*, es decir, queremos ver que existen* $\mathcal{M}$ *y* v *tales que* $M, v \models \Gamma_1$*.*

  Tomamos $MAX = max\{i : \varphi_i \in \Gamma_1\} \cup \{1\}$*.*

  En este caso, nos basta con tomar el modelo usual de los enteros interpretando a $\lhd$ *como* $<_{\mathbb{Z}}$*. En él vale que para todo par de elementos hay una cantidad finita de elementos entre ellos por lo que satisfaceríamos* φ_β*.*

  Si la valuación v *instancia a* x *e* y *en* 1 *y* $MAX + 1$ *respectivamente, entonces es cierto que entre esos dos números hay al menos* MAX *elementos, por lo que satisfaceríamos a cada una de las* φ_i*.*

  Notar que aquí sí fue importante la valuación que utilizamos, ya que necesitamos que entre x *e* y *hubiera al menos* MAX *elementos. Luego, satisfacimos a todas las fórmulas en* Γ_1*.*

  Luego, por compacidad, Γ *es satisfacible*

(e) Como probamos que Γ *es insatisfacible y, a la vez, satisfacible entonces llegamos a un absurdo, que provino de suponer que* φ_β *era expresable en primer orden.*

Ejercicio 6.23.
Sea $\mathcal{L}$ *un lenguaje de primer orden con igualdad, y con numerables símbolos de constante* $c_1, c_2, \dots$ *>Es expresable la propiedad 'Cada constante tiene un valor distinto'?*

Solución 6.23.
Observando que esta propiedad es expresable si y sólo si también lo es su negación, que dice

'Hay al menos dos constantes (distintas) con la misma interpretación.'

Tratemos de llegar a un absurdo asumiendo que esta propiedad es expresable mediante una φ*.*

- *Definimos, para* $i, j \geq 1$ *e* $i \neq j$*,*

  $$\psi_{i,j} : \neg(c_i = c_j)$$

- *Definimos* $\Gamma = \{\psi_{i,j}\}_{i \neq j} \cup \{\varphi\}$*.*

- *Veamos que* Γ *es insatisfacible y satisfacible.*

  (a) *Insatisfacible:*

  Supongamos que $M, v \models \Gamma$*. Que valga* φ *significa que existen dos constantes diferentes tales que* $c_i^{\mathcal{M}} = c_j^{\mathcal{M}}$*. Por otro lado,* $\psi_{i,j}$ *se interpreta en* $\mathcal{M}, v$ *como* $c_i^{\mathcal{M}} \neq c_j^{\mathcal{M}}$*, absurdo.*

(b) *Satisfacible:*

Veamos ésto con compacidad. Sea Γ_1 *un subconjunto finito de* Γ*. Tomemos*

$$k = \text{máx}\{i | \text{existe un } j \text{ tal que } \psi_{i,j} \in \Gamma_1 \text{ o } \psi_{j,i} \in \Gamma_1\} \cup \{1\}$$

Basta entonces considerar a la estructura $\mathcal{M}$ *con universo* $\mathbb{N}$*, y donde definimos*

$$c_i^{\mathcal{M}} = \begin{cases} i & i \leq k \\ k+1 & i > k \end{cases}$$

Verificar que todas las fórmulas de Γ_1 *son válidas sobre* $\mathcal{M}$*.*

- *Como vimos que* Γ *es satisfacible e insatisfacible, llegamos a un absurdo que provino de nuestra suposición inicial que la propiedad 'Cada constante tiene un valor distinto' era expresable.*

Ejercicio 6.24.
Un cuadrado Sudoku es un retículo (o matriz) 9×9 *que representa la solución de algún juego de Sudoku, es decir, satisface que*

(i) en cada fila aparecen todos los números del 1 *al* 9*,*

(ii) en cada columna aparecen todos los números del 1 *al* 9*,*

(iii) está dividido en 3×3 *subretículos de tamaño* 3×3*,*

(iv) en cada subretículo aparecen todos los números del 1 *al* 9*.*

Construir un lenguaje $\mathcal{L}$ *con el cual se puedan interpretar a los cuadrados Sudoku como* $\mathcal{L}$*-estructuras y demostrar que bajo* $\mathcal{L}$ *el conjunto Sudoku*$= \{\mathcal{S} : \mathcal{S}$*es un cuadrado Sudoku*$\}$ *es definible.*

Solución 6.24.
Primero debemos analizar cómo podemos interpretar un cuadrado Sudoku como una estructura sobre un lenguaje de primer orden $\mathcal{L}$*. En este sentido necesitamos pensar en estructuras más generales, por ejemplo matrices de orden* 9×9 *con entradas en* $[9] = \{1, \ldots, 9\}$*.*

La idea es usar este dominio (universo) para determinar qué tipos de relaciones necesitamos con las que podamos describir todas las cosas requeridas en los puntos $(i), (ii), (iii)$ *y* (iv) *del enunciado.*

Recordando álgebra lineal una matriz con entradas numéricas es básicamente una función binaria $f : [m] \times [n] \to \mathcal{N}$*, donde* $m, n \in \mathbb{N}$ *y* $\mathcal{N}$ *es un conjunto no vacío de números. En nuestro caso, debemos fijar* $m = n = 9$ *y* $\mathcal{N} = [9]$ *para obtener números del* 1 *al* 9 *en cada fila y columna de nuestra estructura, donde la fila se interpreta como la entrada de la primera componente de* f *y la columna como la segunda componente. Es decir, que nuestro lenguaje* $\mathcal{L}$ *debe tener un símbolo de función binario* f*.*

Tener el universo $[9]$ *no implica que esté ordenado, pero sería bueno que así fuera para identificar en nuestro universo cuál elemento es el primero, el segundo y así hasta el noveno, por lo que agregaremos a* $\mathcal{L}$ *la relación binaria* $<$ *que se interpretará de la manera usual. Notar que con esta relación todo elemento de* $[9]$ *es distinguible. Denotamos por* $\varphi_i(x)$ *a la fórmula de primer orden con una variable libre que distingue al elemento* $i \in [9]$*.*

Necesitamos identificar también cuáles coordenadas pertenecen al subretículo 1*, al subretículo* 2*, así hasta el subretículo* 9*. Básicamente, esto es una relación entre coordenadas y números, por lo que incluimos en* $\mathcal{L}$ *la relación ternaria* R*, donde* $R(x, y, z)$ *se interpreta como 'la coordenada* (x, y) *se encuentra en el subretículo* z*'.*

Así, $\mathcal{L} = \langle f, R, <, = \rangle$ *y un cuadrado Sudoku es particularmente una* $\mathcal{L}$*-estructura.*

Práctica 7

Sistemas Deductivos, Completitud y Compacidad para Lógica de Primer Orden

Ejercicio 7.1. *Demostrar que MP preserva validez para toda clase de modelos. Es decir que, si* $\mathcal{C} \models (\alpha \to \beta)$ *y* $\mathcal{C} \models \alpha$ *entonces* $\mathcal{C} \models \beta$.

Solución 7.1.

Ejercicio 7.2.
Sea $\Delta = \{SQ1, \ldots, SQ7\}$ *el conjunto de todos los axiomas de SQ.*

a. *Supongamos que agregamos a* Δ *una fórmula* φ *que no es universalmente válida. Mostrar que el sistema resultante no es correcto con respecto a la clase de todos los modelos.*

b. *Yendo al otro extremo, supongamos que eliminamos todos los axiomas, esto es,* $\Delta = \emptyset$. *Mostrar que el sistema resultante no es completo con respecto a la clase de todos los modelos.*

c. *Supongamos que agregamos a* Δ *una nueva fórmula universalmente válida* φ. *Explicar por qué el sistema resultante es correcto y completo con respecto a la clase de todos los modelos.*

Solución 7.2.

a. *Sabemos que* $SQ1, \ldots, SQ7$ *son verdaderos para todos los modelos y, por hipótesis,* φ *no es universalmente válida por lo tanto existe un modelo* $\mathcal{M}$ *tal que hace falsa a* φ.

   Luego, para este modelo $\mathcal{M}$ *se tiene que* $\Delta \cup \{\varphi\}$ *es falso y, por lo tanto, el sistema resultante no es correcto con respecto a la clase de todos los modelos.*

b. *Es claro que si se considera cualquier fórmula* φ *que no sea universalmente válida, entonces como no hay modelos en la clase* Δ, *todos los modelos de esta clase satisfacerán a* φ *pero ésta no es demostrable por SQ por correctitud.*

   Luego SQ no es completa respecto de Δ.

c.

Ejercicio 7.3.
Decidir si cada una de las siguientes afirmaciones es verdadera o falsa y justificar

a. *Si* $\mathcal{C}_1$ *y* $\mathcal{C}_2$ *son dos clases de modelos tal que* Γ *es una axiomatización completa respecto de* $\mathcal{C}_1$

   1. *Si* $\mathcal{C}_1 \subseteq \mathcal{C}_2$, *entonces* Γ *es completa respecto de* $\mathcal{C}_2$.
   2. *Si* $\mathcal{C}_1 \nsubseteq \mathcal{C}_2$, *entonces* Γ *es completa respecto de* $\mathcal{C}_2$.

b. *Sea* $\mathcal{S}$ *un sistema axiomático de primer orden correcto y completo con respecto a una clase de* $\mathcal{L}$*-estructuras* $\mathcal{C}$ *y sean* Γ *un conjunto de fórmulas consistente (en el sentido usual, con respecto a SQ) y una subclase* $\mathcal{C}' \models \Gamma$.

   1. $\mathcal{S} \cup \Gamma$ *es completo con respecto a* $\mathcal{C}$.
   2. $\mathcal{S} \cup \Gamma$ *es correcto con respecto a* $\mathcal{C}$.
   3. *Si* $\mathcal{C} \models \Gamma$, $\mathcal{S} \cup \Gamma$ *es correcto con respecto a* $\mathcal{C}$.
   4. $\mathcal{S}$ *es correcto con respecto a* $\mathcal{C}'$.
   5. $\mathcal{S}$ *es completo con respecto a* $\mathcal{C}'$.

Solución 7.3.

a. 1. ***Verdadero.***

Sabemos que $\mathcal{M} \models \varphi$ para todo $\mathcal{M} \in \mathcal{C}_1$ entonces $\Gamma \vdash \varphi$ y queremos ver que $\mathcal{M} \models \varphi$ para todo $\mathcal{M} \in \mathcal{C}_2$ entonces $\Gamma \vdash \varphi$.

Supongamos entonces $\mathcal{M} \models \varphi$ para todo $\mathcal{M} \in \mathcal{C}_2$. Luego, como $\mathcal{C}_1 \subseteq \mathcal{C}_2$, $\mathcal{M} \models \varphi$ para todo $\mathcal{M} \in \mathcal{C}_1$ y, por lo tanto, $\Gamma \vdash \varphi$ como queríamos ver.

2. ***Falso.***

Un posible contraejemplo es $\Gamma = SQ$, $\mathcal{C}_1$ la clase de todos los modelos, $\mathcal{C}_2 = \emptyset$ ya que $\mathcal{C}_1 \nsubseteq \mathcal{C}_2$, SQ es completa respecto de la clase de todos los modelos y no es completa respecto de la clase vacía.

Para demostrar esto último basta con considerar cualquier fórmula φ que no sea universalmente válida, como no hay modelos en la clase $\mathcal{C}_2$, todos los modelos de esta clase satisfacerán a φ pero ésta no es demostrable por SQ por correctitud.

Luego SQ no es completa respecto de $\mathcal{C}_2$.

b. 1. ***Verdadero.***

Supongamos $\mathcal{C} \models \varphi$. Como $\mathcal{S}$ es completo con respecto a $\mathcal{C}$, entonces $\vdash_{\mathcal{S}} \varphi$. Pero como $\mathcal{S} \subseteq \mathcal{S} \cup \Gamma$, ésto implica que $\vdash_{\mathcal{S} \cup \Gamma} \varphi$.

2. ***Falso.***

Tomemos por ejemplo $\mathcal{S} = SQ$ y $\mathcal{C}$ la clase de todos los modelos. En $\mathcal{L}$ debe haber por lo menos un símbolo de relación R. Tomemos entonces $\Gamma = \{(\forall x)\ xRx\}$. $\mathcal{S}$ es correcto y completo respecto a $\mathcal{C}$, Γ es consistente (pues es satisfacible) pero $\mathcal{S} \cup \Gamma$ no es correcto respecto a $\mathcal{C}$ (tomar la fórmula $\varphi = (\forall x)\ xRx$ entonces hay estructuras que la satisfacen y otras que no).

3. ***Verdadero.***

Supongamos $\vdash_{\mathcal{S} \cup \Gamma} \varphi$. Como $\mathcal{C} \models \Gamma$ y $\mathcal{S}$ es completo respecto a $\mathcal{C}$, $\vdash_{\mathcal{S}} \Gamma$ (demuestra todas fórmulas de Γ). Luego a una demostración de φ que use las fórmulas de Γ la podemos obtener a partir de $\mathcal{S}$, obteniendo que $\vdash_{\mathcal{S}} \varphi$. Finalmente, como $\mathcal{S}$ es correcto respecto a $\mathcal{C}$, tenemos que $\mathcal{C} \models \varphi$.

4. ***Verdadero.***

Como $\mathcal{S}$ es correcto con respecto a $\mathcal{C}$ ocurre que si $\mathcal{S} \vdash \varphi$ entonces φ es válido en cualquier modelo $\mathcal{C}$. En particular sigue siendo válido en cualquier modelo $\mathcal{C}'$. Así que $\mathcal{S}$ es correcto con respecto a $\mathcal{C}'$.

5. ***Falso.***

Consideremos $\mathcal{L} = \{=\}$, $\mathcal{S} = SQ$ y $\mathcal{C}$ la clase de todas las $\mathcal{L}$-estructuras.

Sea C' la subclase de todos los modelos que poseen al menos dos elementos en su dominio. Luego,

$$C' \models \exists xy(x \neq y)$$

Como $\exists xy(x \neq y)$ no es una tautología no es cierto que se puede deducir a partir de $\mathcal{S}$. Por lo tanto $\mathcal{S}$ no es completo con respecto a C'.

Ejercicio 7.4.

Se dice que un modelo de primer orden es transitivo cuando todas sus relaciones binarias son transitivas. Partiendo de la axiomatización para SQ, proponer una extensión SQ^T que caracterice la clase de modelos transitivos.

a. Demostrar que SQ^T es correcta con respecto a la clase de modelos transitivos.

b. Demostrar que SQ^T es completa con respecto a la clase de todos los modelos y, en particular, de los modelos transitivos.

c. Demostrar que SQ^T no es correcta con respecto a la clase de todos los modelos.

Solución 7.4.

Idea: *Buscamos una axiomatización que nos de como teoremas todas las fórmulas verdaderas en la clase de modelos transitivos y sólo las fórmulas verdaderas en la clase de modelos transitivos.*

Sabemos que SQ es correcta tanto para todos los modelos (sean o no transitivos).

Si agregamos fórmulas a SQ que fuercen a que la interpretación de los símbolos de relación binaria se haga con relaciones transitivas, restringimos la clase de modelos para la cual nuestra axiomatización es correcta, a la clase de modelos transitivos.

Proponemos extender SQ con el siguiente esquema axiomático:

$$\boldsymbol{SQ}_R^T \qquad (\forall x)(\forall y)(\forall z)((R(x,y) \wedge R(y,z)) \to R(x,z)) \qquad (R \text{ símbolo de relación binaria})$$

Entones $SQ^T = SQ + \boldsymbol{SQ}_R^T$.

a. *Sea $\mathcal{L}$ un lenguaje de primer orden. Demostremos que SQ^T es correcta con respecto a $\mathcal{C}$, la clase de $\mathcal{L}$-estructuras cuyas relaciones binarias son todas transitivas.*

Para ello vamos a demostrar, por inducción en n, el siguiente predicado $P(n)$:

$$P(n): \forall \mathcal{M} \in \mathcal{C} \text{ y } \varphi \in \mathbf{Form}(\mathcal{L}),\ \text{si } \varphi_1, \ldots, \varphi_n = \varphi \text{ es una derivación de } \varphi \text{ en } SQ^T,\ \text{entonces } \forall 1 \leq i \leq n \quad \mathcal{M} \models \varphi_i$$

$\underline{P(1)}$: *Tenemos 2 posibilidades para φ_1:*

- *φ_1 es un axioma de SQ. Entonces, como SQ es correcta con respecto a la clase de todos los modelos, $\mathcal{M} \models \varphi_1$ para todo $\mathcal{M} \in \mathcal{C}$.*
- *$\varphi_1 = (\forall x)(\forall y)(\forall z)((R(x,y) \wedge R(y,z)) \to R(x,z))$ para algún R símbolo de relación binaria en $\mathcal{L}$. Queremos ver que para todo $\mathcal{M} \in \mathcal{C}$ y toda valuación v, $\mathcal{M} \models \varphi_1[v]$.*

$$\begin{aligned} \mathcal{M} \models (\forall x)(\forall y)(\forall z)((R(x,y) \wedge R(y,z)) \to R(x,z))[v] &\Leftrightarrow \text{ para todo } a,b,c \in U_{\mathcal{M}} \\ \mathcal{M} \models ((R(x,y) \wedge R(y,z)) \to R(x,z))[v(x=a, y=b, z=c)] &\Leftrightarrow \text{ para todo } a,b,c \in U_{\mathcal{M}} \\ (a,b) \notin R^{\mathcal{M}} \text{ o } (b,c) \notin R^{\mathcal{M}} \text{ o } (a,c) \in R^{\mathcal{M}} &\Leftrightarrow R^{\mathcal{M}} \text{ es transitiva} \end{aligned}$$

$\underline{P(n+1)}$: *Nuestra hipótesis inductiva es que vale $P(k)$ para todo $k \leq n$.*
Tenemos tres posibilidades para φ_n:

- *φ_{n+1} es un axioma de SQ. (idem que en $P(1)$).*
- *$\varphi_{n+1} = (\forall x)(\forall y)(\forall z)((R(x,y) \wedge R(y,z)) \to R(x,z))$. (idem que en $P(1)$).*
- *φ_{n+1} se obtiene por MP de φ_i y φ_j (con $i,j \leq n$). Por HI, para todo $\mathcal{M} \in \mathcal{C}$, $\mathcal{M} \models \varphi_i$ y $\mathcal{M} \models \varphi_j$. Por lo tanto, como MP preserva validez, $\mathcal{M} \models \varphi_{n+1}$.*

b. *Ahora probemos que SQ^T es completa con respecto a $\mathcal{C}$, la clase de todos modelos.*

Sea $\mathcal{L}$ un lenguaje de primer orden y sea $\varphi \in \mathbf{Form}(\mathcal{L})$ universalmente válida.

$$\begin{aligned} &\Rightarrow \vdash_{SQ} \varphi && (\text{por completitud de } SQ \text{ con respecto a } \mathcal{C}) \\ &\Rightarrow \vdash_{SQ^T} \varphi && (SQ \subseteq SQ^T) \end{aligned}$$

c. *Ahora probemos que SQ^T no es correcta con respecto a $\mathcal{C}$, la clase de todos modelos.*

Sea $\mathcal{L}$ un lenguaje de primer orden con un símbolo de relación binario R y sea $\mathcal{M} = \langle \{1,2,3\}, R^{\mathcal{M}} \rangle \in \mathcal{C}$ con $R^{\mathcal{M}} = \{(1,2),(2,3)\}$.

Sea

$$\varphi = \boldsymbol{SQ}_R^T = (\forall x)(\forall y)(\forall z)((R(x,y) \wedge R(y,z)) \to R(x,z))$$

Entonces $\vdash_{SQ^T} \varphi$ pero $\mathcal{M} \not\models \varphi$. Por lo tanto, SQ^T no es correcta con respecto a $\mathcal{C}$.

Ejercicio 7.5.
Decimos que una estructura de primer orden es de equivalencia si todas sus relaciones binarias son de equivalencia. Sea $\mathcal{L} = \{R\}$ un lenguaje con un símbolo de predicado binario.

a. *Proponer una axiomatización SQ_{equiv} que extienda a SQ y que sea correcta y completa con respecto a la clase $\mathcal{C}_e$ de todas las $\mathcal{L}$-estructuras de equivalencia.*

b. *Demostrar que la axiomatización propuesta en el ítem anterior es completa pero no correcta con respecto a la clase de <u>todas</u> las $\mathcal{L}$-estructuras.*

Solución 7.5.

a. *Extendemos a SQ con los siguientes axiomas:*

$$\begin{aligned} &\boldsymbol{SE1} \quad (\forall x)(xRx) \\ &\boldsymbol{SE2} \quad (\forall x)(\forall y)(xRy \to yRx) \\ &\boldsymbol{SE3} \quad (\forall x)(\forall y)(\forall z)((xRy \wedge yRz) \to xRz) \end{aligned}$$

Para probar la correctitud, tenemos que ver que para cualquier $\mathcal{L}$-fórmula φ, vale $\vdash_{SQ_{equiv}} \varphi$ implica $\mathcal{C}_e \models \varphi$. Supongamos $\vdash_{SQ_{equiv}} \varphi$. Por definición de $\vdash_{SQ_{equiv}}$, esto significa que $\{SE1, SE2, SE3\} \underbrace{\vdash}_{\vdash_{SQ}} \varphi$. Luego, por correctitud fuerte de SQ, tenemos que $\{SE1, SE2, SE3\} \models \varphi$.

Ahora, si probamos que toda $\mathcal{L}$-estructura en $\mathcal{C}_e$ satisface $\{SE1, SE2, SE3\}$ (es decir que $\mathcal{C}_e \models \{SE1, SE2, SE3\}$) por definición de la consecuencia semántica $\models$ tendríamos que $\mathcal{C}_e \models \varphi$.

Veamos la demostración formal de que $\mathcal{C}_e \models SE2$, las otras dos quedan de ejercicio.

Sea $\mathcal{M} \in \mathcal{C}_e$ y sea v una valuación. Tenemos que

$\mathcal{M}, v \models \forall x \forall y(xRy \to yRx)$ $\qquad$ $(\mathcal{M}, v \models SE2)$

si y sólo si para todo $a \in U_\mathcal{M}$ vale que $\mathcal{M}, v[x \mapsto a] \models \forall y(xRy \to yRx)$

si y sólo si para todo a y para todo $b \in U_\mathcal{M}$ tenemos que $\mathcal{M}, v[x \mapsto a, y \mapsto b] \models (xRy \to yRx)$

si y sólo si para todo a y para todo $b \in U_\mathcal{M}$ no vale $\mathcal{M}, v[x \mapsto a, y \mapsto b]xRy$ o vale $\mathcal{M}, v[x \mapsto a, y \mapsto b]xRy$

si y sólo si ($\mathcal{M}$ estructura de equivalencia), para todo a y para todo $b \in U_\mathcal{M}$ no vale $aR^\mathcal{M}b$ o vale $bR^\mathcal{M}a$

Para ver la completitud, supongamos $\mathcal{C}_e \models \varphi$. Como cualquier estructura que satisface $\{SE1, SE2, SE3\}$ es de equivalencia, y como cualquier estructura de equivalencia satisface φ, tenemos en particular que

$$\{SE1; SE2; SE3\} \models \varphi.$$

Por lo tanto, por completitud fuerte de SQ, $\{SE1, SE2, SE3\} \vdash \varphi$, que es lo mismo que decir $\vdash_{SQ_{equiv}} \varphi$.

b. La completitud sale de la observación, pero también podemos verlo así: si $\mathcal{C}$ es la clase de todas las estructuras y $\mathcal{C} \models \varphi$, entonces por completitud de SQ vale $\vdash_{SQ} \varphi$, lo cual implica trivialmente que $\{SE1, SE2, SE3\} \vdash_{SQ} \varphi$, que es lo mismo que decir $\vdash_{SQ \cup \{SE1, SE2, SE3\}} \varphi$, o sea $\vdash_{SQ_{equiv}} \varphi$.

Para probar la no correctitud, supongamos en cambio que esta axiomatización es correcta. Entonces, como trivialmente $\vdash_{SQ_{equiv}} SE1$, la fórmula $SE1$ (reflexividad) debería ser verdadera en toda $\mathcal{L}$-estructura, lo cual no es cierto.

Ejercicio 7.6.
Sea un lenguaje de primer orden con igualdad, un símbolo de predicado binario P y un símbolo de constante r. Sea la clase de modelos

$$\mathcal{C} = \{\mathcal{M} \mid P^\mathcal{M} \text{ define un árbol con raíz } r \text{ sobre todos los elementos de } \mathcal{M}\}$$

donde por árbol se entiende cualquier árbol dirigido donde cada nodo puede tener una cantidad arbitraria de hijos ($P^\mathcal{M}(x, y)$ afirma que $x^\mathcal{M}$ es el padre de $y^\mathcal{M}$). Considerar la axiomatización SQ_{Tree}, que extiende a la axiomatización SQ vista en clase de la siguiente manera:

$$\begin{array}{rl} \textbf{\textit{SQ8}} & (\forall x)\big(\neg P(x, r) \land \neg P(x, x)\big) \\ \textbf{\textit{SQ9}} & (\forall x)(\forall y)(\forall z)\big((P(y, x) \land P(z, x)) \to (y = z)\big) \\ \textbf{\textit{SQ10}} & (\forall x)\big(((\forall y)\neg P(y, x)) \to x = r\big) \end{array}$$

Suponiendo que SQ es correcta y completa con respecto a la clase de todos los modelos:

a. Demostrar que los axiomas $SQ8, SQ9$ y $SQ10$ son válidos en $\mathcal{C}$.

b. Sea $\varphi = (\forall x)(\forall y)\,(P(x, y) \to \neg P(y, x))$. Demostrar que φ es válida en $\mathcal{C}$, y existe un modelo $\mathcal{M}$ tal que todos los axiomas de SQ_{Tree} son válidos en $\mathcal{M}$, pero $\mathcal{M} \not\models \varphi$.

c. Analizar si φ es válida en $\mathcal{C}$, y si junto con los puntos anteriores se puede afirmar que SQ_{Tree} es correcta pero no es completa con respecto a $\mathcal{C}$.

Solución 7.6.

a. Para ver que los axiomas son válidos en la clase $\mathcal{C}$ primero interpretemos que significarían las fórmulas interpretadas en un modelo de la clase $\mathcal{C}$.

$$\begin{array}{rl} \textbf{\textit{SQ8}} & \textit{Los nodos son irreflexivos y ningún nodo tiene a la raíz como hijo.} \\ \textbf{\textit{SQ9}} & \textit{Los nodos no comparten hijos.} \\ \textbf{\textit{SQ10}} & \textit{El único nodo sin padre es la raíz } r. \end{array}$$

Todas estas propiedades suenan razonables para un modelo de la clase, probemos formalmente que valen.

Si probamos que para cualquier modelo de la clase y cualquier valuación, los axiomas son verdaderos entonces habremos probado que son válidos en la clase.

Fijado un modelo arbitrario $\mathcal{M} \in \mathcal{C}$ y una valuación v, veamos axioma por axioma,

SQ8. $\mathcal{M}, v \models (\forall x)(\neg P(x, r) \land \neg P(x, x))$ *si y sólo si*

$$\mathcal{M}, v[x \mapsto a] \models (\neg P(x, r) \land \neg P(x, x))$$

para cualquier $a \in |\mathcal{M}|$. Y esto ocurre si y sólo si

$$\mathcal{M}, v[x \mapsto a] \models \neg P(x,r) \text{ y } \mathcal{M}, v[x \mapsto a] \models \neg P(x,x)$$

para cualquier $a \in U_{\mathcal{M}}$. Siguiendo las definiciones semánticas, lo anterior sucede si y sólo si

$$\mathcal{M}, v[x \mapsto a] \not\models P(x,r) \text{ y } \mathcal{M}, v[x \mapsto a] \not\models P(x,x)$$

para cualquier $a \in U_{\mathcal{M}}$. Finalmente llegamos a que debe suceder

$$(a, r^M) \notin P^{\mathcal{M}} \text{ y } (a,a) \notin P^{\mathcal{M}}$$

para cualquier $a \in U_{\mathcal{M}}$.

Es muy importante llegar hasta esta instancia para luego poder justificar la validez de la fórmula. Al principio sólo tenemos una fórmula, y una fórmula es solamente una serie de símbolos. Recién en esta instancia tenemos una (o más) condición sobre una relación (que es un elemento matemático) y sabemos que esta relación se interpretará como dicta la definión de $\mathcal{C}$.

Ahora podemos decir que, en la clase $\mathcal{C}$, todo elemento es irreflexivo así que, en particular para un a cualquiera sucede que $(a,a) \notin U_{\mathcal{M}}$. De igual manera justificamos que, al ser r la raíz, no puede tener padre, por lo tanto $(a, r^M) \notin U_{\mathcal{M}}$. Concluimos que $SQ8$ es válido en la clase.

SQ9. *En este punto veremos un ejemplo del tipo de cosas que pueden "saltearse con justi*
cación adecuada". Siempre usen las definiciones de satisfacción hasta llegar a las condiciones sobre elementos del modelo para luego poder justificar su validez.

$$\mathcal{M}, v \models (\forall x)(\forall y)(\forall z)((P(y,x) \wedge P(z,x)) \to (y = z))$$

si y sólo si

$$\mathcal{M}, v[x \mapsto a, y \mapsto b, z \mapsto c] \models (P(y,x) \wedge P(z,x)) \to (y = z)$$

para elementos arbitrarios $a, b, c \in U_{\mathcal{M}}$. Para ahorrar espacio llamemos a una valuación $v' = v[x \mapsto a, y \mapsto b, z \mapsto c]$. La última fórmula equivale a

$$\mathcal{M}, v' \not\models P(y,x) \wedge P(z,x) \text{ o } \mathcal{M}, v' \models (y = z)$$

para elementos arbitrarios $a, b, c \in U_{\mathcal{M}}$. Si y sólo si

$$\mathcal{M}, v' \not\models P(y,x) \text{ o } \mathcal{M}, v' \not\models P(z,x) \text{ o } \mathcal{M}, v' \models (y = z)$$

para elementos arbitrarios $a, b, c \in U_{\mathcal{M}}$. Si y sólo si

$$(b,a) \notin P^{\mathcal{M}} \text{ o } (c,a) \notin P^{\mathcal{M}} \text{ o } b = c$$

para elementos arbitrarios $a, b, c \in U_{\mathcal{M}}$. Ahora que tenemos una condición sobre los elementos del modelo tenemos que probar que se cumple.

Como tengo 3 disyunciones, en lugar de ver que siempre se cumple alguna de las 3 voy a suponer que no se cumple ninguna y ver que eso no es posible.

Eso quiere decir que tengo dos elementos distintos b, c tal que ambos tienen como hijo al elemento a. En un árbol eso no puede suceder. Absurdo.

SQ10. *Procedamos de igual forma.*

$$\mathcal{M}, v \models (\forall x)(((\forall y)\neg P(y,x)) \to (x = r)$$

si y sólo si

$$\mathcal{M}, v[x \mapsto a] \models ((\forall y)\neg P(y,x)) \to (x = r)$$

para elementos arbitrarios $a, b, c \in U_{\mathcal{M}}$. Llamemos a una valuación $v' = v[x \mapsto a, y \mapsto b, z \mapsto c]$. La última fórmula equivale a

$$\mathcal{M}, v' \not\models ((\forall y)\neg P(y,x)) \text{ o } \mathcal{M}, v' \models (x = r)$$

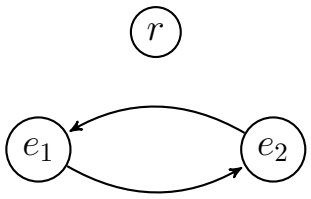
para $a \in U_{\mathcal{M}}$ arbitrario. Ahora, esto último ocurre si y sólo si $b \in |\mathcal{M}|$ tal que

$$\mathcal{M}, v'' \models P(y,x) \text{ o } \mathcal{M}, v'' \models (x = r)$$

donde $v'' = v'[y \mapsto b]$. Y esto último se cumple si y sólo si para todo $a \in U_{\mathcal{M}}$ se tiene que $a = r$, o existe algún $b \in U_{\mathcal{M}}$ tal que $P^{\mathcal{M}}(b,a)$. Es decir, cualquier nodo de $\mathcal{M}$ o es la raíz o tiene un padre.

b. La fórmula $\varphi = (\forall x)(\forall y)(P(x,y) \to \neg P(y,x))$ indica que la relación es antisimétrica. Por lo tanto debemos concentrarnos en dar un modelo donde la relación no sea antisimétrica, de todas maneras no debemos perder de vista que queremos que cumpla con todos los axiomas. Este es el punto en el que debemos notar que "algo" le falta a la axiomatización. Nosotros pensamos en árboles antisimétricos pero evidentemente nuestros axiomas no lo están forzando.

Consideremos el siguiente modelo $\mathcal{M}$:



Debemos mostrar que todos los axiomas son válidos en el modelo. >Debemos hacer todo ese trabajo infernal nuevamente? No. Gracias al análisis que ya hicimos sobre las fórmulas ya sabemos que condiciones imponen sobre el modelo y podemos partir directamente de ahí.

Veamos que

$$\textbf{SQ8.} \qquad (a, r^M) \notin P^{\mathcal{M}} \text{ y } (a,a) \notin P^{\mathcal{M}}$$

vale en el modelo para cualquier a. No hay elementos reflexivos y tampoco hay elementos que tengan a r como hijo. Vale.

Veamos también que

$$\textbf{SQ9.} \qquad (b,a) \notin P^{\mathcal{M}} \text{ o } (c,a) \notin P^{\mathcal{M}} \text{ o } b = c$$

vale en el modelo para cualquier a, b, y c. No hay elementos con dos hijos. Vale.

*Veamos que **SQ10.** vale en el modelo para cualquier a, b. Todos tienen padre menos r. Vale.*

El paso siguiente es ver que φ no vale en el modelo. Queda como ejercicio ver que para la valuación donde $v(x) = e_1$, $v(y) = e_2$ la fórmula es falsa.

c. En el primer inciso probamos que todos los axiomas de SQ_{Tree} son válidos en $\mathcal{C}$. También sabemos que MP preserva validez, ya que, si $\mathcal{C} \vdash \alpha \to \beta$, $\mathcal{C} \vdash \alpha$ entonces $\mathcal{C} \vdash \beta$. Juntando estas dos cosas probamos que si $SQ_{Tree} \vdash \varphi$ entonces $\mathcal{C} \models \varphi$, o sea, que el sistema axiomático es correcto con respecto a la clase $\mathcal{C}$.

Todo árbol con raíz es antisimétrico por lo tanto φ valdrá en todo modelo donde P sea interpretada como un árbol.

Por otra parte, notemos que como todos nuestros axiomas son válidos en el modelo $\mathcal{M}$ y modus ponens preserva validez, todo teorema de esta teoría será válido en $\mathcal{M}$. Escrito de otra manera, si $SQ_{Tree} \vdash \varphi$ entonces $\mathcal{M} \models \varphi$.

También probamos que φ es válida en la clase, entonces nos gustaría que fuera un teorema de nuestra teoría pero, como $\mathcal{M} \not\models \varphi$, usando el contrarecíproco de la observación anterior llegamos a que $SQ_{Tree} \nvdash \varphi$.

Por lo tanto tenemos una fórmula φ (la de la antisimetría) tal que $\mathcal{C} \models \varphi$ (es válida en la clase) y $SQ_{Tree} \nvdash \varphi$ (no es teorema). Concluimos que la teoría es incompleta respecto a $\mathcal{C}$.

Ejercicio 7.7. *Sea un lenguaje de primer orden con dos símbolos de predicado binarios P y T. Sea la clase de modelos*

$$\mathcal{C} = \{\mathcal{M} \mid P_{\mathcal{M}} \text{ y } T_{\mathcal{M}} \text{ son relaciones binarias, y } P^+_{\mathcal{M}} = T_{\mathcal{M}}\}$$

en donde $P^+_{\mathcal{M}}$ representa la clausura transitiva de $P_{\mathcal{M}}$ (i.e. $P^+_{\mathcal{M}}$ es la mínima relación transitiva tal que $P_{\mathcal{M}} \subseteq P^+_{\mathcal{M}}$). Considerar la axiomatización SQ^+, que extiende la axiomatización SQ vista en clase de la siguiente manera:

$$\begin{array}{ll} \textbf{SQ8} & (\forall x)(\forall y)\big(P(x,y) \to T(x,y)\big) \\ \textbf{SQ9} & (\forall x)(\forall y)(\forall z)\big((T(x,y) \wedge P(y,z)) \to T(x,z)\big) \\ \textbf{SQ10} & (\forall x)(\forall y)\big((T(x,y) \wedge \neg P(x,y)) \to (\exists z)(T(x,z) \wedge P(z,y))\big) \end{array}$$

Sabiendo que SQ es correcta y completa con respecto a la clase de todos los modelos:

a. Demostrar que los axiomas $SQ8, SQ9$ y $SQ10$ son válidos en $\mathcal{C}$. ¿Se puede concluir a partir de ésto que SQ^+ es correcta con respecto a $\mathcal{C}$? Justificar brevemente.

b. Sea $\varphi = (\forall x)(\forall y)\big(T(x,y) \to ((\exists z)P(x,z))\big)$. Mostrar que existe un modelo $\mathcal{M}$ en donde todos los axiomas de SQ^+ son válidos, pero $\mathcal{M} \not\models \varphi$.

c. Suponiendo que φ es válida en $\mathcal{C}$, demostrar que SQ^+ no es completa con respecto a $\mathcal{C}$.

Solución 7.7.

Ejercicio 7.8. *Considerar un lenguaje de primer orden igualdad, un símbolo de función binario + y dos constantes 0 y 1. Sea P la axiomatización que extiende a SQ con:*

$$\begin{array}{ll} \boldsymbol{P1} & (\forall x)(\neg(0 = x + 1)) \\ \boldsymbol{P2} & (\forall x)(\forall y)((x + 1 = y + 1) \to (x = y)) \\ \boldsymbol{P3} & (\forall x)(\forall y)((x + y) + 1 = x + (y + 1)) \end{array}$$

Demostrar que P no es completa con respecto al modelo de los naturales con la suma.

Solución 7.8.

Ejercicio 7.9.
Sea $\mathcal{L} = \{c, f, R, =\}$ un lenguaje de primer orden con igualdad, donde c es un símbolo de constante, f es un símbolo de función unaria y R es un símbolo de predicado binario.

Sea $\mathcal{Z} = \{\mathbb{Z}, 0, |\cdot|, <\}$ la estructura de los números enteros con la función 'valor absoluto' y con la relación de orden estricto usual. Consideremos SQ_Z, la axiomatización que extiende a $SQ^=$ con los siguientes axiomas:

$$\begin{array}{ll} \boldsymbol{Z1} & (\forall x)(f(x) = x \to \neg(xRf(x))) \\ \boldsymbol{Z2} & (\forall x)(\neg(f(x) = x) \to xRf(x)) \\ \boldsymbol{Z3} & (\forall x)(x \neq c \to \exists y(x \neq y \wedge f(x) = f(y) \wedge \forall z(f(z) = f(x) \to (z = x \vee z = y)))) \end{array}$$

a) Probar que SQ_Z es correcta respecto a $\mathcal{Z}$.

b) Demostrar SQ_Z no es completa respecto a $\mathcal{Z}$.

Solución 7.9.

a) Supongamos $\vdash_{SQ_Z} \varphi$. Esto significa que $\{Z1, Z2, Z3\} \vdash_{SQ^=} \varphi$, que por correctitud fuerte de $SQ^=$ implica que $\{Z1, Z2, Z3\} \models \varphi$. Ahora podemos ver que $\mathcal{Z}$ satisface $Z1, Z2$ y $Z3$ interpretándolas semánticamente, y obtenemos que $\mathcal{Z} \models \varphi$.

b) Para probar la no completitud, busquemos una fórmula ψ y un modelo $\mathcal{M}$ de SQ_Z tal que $\mathcal{M} \not\models \psi$ pero $\mathcal{Z} \models \psi$. Una posibilidad es tomar $\mathcal{M} = \langle\{0\}, 0, id, \emptyset, =\rangle$ (ver que aquí son verdaderas $Z1, Z2, Z3$) y $\psi : \exists x \exists y(x \neq y)$.

Ahora, vemos que $\mathcal{Z} \models \psi$, ya que $\mathbb{Z}$ tiene al menos dos elementos distintos. Si valiese que $\vdash_{SQ_Z}$, tendríamos que $\{Z1, Z2, Z3\} \vdash \psi$, lo cual por correctitud fuerte de $SQ^=$ implica que $\{Z1, Z2, Z3\} \models \psi$, pero esto no es cierto por el $\mathcal{M}$ que construimos antes, que cumplía $\mathcal{M} \models \{Z1, Z2, Z3\}$ pero $\mathcal{M} \not\models \psi$. Luego $\not\vdash_{SQ_Z}$ y por lo tanto SQ_Z no es completo respecto a $\mathcal{Z}$.

Ejercicio 7.10. *Sea $\mathcal{L}$ un lenguaje con igualdad.*

1. *Dar un conjunto de fórmulas Γ tal que si Γ es satisfacible en un modelo $\mathcal{M}$ entonces el dominio de $\mathcal{M}$ sea infinito.*

   Sugerencia: escribir una fórmula que, dado un n fijo, fuerce a que el modelo tenga al menos n elementos.

2. *Usando compacidad y el ítem anterior, demostrar que no existe ninguna fórmula φ tal que φ es satisfacible en un modelo $\mathcal{M}$ si y sólo si el dominio de $\mathcal{M}$ es finito. En otras palabras, demostrar que la clase de las estructuras con dominio finito no es definible en primer orden.*

Solución 7.10.

Ejercicio 7.11.
Demuestre que son equivalentes los siguientes enunciados, donde Δ es un conjunto de fórmulas de primer orden cualesquiera:

(a) Para toda fórmula de primer orden φ, $\Delta \models \varphi$ implica $\Delta \vdash \varphi$.

(b) Si Δ es consistente, entonces tiene un modelo

Solución 7.11.

$\underline{(a) \Rightarrow (b)}$: *Procedamos por Reducción al absurdo: supongamos que Δ es consistente y que no tiene modelos. Sea φ una fórmula de primer orden tal que $\Delta \models \varphi$. Es "claro" que $\Delta \models \neg\varphi$. En efecto, de lo contrario habrá un modelo $\mathcal{A}$ y una valuación v que satisface Δ y tal que $\mathcal{A}, v \models \neg\varphi$, pero hemos supuesto que Δ no tiene modelos.*

Por (a), tenemos que $\Delta \vdash \varphi$ y$\Delta \vdash \neg\varphi$, o en otras palabras, Δ es inconsistente. Contradicción.

$\underline{(a) \Leftarrow (b)}$: *Sea φ tal que $\Delta \models \varphi$. Sabemos entonces que cualquier modelo de Δ no satisface $\neg\varphi$ y por lo tanto el conjunto $\Delta' = \Delta \cup \{\neg\varphi\}$ no tiene modelos. Por (el contrarrecíproco) de (b), tenemos que el conjunto de fórmulas de primer orden Δ' es inconsistente. De donde, $\Delta \vdash \varphi$.*

En los siguientes ejercicios todos los lenguajes se consideran con igualdad.

Ejercicio 7.12. *Sea $\mathcal{L}$ un lenguaje con un símbolo de predicado R binario y $\mathcal{M}$ un modelo del lenguaje $\mathcal{L}$. Demostrar, usando compacidad, que no existe una fórmula $\varphi_R(x,y)$ tal que su interpretación represente que (x,y) pertenece a la clausura transitiva de la relación binaria $R^{\mathcal{M}}$.*

Solución 7.12.

Ejercicio 7.13.
Sea $\mathcal{L}$ un lenguaje de primer orden con igualdad. No existe un enunciado $\varphi \in \mathbf{Form}(\mathcal{L})$ tal que interpretada en una $\mathcal{L}$-estructura $\mathcal{M}$ cualquiera, diga "el dominio de $\mathcal{M}$ tiene infinitos elementos".

Solución 7.13.
Sean las siguientes fórmulas

$$\begin{aligned}\varphi_1 &= (\exists)x(x=x)\\ \varphi_2 &= (\exists x)(\exists y)(x\neq y)\dots\\ \varphi_n &= (\exists x_1)\dots(\exists x_n)\left(\bigwedge_{i\neq j} x_i \neq x_j\right)\end{aligned}$$

Y sea $\Gamma = \{\varphi_n : n\in\mathbb{N}\}$. Luego, $\Gamma \models \varphi$ donde

$$\varphi : \text{"el dominio de } \mathcal{M} \text{ tiene infinitos elementos"}$$

Luego, por el Teorema de Compacidad, existe $\Gamma_0 \subseteq \Gamma$ finito tal que $\Gamma_0 \models \varphi$.

Por otra parte, existe $N\in\mathbb{N}$ tal que $\Gamma_0 = \{\varphi_n : 1\leq n\leq N\}$. Por lo tanto, $\{\varphi_n : 1\leq n\leq N\}\models\varphi$ lo que resulta una contradicción.

Ejercicio 7.14.
Sea $\mathcal{L} = \{=, R\}$, R símbolo de predicado binario.

Sea $\mathcal{G}$ la clase de $\mathcal{L}$-estructuras que satisfacen las dos fórmulas

$$(\forall x)(\neg R(x,x))$$
$$(\forall x)(\forall y)(R(x,y)\to R(y,x))$$

Demostrar que no existe una sentencia φ del lenguaje $\mathcal{L}$ tal que interpretada en una estructura arbitraria $\mathcal{M}\in\mathcal{G}$, diga "$\mathcal{M}$ es conexo".

Solución 7.14.
Una posible forma de pensar estas $\mathcal{L}$-estructuras es usando grafos, donde R expresa

$$\text{"Hay una arista de } x \text{ a } y \text{ si y sólo si } R(x,y)\text{."}$$

Supongamos, entonces, que la sentencia φ : "$\mathcal{M}$ es conexo" (es decir que entre dos nodos cualesquiera en el grafo hay un camino de longitud finita) es expresable. Como φ es expresable, entonces $\neg\varphi$ (existen dos nodos tales ningún camino de longitud finita los une) también lo es.

Consideremos para cada $n\geq 2$, las siguientes fórmulas

$$\neg\varphi_n(x,y) = (\forall z_1)\dots(\forall z_n)\left(\bigwedge_{i=1}^{n-1} R(z_i,z_{i+1})\to((z_1=x\wedge z_n=y)\vee(z_1yx\wedge z_n=x))\right)$$

Sea $\Gamma = \{\neg\varphi_n(x,y) : n\geq 2\}$. Es claro que como Γ satisface a $\neg\varphi_n(x,y)$ para todo $n\geq 2$ entonces $\Gamma\models\neg\varphi$ y, por el Teorema de Compacidad, existe $\Gamma_0\subseteq\Gamma$ finito tal que $\Gamma_0\models\neg\varphi$.

Por lo lado, como $\Gamma_0\subseteq\Gamma$ es finito, existe $N\in\mathbb{N}$ tal que $\Gamma_0 = \{\neg\varphi_n(x,y) : 2\leq n\leq N\}\models\neg\varphi$. Es decir que fijado un modelo en $\mathcal{G}$ y una valuación, si no existe ningún camino de longtud menor o igual a N que una a x con y entonces no existe un camino que una x con y. Pero ésto es falso puesto que, por ejemplo, el camino puede ser de $N+1$.

Luego, la sentencia $\neg\varphi$ no es expresable y, por lo tanto, φ tampoco.

Ejercicio 7.15.
Sea $\mathcal{L} = \{=, f, c\}$ (f unaria). Demostrar que no es posible expresar la siguiente propiedad:

$$\text{"Para todo } x\text{, existe } n = n(x)\in\mathbb{N} \text{ tal que } f^{(n)}_{\mathcal{M}}(x) = c_{\mathcal{M}}\text{"}$$

Solución 7.15.
Supongamos que la sentencia

$$\varphi: \text{ “existe } x \text{ tal que para todo } n \in \mathbb{N} \text{ se tiene que } f_{\mathcal{M}}^{(n)}(x) \neq c_{\mathcal{M}}\text{”}$$

es expresable con un enunciado de $\mathcal{L}$.

Sea $\Gamma = \{f_{\mathcal{M}}^{(n)}(x) \neq c_{\mathcal{M}} : n \in \mathbb{N}\}$. *Luego, se tiene que* $\Gamma \models \varphi$ *y por el Teorema de Compacidad existe un* $N \in \mathbb{N}$ *tal que* $\Gamma_0 = \{f_{\mathcal{M}}^{(n)}(x) \neq c_{\mathcal{M}} : 1 \leq n \leq N\} \models \varphi$.

Sea el modelo $\mathcal{M} = \langle \{0, 1, \ldots, N+1\}, f_{\mathcal{M}}, c_{\mathcal{M}} = 0\rangle$ *donde* $f_{\mathcal{M}}(n) = \begin{cases} 0 & n = 1 \\ n-1 & n \geq 1 \end{cases}$ *y sea una valuación* v *tal que* $v(x) = N+1$.

Luego, según estas interpretaciones $f_{\mathcal{M}}(N+1) \neq 0$

$$\begin{aligned} f_{\mathcal{M}}(N+1) &\neq 0 \\ f_{\mathcal{M}}^2(N+1) &= N+1-2 \neq 0 \\ &\cdots \\ f_{\mathcal{M}}^j(N+1) &= N+1-j \neq 0 \end{aligned}$$

Por lo tanto, para todo $a \in \mathcal{M}$ *se tiene que* $f_{\mathcal{M}}^j(a) = a - (N+1) = 0$ *pues* $a \in \{0, 1, \ldots, N+1\}$. *Entonces para este modelo y valución no se satisface* φ, *lo que resulta ser una contradicción.*

Ejercicio 7.16. *Sea* $\mathcal{L}$ *un lenguaje con un símbolo de predicado* R *binario y* $\mathcal{M}$ *cualquier modelo cuyo dominio represente a los nodos de un grafo no orientado, y el símbolo* R *pueda ser interpretado como la relación “es adyacente a” (esto es, cualquier interpretación donde la relación* $R^{\mathcal{M}}$ *sea irreflexiva y simétrica). Demostrar que no es posible expresar la propiedad que afirma que un grafo es conexo, es decir, que entre cualquier par de nodos hay un camino de longitud finita.*

Solución 7.16.

Ejercicio 7.17. *Una función* f *se dice circular cuando para todo elemento* e *en el dominio de* f *existe un natural* $n > 0$ *tal que* $f^n(e) = e$, *en donde* f^n *representa el resultado de aplicar* n *veces la función* f *en forma sucesiva. Mostrar que no es expresable en primer orden la proposición “*f *es una función circular”.*

Solución 7.17.

Ejercicio 7.18.
Un número r *es llamado infinitesimal si es mayor que cero y menor que todos los reales positivos. Notemos con* $\mathcal{R}$ *al modelo estándar de los reales. Claramente, en* $\mathcal{R}$ *no hay infinitesimales.*

Sea $SQ_{\mathbb{R}}$ *una axiomatización de primer orden correcta con respecto a* $\mathcal{R}$ *sobre la signatura* $S = \{+, -, *, <, 0, 1\}$ *y sea* $SQ_{\mathbb{R}}^+$ *una extensión de* $SQ_{\mathbb{R}}$ *en donde se agrega un nuevo símbolo de constante* c *y los siguientes (infinitos) axiomas:*

$$\begin{aligned} &\boldsymbol{Positivo} && c > 0 \\ &\boldsymbol{Menor}_n && c * suc^{(n)}(0) < 1 \quad \text{para todo } n > 1 \end{aligned}$$

a. Demostrar que si $\mathcal{M}$ *es modelo de* $SQ_{\mathbb{R}}^+$, *entonces* $\mathcal{M}$ *es modelo de* $SQ_{\mathbb{R}}$.

b. Demostrar que $SQ_{\mathbb{R}}^+$ *es satisfacible. (Sugerencia: usar compacidad).*

c. Demostrar que cualquier axiomatización correcta con respecto a $\mathcal{R}$ *admite un modelo que posee números infinitesimales.*

Nota: $suc_{\mathcal{R}}$ *está definida como* $suc_{\mathcal{R}}(n) = n+1$ *para todo* n *natural y* $suc_{\mathcal{R}}(n) = 0$ *para todo* n *no natural.*

Solución 7.18.

a. Dado que $SQ_{\mathbb{R}} \subset SQ_{\mathbb{R}}^+$ *podemos afirmar que si* $\mathcal{M} \models SQ_{\mathbb{R}}^+$ *entonecs* $\mathcal{M} \models SQ_{\mathbb{R}}$.

b. Sea $\Delta \subset SQ_{\mathbb{R}}^+$ *un conjunto finito. Analicemos dos casos.*

1. *<u>$\Delta \subset SQ_{\mathbb{R}}$:</u>*
   Tenemos que dar un modelo para Δ.
   Sea entonces $\mathcal{M} = \langle \mathbb{R}, S_{\mathbb{R}}^1, c_{\mathcal{M}} = 42\rangle$. *En criollo,* $\mathcal{M}$ *es un modelo cuyo universo son los números reales, la interpretación de los símbolos de* S *son los estándares, y el símbolo de constante* c *se interpreta como el número real* 42.
   Como ninguna de las fórmulas en Δ *utilizan el símbolo de constante* c, *y siendo que* $SQ_{\mathbb{R}}$ *es una axiomatización correcta respecto a* $\mathbb{R}$, *podemos afirmar que* $\mathcal{M} \models \Delta$.

2. $\underline{\Delta \not\subset SQ_{\mathbb{R}}}$:

   Vale que Δ contiene fórmulas de $SQ^+_{\mathbb{R}}$ que no están en $SQ_{\mathbb{R}}$.

   Analicemos el caso en el que Δ contiene la fórmula Positivo y una cantidad finita de fórmulas $Menor_n$. Llamemos k al máximo índice de dichas fórmulas. Es entonces $Menor_k$ la fórmula con mayor índice.

   Sea entonces $\mathcal{M} = \langle \mathbb{R}, S_{\mathbb{R}}, c_{\mathcal{M}} = \frac{1}{k+1} \rangle$. En criollo, $\mathcal{M}$ es un modelo cuyo universo son los números reales, la interpretación de los símbolos de S son los estándares, y el símbolo de constante c se interpreta como el número real $\frac{1}{k+1}$.

   Veamos que vale que $\mathcal{M} \models \Delta$. Sea $\phi \in \Delta$.

   - *Si $\phi \in SQ_{\mathbb{R}}$ entonces sucede $\mathcal{M} \models \phi$, pues ϕ no utiliza el símbolo de constante c, $SQ_{\mathbb{R}}$ es correcta respecto a $\mathcal{R}$, y nuestro modelo $\mathcal{M}$ es básicamente el modelo $\mathcal{R}$ pero con la interpretación del símbolo c.*
   - *Si ϕ es Positivo, es decir $\phi : c > 0$.*

     Claramente sucede que $\frac{1}{k+1} >_{\mathbb{R}} 0$. Luego $\mathcal{M} \models \phi$.
   - *Si ϕ es $Menor_n$, es decir $c < \frac{1}{suc^{(n)}(0)}$.*

     Por lo mencionado previamente tiene que ser $n \leq k$. Entonces vale que $\frac{1}{k+1} <_{\mathbb{R}} \frac{1}{k} \leq_{\mathbb{R}} \frac{1}{n}$.

     Luego $\mathcal{M} \models \phi$.

   Podemos entonces ahora decir que por compacidad existe un modelo $\mathcal{G}$ tal que $\mathcal{G} \models SQ^+_{\mathbb{R}}$. Notar que no podemos saber nada sobre ese modelo, más allá de que satisface a todas las fórmulas de $SQ^+_{\mathbb{R}}$. Si sucede que este modelo posee números infinitesimales.

   Notemos también que, utilizando lo probado para el ítem 1, podemos afirmar que $\mathcal{G} \models SQ_{\mathbb{R}}$.

c. *Sea SQR' alguna axiomatización correcta respecto a $\mathcal{R}$. Sea $SQR'+$ su versión extendida agregando el axioma Positivo y los infinitos $Menor_n$.*

   Podemos ahora hacer el mismo razonamiento que hicimos en el ítem 1 y 2 (pues en su momento nada supusimos sobre el sistema axiomático $SQ_{\mathbb{R}}$ más allá de que era correcto respecto a $\mathcal{R}$). Entonces mostraremos así la existencia de un modelo $\mathcal{G}'$ que posee números infinitesimales y que además cumple que $\mathcal{G}' \models SQR'$.

   Era justamente lo que queríamos demostrar: que cuaquier axiomatización correcta respecto a $\mathcal{R}$ admite un modelo con números infinitesimales.

$\underline{\text{Observación:}}$ *Para cada axiomatización, habrá un modelo potencialmente distinto.*

Ejercicio 7.19. *Vamos a llamar $\mathcal{N} = \langle \mathbb{N}; 0; suc \rangle$ al modelo usual de los números naturales con cero y sucesor. Considerar una signatura de primer orden con igualdad $\mathcal{L}$ con un símbolo de constante 0 y un símbolo unario de función suc. Sea la siguiente axiomatización $SQ_{\mathcal{N}}$, que extiende a SQ con infinitos axiomas:*

$$
\begin{array}{ll}
\boldsymbol{S1} & (\forall x) suc(x) \neq 0 \\
\boldsymbol{S2} & (\forall x)(\forall y)(suc(x) = suc(y) \to x = y) \\
\boldsymbol{S3} & (\forall y)\big(y \neq 0 \to (\exists x)(y = suc(x))\big) \\
\boldsymbol{S4_n} & (\forall x)(suc^{(n)}(x) \neq x) \qquad \text{para todo } n > 1
\end{array}
$$

a. *Demostrar que $S1$ y toda instancia de $S4_n$ es verdadera en $\mathbb{N}$.*

b. *Dado un conjunto de fórmulas de primer orden Σ, demostrar que si existe un conjunto finito de fórmulas Γ tal que $\mathbf{Con}(\Gamma) = \mathbf{Con}(\Sigma)$, entonces existe un conjunto finito $\Sigma_0 \subseteq \Sigma$ tal que $\Sigma_0 \models \Sigma$.*

   Sugerencia: usar alguna de las formulaciones del teorema de compacidad.

c. *Demostrar que para cualquier subconjunto finito Γ de axiomas de $SQ_{\mathcal{N}}$ existe un modelo $\mathcal{M}$ tal que $\mathcal{M} \models \Gamma$ pero $\mathcal{M} \not\models SQ_{\mathcal{N}}$.*

d. *Sabiendo que SQ_N es correcta y completa con respecto a $\mathcal{N}$, demostrar que ninguna axiomatización correcta y finita de primer orden es completa con respecto a $\mathcal{N}$.*

   Sugerencia: aplicar el punto b al ítem anterior.

Solución 7.19.

Ejercicio 7.20.
Vamos a llamar $\mathcal{Z} = \langle \mathbb{Z}_{>0}, 1, div^{\mathcal{Z}}, primo^{\mathcal{Z}} \rangle$ a la estructura cuyo universo son los números enteros postivos, $div^{\mathcal{Z}} \subset \mathbb{Z}^2_{>0}$ es la relación binaria de divisibilidad, y $primo^{\mathcal{Z}} \subset \mathbb{Z}_{>0}$ es el conjunto de números primos.

Consideremos la siguiente axiomatización $SQ_{\mathcal{Z}}$ (escrita en el lenguaje con igualdad sobre la signatura $\langle 1, div, primo \rangle$) que extiende a SQ con los axiomas:

S1	$(\forall x)(\forall y)(\forall z)((x \; div \; y \wedge y \; div \; z) \rightarrow (x \; div \; z))$
S2	$(\forall x)(\forall y)((x \; div \; y \wedge y \; div \; x) \rightarrow (x = y))$
S3	$(\forall x)(x \; div \; x \wedge 1 \; div \; x)$
S4	$(\forall x)(primo(x) \longleftrightarrow (x \neq 1 \wedge (\forall y)(y \; div \; x \longleftrightarrow (y = x \vee y = 1))))$
S5	$(\forall x)(\exists y)(\neg(x = y) \wedge x \; div \; y)$

a. Demostrar que ***S5*** *es verdadera en* $\mathcal{Z}$.

b. Dar una fórmula φ *y un modelo* $\mathcal{M}$ *tal que:*

1. *todos los axiomas de* $SQ_{\mathcal{Z}}$ *sean válidos en* $\mathcal{M}$;
2. $\mathcal{M} \not\models \varphi$;
3. $\mathcal{Z} \models \varphi$;

c. Asumiendo que $SQ_{\mathcal{Z}}$ *es correcto con respecto* $\mathcal{Z}$, *demostrar que* $SQ_{\mathcal{Z}}$ *no es completa con respecto a* $\mathcal{Z}$.

Solución 7.20.

a. Vamos a demostrar que ***S5*** *es válido en* $\mathcal{Z}$. *Sea* v *una valuación.*

$$\begin{aligned}
&\mathcal{Z} \models (\forall x)(\exists y)(\neg(x = y) \wedge x \; div \; y)[v] && \Leftrightarrow \forall n \in \mathbb{Z}_{>0}, \\
&\mathcal{Z} \models (\exists y)(\neg(x = y) \wedge x \; div \; y)[v(x = n)] && \Leftrightarrow \forall n \in \mathbb{Z}_{>0}, \exists m \in \mathbb{Z}_{>0}, \\
&\mathcal{Z} \models (\neg(x = y) \wedge x \; div \; y)[v(x = n, y = m)] && \Leftrightarrow \forall n \in \mathbb{Z}_{>0}, \exists m \in \mathbb{Z}_{>0}, \\
&\mathcal{Z} \models \neg(x = y)[v(x = n, y = m)] \text{ y } \mathcal{Z} \models x \; div \; y[v(x = n, y = m)] && \Leftrightarrow \forall n \in \mathbb{Z}_{>0}, \exists m \in \mathbb{Z}_{>0}, \\
&\mathcal{Z} \not\models (x = y)[v(x = n, y = m)] \text{ y } \mathcal{Z} \models x \; div \; y[v(x = n, y = m)] && \Leftrightarrow \forall n \in \mathbb{Z}_{>0}, \exists m \in \mathbb{Z}_{>0}, n \neq m \text{ y } n \; div^{\mathcal{Z}} \; m
\end{aligned}$$

Y esta última afirmación es cierta tomando, por ejemplo, $m = 2n$.

b. Una posible solución es:

$$\begin{aligned}
\mathbb{U} &= \left\{n \in \mathbb{Z}_{>0} | 2 \; div^{\mathcal{Z}} \; n\right\} \cup \{1\} \\
\mathcal{M} &= \langle \mathbb{U}, 1, div^{\mathcal{Z}} \cap \mathbb{U}^2, primo^{\mathcal{Z}} \cap \mathbb{U} \rangle \\
\varphi &:= (\exists x)(\exists y)(\neg(x = y) \wedge (primo(x) \wedge primo(y)))
\end{aligned}$$

c. Nuestro $\mathcal{M}$ *del inciso anterior es modelo de todos los axiomas de* $SQ_{\mathcal{Z}}$.

Por lo tanto, $SQ_{\mathcal{Z}}$ *es correcta con respecto a* $\mathcal{M}$. *Además, vimos que* $\mathcal{M} \models \varphi$. *Por lo tanto, como* $SQ_{\mathcal{Z}}$ *es correcta con respecto a* $\mathcal{M}, \forall_{SQ_{\mathcal{Z}}} \varphi$.

Entonces, como $\mathcal{Z} \models \varphi$, $SQ_{\mathcal{Z}}$ *no es completa con respecto a* $\mathcal{Z}$.

Ejercicio 7.21.
Sea $\mathcal{Z} = \langle \mathbb{Z}, <_{\mathcal{Z}} \rangle$ *el modelo usual de los enteros con la relación es menor a. Considerar un lenguaje de primer orden con igualdad* $\mathcal{L}$ *con un símbolo de predicado* $<$. *Sea la siguiente axiomatización* $SQ_{\mathcal{Z}}$ *que extiende a SQ con los siguientes axiomas:*

S1	$(\forall x)\neg(x < x)$
S2	$(\forall x)(\forall y)(x < y \rightarrow \neg(y < x))$
S3	$(\forall x)(\forall y)(\forall z)((x < y \wedge y < z)to(x < z))$
S4	$(\forall x)(\forall y)(\neg(x = y) \rightarrow (x < y \vee y < x))$
S5	$(\forall x)(\exists y)(x < y)$

1. *>Es* $SQ_{\mathcal{Z}}$ *correcta en* $\mathcal{Z}$?
2. *>Es* $SQ_{\mathcal{Z}}$ *completa respecto a la clase de todos los modelos?*
3. *>Es* $SQ_{\mathcal{Z}}$ *correcta respecto a la clase de todos los modelos?*
4. *>Es* $SQ_{\mathcal{Z}}$ *completa respecto a* $\mathcal{Z}$?

Solución 7.21.

1. *Recordemos que* $SQ_{\mathcal{Z}} = SQ \cup \{S1, \ldots, S5\}$. *Como SQ es correcto para la clase de todos los modelos, en particular sucede que* $\mathcal{Z} \models SQ$. *Ahora falta ver que* $\{S1, \ldots, S5\}$ *valen en* $\mathcal{Z}$. *Una vez visto eso, argumentamos que modus ponens conserva validez, y con eso podemos concluir que* $SQ_{\mathcal{Z}}$ *es correcta para* $\mathcal{Z}$.

2. *Sea ϕ tal que para cualquier modelo $\mathcal{M}$ sucede que $\mathcal{M} \models \phi$ (en criollo, ϕ es una verdad de la clase de todos los modelos). Como SQ es completo respecto a la clase de todos los modelos, podemos decir que $SQ \vdash \phi$. Siendo que $SQ \subset SQ_{\mathcal{Z}}$, vale entonces que $SQ_{\mathcal{Z}} \vdash \phi$. Luego, $SQ_{\mathcal{Z}}$ es completa respecto a la clase de todos los modelos.*

3. *Supongamos que $SQ_{\mathcal{Z}}$ es correcta respecto a la clase de todos los modelos. Vale que $SQ_{\mathcal{Z}} \vdash S3$.*

   Luego para cualquier modelo $\mathcal{M}$ sucede que $\mathcal{M} \models S3$. Ahora bien, sea $\mathcal{M}_0$ un modelo cuya relación no es transitiva. Claramente sucede que $\mathcal{M}_0 \not\models S3$. Absurdo! Luego $SQ_{\mathcal{Z}}$ no es correcta respecto a la clase de todos los modelos.

4. *Sea la fórmula $\phi : (\exists x)(\forall y)(x \neq y \to x < y)$. Definimos también el siguiente modelo $\mathcal{N} = \langle \mathbb{N}, <_{\mathbb{N}} \rangle$ (los naturales con la relación es menor a). Observemos lo siguiente:*

   a. *$\mathcal{N} \models \phi$ (pues existe un mínimo, el número cero). Entonces $\mathcal{N} \not\models \neg\phi$.*

   b. *$\mathcal{Z} \models \neg\phi$ (pues justamente no existe un mínimo).*

   c. *$\mathcal{N} \models SQ_{\mathcal{Z}}$.*

   Se desprende de la observación 4c, utilizando un razonamiento análogo al del ítem 1 de este ejercicio, que el sistema $SQ_{\mathcal{Z}}$ es correcto respecto a $\mathcal{N}$.

   Ahora bien, supongamos que $SQ_{\mathcal{Z}}$ fuera completo respecto a $\mathcal{Z}$. Entonces, dado que $\mathcal{Z} \models \neg\phi$ debe suceder que $\mathcal{Z} \vdash \neg\phi$. Pero siendo $SQ_{\mathcal{Z}}$ correcto respecto a $\mathcal{N}$, debe entonces suceder que $\mathcal{N} \models \neg\phi$. Pero según mencionamos en la observación 4a, sucede lo contrario. Absurdo! Concluimos entonces que $SQ_{\mathcal{Z}}$ no es completa respecto a $\mathcal{Z}$.

Ejercicio 7.22.
Sea $\mathcal{L} = \{c, f, =\}$ un lenguaje de primer orden con c un símbolo de constante y f un símbolo de función unario. Sea $\mathcal{N} = \langle \mathbb{N}, 0, *_2, = \rangle$ la estructura usual de los naturales con el 0 y la función multiplicar por 2.

Consideramos $SQ_{\mathcal{N}}$ la axiomatización que extiende SQ con los siguientes axiomas:

$$\textbf{\textit{S1}} \quad (\forall x)(f(x) = x \longleftrightarrow x = c)$$
$$\textbf{\textit{S2}} \quad (\forall x)(\forall y)(x \neq y \to f(x) \neq f(y))$$

a. *Probar que $SQ_{\mathcal{N}}$ es correcta respecto de $\mathcal{N}$.*

b. *Hallar una fórmula φ y un modelo $\mathcal{M}$ de $SQ_{\mathcal{N}}$ tales que $\mathcal{N} \models \varphi$ y $\mathcal{M} \not\models \varphi$.*

c. *Probar que $SQ_{\mathcal{N}}$ no es completa respecto de $\mathcal{N}$.*

Solución 7.22.

a. *Hay que ver que si $SQ_{\mathcal{N}} \vdash \varphi$ entonces $\mathcal{N} \models \varphi[v]$ para toda valuación v de $\mathcal{N}$.*

   Recordar que también se tiene que $SQ_{\mathcal{N}}$ es correcta y completa con respecto a $\mathcal{C} = \{\mathcal{A} | \mathcal{A} \models SQ\}$.

   Supongamos $\vdash_{SQ_{\mathcal{N}}} \varphi$. Entonces (por definición de $\vdash_{SQ_{\mathcal{N}}}$ más el hecho de que $SQ \subseteq SQ_{\mathcal{N}}$) tenemos que $\{S1, S2\} \vdash \varphi$. Luego por la correctitud de SQ vista en la teórica, tenemos que $\{S1, S2\} \models \varphi$. Pero $\mathcal{N}$ satisface $S1$ y $S2$. Luego por definición de consecuencia semántica, satisface φ.

b. *$\mathcal{M} = \langle \{0\}, 0, id, = \rangle$, $\varphi = (\exists x)(\exists y)(x \neq y)$.*

   Otro ejemplo se tiene si $\mathcal{M} = \langle \{1, 2, 3\}, 1, f_{\mathcal{M}}, = \rangle$, donde $f_{\mathcal{M}}(1) = 1, f_{\mathcal{M}}(2) = 3$ y $f_{\mathcal{M}}(3) = 2$. Y sea $\varphi = \neg(\forall x)(\exists y)(f(y) = x)$.

c. *Si lo fuese, como $\mathcal{N} \models \varphi$ (φ es la del ítem anterior), tendríamos que $\vdash_{SQ_{\mathcal{N}}} \varphi$. Pero si fuese así, tendríamos que $SQ_{\mathcal{N}} \vdash \varphi$ y por correctitud de SQ, valdría que $SQ_{\mathcal{N}} \models \varphi$ pero esto no es cierto por el ítem b. Luego no es completa.*

Isomorfismos y conjuntos expresables

Ejercicio 7.23.
Sea $\mathcal{L}$ un lenguaje de primer orden. Sean U_1 y U_2 dos $\mathcal{L}$-estructuras con dominios A_1 y A_2 respectivamente.

Un isomorfismo entre estas estructuras es una función $g : A_1 \to A_2$ que verifica las siguientes condiciones:

(a) g es biyectiva.

(b) Si P es un símbolo de predicados n-ario de $\mathcal{L}$ y $a_1, \ldots, a_n \in A_1$ entonces

$$(a_1, \ldots, a_n) \in P_{A_1} \text{ si y sólo si } (g(a_1), \ldots, g(a_n)) \in P_{A_2}$$

(c) Si f es un símbolo de función n-ario de $\mathcal{L}$ y $a_1, \ldots, a_n \in A_1$ entonces

$$g(f_{A_1}(a_1, \ldots, a_n)) = f_{A_2}(g(a_1), \ldots, g(a_n)).$$

(d) Si c es un símbolo de constante de $\mathcal{L}$ entonces $g(c_{A_1}) = c_{A_2}$.

Supongamos que existe $g : A_1 \to A_2$ isomorfismo y ϕ una fórmula de $\mathcal{L}$.

Probar que para toda valuación $v : \mathbf{Var} \to A_1$ se tiene que $A_1 \models \phi[v]$ si y sólo si $A_2 \models \phi[g \circ v]$, donde $g \circ v : \mathbf{Var} \to A_2$ es la valuación dada por la composición.

Notar que en particular, si ϕ es un enunciado de $\mathcal{L}$ se tiene que $A_1 \models \phi$ si y sólo si $A_2 \models \phi$.

Solución 7.23.
La prueba se hace por inducción en el número de conectivos que aparece en ϕ.

(i) Si el número es 0 entonces ϕ es una fórmula atómica. Luego $\phi = P(t_1, \dots, t_n)$ siendo $t_1, \dots, t_n$ términos.

Veamos primero que si t es un término de $\mathcal{L}$ entonces $g(\overline{v}(t)) = \overline{g \circ v}(t)$, donde v es la extensión de la valuación $\overline{v}$ al conjunto de los términos. Esto se hace por inducción.

- *Si t es una variable $x_i, g(\overline{v}(t)) = g(v(x_i)) = \overline{g \circ v}(x_i)$.*
- *Si $t = c$, con c un símbolo de constante, $g(\overline{v}(c)) = g(c_{A_1}) = c_{A_2}$ mientras que $\overline{g \circ v}(x_i) = c_{A_2}$ y, por lo tanto, coinciden.*
- *Si $t = f(t_1, \dots, t_n)$ con $t_1, \dots, t_n$ términos se tiene que*

$$g(\overline{v}(t)) = g\left(f_{A_1}(\overline{v}(t_1), \dots, \overline{v}(t_n))\right) = f_{A_2}(g(\overline{v}(t_1)), \dots, g(\overline{v}(t_n))) = \overline{g \circ v}(f(t_1, \dots, t_n))$$

Lo que prueba que $g(\overline{v}(t)) = \overline{g \circ v}(t)$ para todo término t.

Por lo tanto, $A_1 \models P(t_1, \dots, t_n)[v]$ si y sólo si $(\overline{v}(t_1), \dots, \overline{v}(t_n)) \in P_{A_1}$. Lo que equivale, por la parte (b) de la definición de isomorfismo que

$g(\overline{v}(t_1)), \dots, g(\overline{v}(t_n)) \in P_{A_2}$ si y sólo si (por la identidad anterior) $(\overline{g \circ v}(t_1), \dots, \overline{g \circ v}(t_n)) \in P_{A_1}$ si y sólo si $A_2 \models \phi[g \circ v]$.

Luego vale la propiedad para el caso base; es decir, para las fórmulas atómicas.

(ii) Si ϕ no es atómica tenemos los siguientes casos:

1. *$\phi = \neg\beta$, con $\beta \in \mathbf{Form}$*
2. *$\phi = (\alpha_1 \vee \alpha_2)$, con $\alpha_1, \alpha_2 \in \mathbf{Form}$*
3. *$\phi = (\alpha_1 \wedge \alpha_2)$, con $\alpha_1, \alpha_2 \in \mathbf{Form}$*
4. *$\phi = (\alpha_1 \to \alpha_2)$, con $\alpha_1, \alpha_2 \in \mathbf{Form}$*
5. *$\phi = (\forall x_i)\beta$, con $\beta \in \mathbf{Form}$ y x_i una variable.*
6. *$\phi = (\exists x_i)\beta$, con $\beta \in \mathbf{Form}$ y x_i una variable.*

Los casos 1. a 4. salen inmediatamente usando la definición inductiva del valor de verdad en términos de los 4 conectivos de la lógica proposicional.

Para el caso 5. se sigue que $A_1 \models (\forall x_i)\beta[v]$ si y sólo si para cualquier $a \in A_1$ se tiene que $A_1 \models \beta[v(x = a)]$ (definición inductiva del valor de verdad del cuantificador $\forall$).

Por hipótesis inductiva se sigue que ésto equivale a decir que $A_2 \models \phi[g \circ v(x = a)] \forall a \in A_1$.

Como g es biyectiva esto equivale a decir que $A_2 \models \phi[g \circ v(x = a)] \forall a \in A_2$ lo que implica que $A_2 \models (\forall x_i)\beta[g \circ v]$.

La prueba del caso 6. se hace de forma análoga.

Cuando ϕ es un enunciado el valor de verdad de ϕ no depende de la valuación que se tome luego ϕ es válido en A_1 si y sólo si es válido en A_2

Ejercicio 7.24.
Sea $\mathcal{L}$ un lenguaje de primer orden y sea U una $\mathcal{L}$-estructura con dominio D. Si A es un subconjunto de D diremos que A es expresable si existe una fórmula ϕ de $\mathcal{L}$ con una única variable libre x tal que para toda valuación $v : \mathbf{Var} \to D$ se tiene que $D \models \phi[v]$ si y sólo si $v(x) \in A$.

Probar que si $A \subseteq D$ es expresable y existe $g : D \to D$ isomorfismo, entonces $g(A) = A$.

Solución 7.24.
Sea $d \in A$ y v una valuación tal que $v(x) = d$. Luego $A \models \phi[v]$.

Como g es isomorfismo se sigue que $A \models \phi[g \circ v]$ (ejercicio anterior) lo que implica que $g(d) \in A$.

En forma análoga se prueba que si $g(a) \in A$ entonces $a \in A$ ya que la inversa de un isomorfismo es un isomorfismo.

Ejercicio 7.25.
Sea $\mathcal{L}$ un lenguaje de primer orden con igualdad y un símbolo de función binario f. Probar que

a. $U_1 = (\mathbb{N}, +)$ *y* $U_2 = (\mathbb{N}, \cdot)$ *no son isomorfismos.*

b. $U_1 = (\mathbb{N}\setminus\{0\}, +)$ *y* $U_2 = (\mathbb{N}\setminus\{0\}, \cdot)$ *no son isomorfismos.*

c. $U_1 = (\mathbb{R}, +)$ *y* $U_2 = (\mathbb{R}_{>0}, \cdot)$ *son isomorfismos.*

Solución 7.25.

a. Basta ver que existe un enunciado ϕ *que sea válida en la primer estructura y no en la segunda. Es decir* $U_1 \models \phi$ *y* $U_2 \not\models \phi$.

Para ello debemos encontrar una propiedad que sea expresable en este lenguaje y que sea válido en una estructura y no en la otra.

Un ejemplo sencillo es la propiedad cancelativa. La suma es cancelativa pero el producto no ya que por ejemplo $0 \cdot 3 = 0 \cdot 2 = 0$ *pero* $2 \neq 3$.

La propiedad cancelativa se expresa mediante el enunciado

$$\phi : (\forall x)(\forall y)(\forall z)(f(x,z) = f(y,z) \to y = z)$$

b. Sabemos que el orden usual de los números naturales es un orden total.

Por otro lado la relación $\leq$ *se puede expresar en* $\mathbb{N}\setminus\{0\}$ *por medio de la suma por la fórmula* $x \leq y$ *si y sólo si* $x = y$ *o bien existe* $z \in \mathbb{N}\setminus\{0\}$ *tal que* $x = z = y$. *Luego definimos el siguiente enunciado*

$$\phi = (\forall x)(\forall y)((x = y) \vee ((\exists z)(f(x,z) = y)) \vee (\exists z)(f(y,z) = x))$$

Este enunciado en válido en U_1. *Sin embargo no es válido en* U_2 *ya que si cambiamos la suma por el producto lo que expresa el enunciado en esta estructura es que para todo par de números naturales* x, y *o son iguales o* x *divide a* y *o bien* y *divide a* x, *hecho que no es cierto. Por ejemplo* 2 *no es divisor de* 3 *ni* 3 *es divisor de* 2.

c. La función exponencial con base e *es un ejemplo de un isomorfismo. En efecto si* $g : \mathbb{R} \to \mathbb{R}_{>0}$ *es la función* $g(x) = e^x$ *es una biyección y por propiedades de la exponencial se tiene que* $g(x + y) = e^{x+y} = e^x e^y$ *para todo par* $x, y \in \mathbb{R}_{>0}$.

Ejercicio 7.26.
Sea $\mathcal{L}$ *un lenguaje con igualdad y que no contiene ningún otro símbolo de predicados ni tampoco contiene símbolos de función ni de constantes.*

Probar que si U *es una* $\mathcal{L}$ *estructura con dominio* D, *entonces* $A \subseteq D$ *es expresable si y sólo si* $A = \emptyset$ *o bien* $A = D$.

Solución 7.26.
Es claro que el vacío es epresable por la fórmula $\neg(x = x)$ *y* D *por la fórmula* $x = x$.

Supongamos que A *es un subconjunto expresable que no sea vacío y* $A \neq D$. *Luego existen elementos* a, b *con* $a \in A$ *y* $b \notin A$.

Sea $g : D \to D$ *la función definida por* $g(a) = b, g(b) = a$ *y* $g(x) = x$ *si* $x \neq a$ *y* $x \neq b$. *Es inmediato ver que* g *es una biyección y por ende un isomorfismo.*

Notar que en este lenguaje cualquier biyección es automáticamente un isomorfismo.

Se sigue que $g(A) = A$ *lo que es imposible ya que* $g(a) = b$ *y* $b \notin A$.

Ejercicio 7.27.
Sea $\mathcal{L}$ *un lenguaje con igualdad y sea* U *es una* $\mathcal{L}$*-estructura con dominio* D. *Probar que si* $A, B \subseteq D$ *son expresables entonces* $A \cup B$ *y* $A \cap B$ *son expresables.*

Más aún, A *es expresable si y sólo si* A^c *es expresable.*

Solución 7.27.
Sean $\phi, \beta \in$ **Form** *con una variable libre que expresan a* A *y a* B *respectivamente.*

Es inmediato ver que $(\phi \vee \beta)$

expresa la unión $A \cup B$ *y* $(\phi \wedge \beta)$ *expresa* $A \cap B$.

Por otro lado $\neg\phi$ *expresa a* A^c *y* $\neg\neg\phi$ *expresa a* A *lo que prueba la última equivalencia.*

Ejercicio 7.28.
Sea $\mathcal{L}$ *un lenguaje con un símbolo de predicados binario. Sea* U *una* $\mathcal{L}$ *estructura ordenada (es decir, el símbolo* P *se interpreta como la relación de orden parcial).*

El dominio de U *es el conjunto de* 4 *elementos* $\{0, a, b, 1\}$ *donde* 0 *es el primer elemento de* U, 1 *es el último elemento de* U, *y los elementos* a *y* b *son incomparables.*

Probar que los únicos elementos distinguibles son 0 *y* 1.

Solución 7.28.
Sean las fórmulas ϕ_0 y ϕ_1

$$\phi_0 = (\forall z)P(x,z)$$
$$\phi_1 = (\forall z)P(z,x)$$

Es claro que ϕ_0 expresa al conjunto unitario $\{0\}$ y que ϕ_1 expresa al conjunto unitario $\{1\}$ lo que dice que 0 y 1 son distinguibles.

Por otro lado la función $g : U \to U$ dada por $g(0) = 0, g(a) = b, g(b) = a$ y $g(1) = 1$ es un isomorfismo que no deja fijo ni a a ni a b.

Notar que la noción de isomorfismo entre dos conjuntos ordenados es un caso particular de la noción de isomorfismo entre $\mathcal{L}$-estructuras.

Ejercicio 7.29.
Sea $\mathcal{L}$ un lenguaje con igualdad y un símbolo de función binario f. Sea $U = (\mathbb{N}, \cdot)$.

Probar que el conjunto de los números primos es expresable.

Solución 7.29.
Recordando que un número p es primo si $p \neq 0, p \neq 1$ y si a es un divisor de p entonces $a = 1$ o bien $a = p$.

El problema es como expresar el complemento del conjunto formado por el 0 y por el 1. Para ello hay que notar que la fórmula $x^2 = x$ se satisface si y sólo si $x = 0$ o bien $x = 1$.

Luego la siguiente fórmula expresa al conjunto de los números primos:

$$\phi(x) = \neg(f(x,x) = x) \wedge (\forall z)((\exists y)f(z,y) = x \to (f(z, z = z \vee z = x)))$$

Ejercicio 7.30.
Sea Sea $\mathcal{L}$ un lenguaje con igualdad y un símbolo de función binario f. Sea $U = (\mathbb{N}, +)$. Probar que si $g : \mathbb{N} \to \mathbb{N}$ es un isomorfismo entonces g es la identidad.

Solución 7.30.
Como g debe preservar la suma se sigue que $g(0) = g(0 + 0) = g(0) + g(0)$ lo que implica que $g(0) = 0$.

Probemos por inducción en n que $g(n) = n$ para todo $n \in \mathbb{N}$.

Ya vimos que si $n = 0$ entonces $g(n) = n$.

Asumamos por hipótesis inductiva que $g(x) = x$ para todo $x < n$, siendo $n > 0$. Como $n = (n-1)+1$ se sigue que $g(n) = g(n-1) + g(1) = n - 1 + 1 = n$.

Luego $g(x) = x$ para todo $x \in \mathbb{N}$.

Ejercicio 7.31.
Sea Sea $\mathcal{L}$ un lenguaje con igualdad y un símbolo de función binario f. Sea $U = (\mathbb{N}, +)$.

a. *Probar que si A es un subconjunto de $\mathbb{N}$ tal que A es finito o A^c es finito entonces A es expresable.*

b. *Dar un ejemplo de un subconjunto de $\mathbb{N}$ tal que sea expresable pero ni que él ni su comlemento s ean conjuntos finitos.*

Solución 7.31.

a. *Basta ver que si $n \in \mathbb{N}$ entonces n es distinguible ya que la unión finita de conjuntos expresables es expresable y, por lo tanto, al ser todo conjunto finito la unión de sus subconjuntos unitarios el resultado sigue.*

   Para ello hacemos inducción en n. Si $n = 0$, 0 es distinguible por la fórmula $\phi_0 = f(x,x) = x$.

   Asumamos que k sea distinguible para todo $k < n$ con $n > 0$. Sean las $n-1$ fórmulas $\phi_0, \ldots, \phi_{n-1}$ que distinguen a $0, \ldots, n-1$.

   La siguiente fórmula distingue a n,

$$\phi = \neg\phi_0(x) \wedge \ldots \wedge \neg\phi_{n-1} \wedge (\forall z)(\phi_0(z) \vee \ldots \vee \phi_{n-1}(z) \vee (\exists y)f(y, x = z))$$

b. *Un ejemplo de un subconjunto expresable que no sea finito y que tampoco lo sea su complemento es el conjunto de los números pares.*

   En efecto los pares se expresan por la fórmula

$$\alpha = (\exists y)(f(y,y) = x)$$

Ejercicio 7.32.
Sea σ el lenguaje vacío, es decir, σ no posee ningún símbolo. Una σ-estructura es simplemente un conjunto no vacío. Estamos interesados en la consulta EVEN *definida como*

$$\text{EVEN}(A) = 1 \text{ si y sólo si } |A| \text{ es par.}$$

para cualquier conjunto finito no vacío A. Sobre conjuntos infinitos el valor de EVEN *es arbitrario.*

Demuestre que EVEN *no es expresable mediante una σ-fórmula*

Solución 7.32.
En la mayoría de los casos se procede por reducción al absurdo. Supongamos que EVEN *sí es expresable en el lenguaje σ y sea ϕ tal fórmula.*

La idea es construir un sistema axiómatico que fuerce la existencia de un par de modelos infinitos, uno para ϕ y otro para $\neg\phi$. Luego Lwenheim-Skolem hará el resto.

Consideremos para cada $n \in \mathbb{N}$ la fórmula:

$$\lambda_n \equiv \exists x_1 \dots \exists x_n \bigwedge_{i \neq k} \neg(x_i = x_j), \quad n \in \mathbb{N}$$

la cual expresa que existen al menos n elementos en el dominio de alguna estructura. Consideremos los siguientes sistemas: $\Delta_1 = \{\phi\} \cup \{\lambda_n : n \in \mathbb{N}\}$ y $\Delta_2 = \{\neg\phi\} \cup \{\lambda_n : n \in \mathbb{N}\}$.

Veamos que Δ_1 es satisfacible, usando el teorema de compacidad. Sea Δ_1' un subconjunto finito de Δ_1.

Sea m algún entero par tal que $n \leq m$ siempre que $\lambda_n \in \Delta_1'$ (si en Δ_1' no hay fórmulas λ_n tomamos $m = 2$).

Entonces el modelo $A = \{1, 2, \dots, m\}$ es un modelo de Δ_1'.

De manera análoga se puede demostrar que todo subconjunto finito de Δ_2 es satisfacible. Luego, Δ_1 y Δ_2 son satisfacibles.

Notar además que ambos conjuntos sólo admiten modelos infinitos y por Lowenheim-Skolem, podemos valernos de modelos A_i de cardinal infinito numerable para cada Δ_i.

>Es esto realmente posible? Notemos que los modelos A_1 y A_2 son isomorfos pero $A_1 \models \varphi$ y $A_2 \models \neg\varphi$. Esto es una contradicción.

Ejercicio 7.33.
Consideremos el lenguaje $\mathcal{L} = \{U\}$, donde U es una relación unaria. Demuestre que sobre $\mathcal{L}$ no es expresable en primer orden la siguiente propiedad:

$$\text{para una estructura } \mathcal{A} = (A, U^{\mathcal{A}}), \text{ tanto } U^{\mathcal{A}} \text{ como } |A - U^{\mathcal{A}}| \text{ son pares.}$$

Solución 7.33.
Supongamos que tal fórmula sí existe y llamémosla ψ. Dado que ya sabemos que EVEN *no es expresable en primer orden sobre el lenguaje vacío sería bueno aprovechar ese resultado.*

La idea que aplicaremos a continuación no se aleja mucho de los méetodos de reducción vistos en temas anteriores.

Veamos que para cualquier σ-estructura B existe una $\mathcal{L}$-estructura $\mathcal{A}$ tal que

$$\text{EVEN}(B) = 1 \text{ si y sólo si } \mathcal{A} \models \psi$$

En efecto, basta con definir $\mathcal{A} = \{A, U^{\mathcal{A}}\}$ con $A = U^{\mathcal{A}} = B$.

Notar ademáas que por de
finición $U^{\mathcal{A}} = \{x \in A : x = x\}$. Entonces, para cualquier valuación v, es cierto que

$$B, v \models (x = x) \text{ si y sólo si } \mathcal{A}, v \models U(x)$$

Consideremos la fórmula ψ' que se obtiene de sustituir en ψ cualquier átomo $U(x)$ por $x = x$. Notar que esta es una fórmula sobre el lenguaje σ y es tal que

$$B \models \psi' \text{ si y sólo si } \mathcal{A} = \models \psi$$

pero esto es claramente una contradicción.