Lógica y Computabilidad - Recopilación

Lucas Di Salvo

Resumen

Escribí este apunte al momento de preparar el examen final, recopilando las diapositivas teóricas/prácticas para un estudio y lectura más agradable. No pretende ser un remplazo del asistir a las clases ni de realizar los ejericios de las guías prácticas. A su vez, el mismo puede estar incompleto o contener errores.

Contenidos

1. Máquinas de Turing

Una máquina de Turing se compone de una cinta infinita que se extiende en ambas direcciones y está dividida en celdas. Cada celta contiene un símbolo de un alfabeto dado Σ, el cual adicionalmente tiene un símbolo especial $*$ que representa el blanco en una celda.

Por otro lado, los símbolos L y R no pertenecen a Σ por se símbolos reservados que representan acciones que puede realizar la cabeza; la misma lees y escribe un símbolo a la vez, moviendose una posición a la izquierda o una posición a la derecha.

Por último, se tiene una tabla finita de instrucciones, que indica qué hacer en cada paso.

1.1. Tabla de instrucciones

La tabla de instrucciones se compone de:

- Un alfabeto Σ con $* \in \Sigma$ y $L, R \notin \Sigma$.
- Un conjunto finito de estados Q.
- Un conjunto de acciones $A = \Sigma \cup \{L, R\}$.
  - Un símbolo $s \in \Sigma$ se interpreta como "escribir s en la posición actual".
  - L se interpreta como "mover la cabeza una posición hacia la izquierda".
  - R se interpreta como "mover la cabeza una posición hacia la derecha".

Esta tabla de instrucciones es un subconjunto finito T de

$$Q \times \Sigma \times A \times Q,$$

donde la tupla

$$(q, s, a, q') \in T$$

se interpreta como "Si la máquina está en el estado q leyendo en la cinta el símbolo s, entonces realiza la acción a y pasa al estado q'".

1.2. Definición de máquina de Turing

Una máquina de Turing es una tupla

$$(\Sigma, Q, T, q_0, q_f)$$

donde

- Σ es el conjunto finito de símbolos ($L, R \notin \Sigma$, $* \in \Sigma$).
- Q es el conjunto finito de estados, de los cuales, dos son distinguidos:
  - $q_0 \in Q$ es el estado inicial.
  - $q_f \in Q$ es el estado final.
- $T \subseteq Q \times \Sigma \times \Sigma \cup \{L, R\} \times Q$ es la tabla de instrucciones (va a ser finita porque Σ y Q lo son).

- Cuando no hay restricciones sobre T decimos que $\mathcal{M}$ es una máquina de Turing no determinística.
- Cuando no hay dos instrucciones en T que empiezan con las mismas primeras dos coordenadas, decimos que $\mathcal{M}$ es una máquina de Turing determinística.

1.3. Representación de números y tuplas

Fijamos $\Sigma = \{*, 1\}$

- Representamos a los números naturales en unario. El número $x \in \mathbb{N}$ se representa como

$$\overline{x} = \underbrace{1 \dots 1}_{x+1}$$

- Representamos a las tuplas $(x_1, \dots, x_n)$ como una lista de la representación de los x_i separados blancos. La tupla $(x_1, \dots, x_n)$ se representa como

$$*\overline{x_1} * \overline{x_2} * \cdots * \overline{x_n} *$$

Por ejemplo

- El número 0 se representa como 1
- El número 3 se representa como 1111
- La tupla (1,2) se representa como $*11 * 111*$
- La tupla (0,0,1) se representa como $*1 * 1 * 11*$

2. Funciones parciales

Siempre se va a trabajar con funciones $f : \mathbb{N}^n \to \mathbb{N}$.

Pero podrán ser funciones parciales. Una función parcial f es una función que puede estar indefinida para algunos (tal vez ninguno; tal vez todos) sus argumentos.

- Notamos $f(x_1, \dots, x_n) \downarrow$ cuando f está definida para $(x_1, \dots, x_n)$. En esta caso $f(x_1, \dots, x_n)$ es un número natural.
- Notamos $f(x_1, \dots, x_n) \uparrow$ cuando f está indefinida para $(x_1, \dots, x_n)$.

El conjunto de argumentos para los que f está definida se llama dominio de f, notado $dom(f)$.

$$dom(f) = \{(x_1, \dots, x_n) \ : \ f(x_1, \dots, x_n) \downarrow\}$$

f es total si $dom(f) = \mathbb{N}^n$.

3. Funciones Turing computables

Una función parcial $f : \mathbb{N}^n \to \mathbb{N}$ es Turing computable si existe una máquina de Turing determinística $\mathcal{M} = (\Sigma, Q, T, q_0, q_f)$ con $\Sigma = \{*, 1\}$ tal que cuando empieza con una configuración inicial de la forma

$\cdots$	$*$	$*$	$*$	$\overline{x_1}$	$*$	$\overline{x_2}$	$*$	$\cdots$	$*$	$\overline{x_n}$	$*$	$*$	$*$	$\cdots$
											$\uparrow_{q_0}$			

- Si $f(x_1, \dots, x_n) \downarrow$ entonces siguiendo sus instrucciones en T, se llega a una configuración final de la forma

$\cdots$	$*$	$\overline{f(x_1, \dots, x_n)}$	$*$	$\cdots$
			$\uparrow_{q_f}$	

(con la posibilidad de datos residuales adicionales en la cinta).

- Si $f(x_1, \dots, x_n) \uparrow$ entonces nunca termine en el estado q_f.

3.1. Poder de cómputo

Teorema

Sea $f : \mathbb{N}^m \to \mathbb{N}$ una función parcial. Son equivalentes:

1. f es computable en Java.
2. f es computable en C.
3. f es computable en Haskell.
4. f es Turing computable.

No es relevante qué base se utiliza para representar los números, bien podría utilzarse una representación binaria o decimal. De la misma manera, no es relevante si la cinta contiene solo la salida u otras cosas escritas, o inclusive si se utiliza una cinta de entrada y otra de salida, u otra arquitectura distinta.

4. Funciones iniciales

Otra manera de formalizar la idea de función calculable de manera efectiva es la siguiente:

- Empezar por funciones muy simples, efectivas intuitivamente.
- Si mezclamos de alguna manera efectiva dos o más funciones que ya eran efectivas, entonces obtenemos a su vez, una función calculable de manera efectiva.

Definición

Las siguientes funciones se llaman iniciales:

- $s(x) = x + 1$
- $n(x) = 0$
- Proyecciones: $u_i^n(x_1, \ldots, x_n) = x_i$ para $i \in \{1, \ldots, n\}$

5. Composición y Recursión

Definición

Sea $f : \mathbb{N}^k \to \mathbb{N}$ y $g_1, \ldots, g_k : \mathbb{N}^n \to \mathbb{N}$. $h : \mathbb{N}^n \to \mathbb{N}$ se obtiene a partir de f y $g_1, \ldots, g_k$ por composición si

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

Definición

Una clase de funciones $\mathcal{C}$ es cerrada por composición si para cualquier elección de funciones $f, g_1, \ldots, g_k \in \mathcal{C}$, la función h obtenida por composición de ellas está en $\mathcal{C}$.

Definición

$h : \mathbb{N}^{n+1} \to \mathbb{N}$ se obtiene a partir de $g : \mathbb{N}^{n+2} \to \mathbb{N}$ y $f : \mathbb{N}^n \to \mathbb{N}$ por recursión primitiva si

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

(En este contexto, una función 0-aria es un constante k. Si h es 1-aria y $t = 0$, entonces $h(t) = k = s^{(k)}(n(t))$.)

Definición

Una clase de funciones $\mathcal{C}$ es cerrada por recursión primitiva si para cualquier elección de $f, g \in \mathcal{C}$ la h obtenida por recursión primitiva a partir de ellas está en $\mathcal{C}$

6. Clases PRC

Una clase $\mathcal{C}$ de funciones totales es PRC (primitive recursive closed) si

1. Las funciones iniciales están en $\mathcal{C}$.
2. Si una función f se obtiene a partir de otras pertenecientes a $\mathcal{C}$ por medio de composición o recursión primitiva, entonces f también está en $\mathcal{C}$.

Teorema

La clase de funciones totales Turing computables es una clase PRC.

7. Funciones primitivas recursivas

Una función es primitiva recursiva (p.r.) si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de composición y recursión primitiva.

Teorema

f es una funcón p.r. $\iff$ f pertenece a toda clase PRC.

Demostración.

($\Leftarrow$) La clase de funciones p.r. es una clase PRC. Luego, si f pertenece a toda clase PRC, en particular f es p.r.

($\Rightarrow$) Sea f p.r. y sea $\mathcal{C}$ una clase PRC. Como f es p.r., hay una lista de funciones $f_1, f_2, \ldots, f_n$ tal que

- $f = f_n$
- f_i es inicial (con lo cual está en $\mathcal{C}$) o se obtiene por composición o recursión primitiva a partir de funciones f_j con $j < i$ (luego, también está en $\mathcal{C}$).

Entonces, por inducción, todas las funciones de la lista están en $\mathcal{C}$, y en particular, $f_n \in \mathcal{C}$. □

Con esto, la clase de funciones p.r. es la clase PRC más chica.

Corolario

Toda función f p.r. es total y Turing computable.

Demostración. Se sabe que la clase de funciones totales Turing computables es PRC. Por el Teorema previo, si f es p.r., entonces f pertenece a la clase de funciones Turing computables. □

No toda función parcial Turing computable es p.r. porque toda función p.r. es total.

7.1. Ejemplo de función p.r.

La función $suma(x, y) = x + y$ es p.r. porque

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

donde

$$g(x_1, x_2, x_3) = s(u_1^3(x_1, x_2, x_3))$$

Otras funciones preimitivas recursivas son

- $x \cdot y$
- $x!$
- x^y
- $x \dot{-} y = \begin{cases} x - y & \text{si } x \geq y \\ 0 & \text{sino} \end{cases}$
- $\alpha(x) = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{sino} \end{cases}$

7.2. Predicados primitivos recursivos

Los predicados son simplemente funciones que toman valores en $\{0, 1\}$, interpretandose como *falso* y *verdadero* repectivamente.
Los predicados p.r. son aquellos representados por funciones p.r. en $\{0, 1\}$. Por ejemplo, el predicado $x \leq y$ es p.r. porque se puede definir como $\alpha(x \dot{-} y)$.

7.2.1. Operadores lógicos

Teorema

Sea $\mathcal{C}$ una clase PRC. Si p y q son predicados en $\mathcal{C}$, entonces $\neg p$, $p \wedge q$ y $p \vee q$ están en $\mathcal{C}$.

Demostración.

- $\neg p$ se puede definir como $\alpha(p)$.
- $p \wedge q$ se puede definir como $p \cdot q$.
- $p \vee q$ se puede definir como $\neg(\neg p \wedge \neg q)$.

□

Corolario

Si p y q son predicados p.r., entonces también lo son los predicados $\neg p$, $p \wedge q$ y $p \vee q$.

Corolario

Si p y q son predicados totales Turing computables, entonces también lo son los predicados $\neg p$, $p \wedge q$ y $p \vee q$.

7.3. Definición por casos (2)

Teorema

Sea $\mathcal{C}$ una clase PRC. Sean $h, g : \mathbb{N}^n \to \mathbb{N}$ funciones en $\mathcal{C}$ y sea $p : \mathbb{N}^n \to \{0,1\}$ un predicado en $\mathcal{C}$. La función

$$f(x_1,\ldots,x_n) = \begin{cases} g(x_1,\ldots,x_n) & \text{si } p(x_1,\ldots,x_n) \\ h(x_1,\ldots,x_n) & \text{sino} \end{cases}$$

está en $\mathcal{C}$.

Demostración. $f(x_1,\ldots,x_n) = g(x_1,\ldots,x_n)\cdot p(x_1,\ldots,x_n) + h(x_1,\ldots,x_n)\cdot \alpha(p(x_1,\ldots,x_n))$ □

7.4. Definición por casos $(m+1)$

Teorema

Sea $\mathcal{C}$ una clase PRC. Sean $g_1,\ldots,g_m,h : \mathbb{N}^n \to \mathbb{N}$ funciones en $\mathcal{C}$ y sean $p_1,\ldots,p_m : \mathbb{N}^n \to \{0,1\}$ predicados mutuamente excluyentes en $\mathcal{C}$. La función

$$f(x_1,\ldots,x_n) = \begin{cases} g_1(x_1,\ldots,x_n) & \text{si } p(x_1,\ldots,x_n) \\ \vdots & \\ g_m(x_1,\ldots,x_n) & \text{si } p(x_1,\ldots,x_n) \\ h(x_1,\ldots,x_n) & \text{sino} \end{cases}$$

está en $\mathcal{C}$.

8. Sumatorias y productorias

8.1. Sumatorias y Productorias (desde 0)

Teorema

Sea $\mathcal{C}$ una clase PRC. Si $f : \mathbb{N}^{n+1} \to \mathbb{N}$ está en $\mathcal{C}$ entonces también están las funciones

$$g(y,x_1,\ldots,x_n) = \sum_{t=0}^{y} f(t,x_1,\ldots,x_n)$$
$$h(y,x_1,\ldots,x_n) = \prod_{t=0}^{y} f(t,x_1,\ldots,x_n)$$

Demostración. Se tiene que

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

y de forma análoga para h con $\cdot$ en lugar de $+$. □

Es útil notar que no es relevanta la variable sobre la que se hace la recursión, bien se puede definir $g'(x,t)$ como $g(t,x) = g'(u_2^2(t,x),u_1^2(t,x)) = g'(x,t)$.

8.2. Sumatorias y Productorias (desde 1)

Teorema

Sea $\mathcal{C}$ una clase PRC. Si $f : \mathbb{N}^{n+1} \to \mathbb{N}$ está en $\mathcal{C}$ entonces también están las funciones

$$g(y, x_1, \ldots, x_n) = \sum_{t=1}^{y} f(t, x_1, \ldots, x_n)$$

$$h(y, x_1, \ldots, x_n) = \prod_{t=1}^{y} f(t, x_1, \ldots, x_n)$$

Demostración. Se tiene que

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

y de forma análoga para h con $\cdot$ en lugar de $+$ (con 1 en lugar de 0 en el caso base). □

9. Cuantificadores acotados

Los cuantificadores $\forall$ y $\exists$ se leen "para todo" y "existe" respectivamente. Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado, se tiene que

$(\forall t)_{\leq y}\ p(t, x_1, \ldots, x_n)$ es verdadero si y solo si

- $p(0, x_1, \ldots, x_n)$ es verdadero y

  $\vdots$

- $p(y, x_1, \ldots, x_n)$ es verdadero

$(\exists t)_{\leq y}\ p(t, x_1, \ldots, x_n)$ es verdadero si y solo si

- $p(0, x_1, \ldots, x_n)$ es verdadero o

  $\vdots$

- $p(y, x_1, \ldots, x_n)$ es verdadero

Lo mismo se puede definir con $< y$ en lugar de $\leq y$.

9.1. Cuantificadores acotados (con $\leq$)

Teorema

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado perteneciente a una clase PRC $\mathcal{C}$. Los siguientes predicados también están en $\mathcal{C}$:

$$(\forall t)_{\leq y}\ p(t, x_1, \ldots, x_n)$$
$$(\exists t)_{\leq y}\ p(t, x_1, \ldots, x_n)$$

Demostración.

$$(\forall t)_{\leq y}\ p(t, x_1, \ldots, x_n) \iff \prod_{t=0}^{y} p(t, x_1, \ldots, x_n) = 1$$

$$(\exists t)_{\leq y}\ p(t, x_1, \ldots, x_n) \iff \sum_{t=0}^{y} p(t, x_1, \ldots, x_n) \neq 0$$

- La sumatoria y productoria están en $\mathcal{C}$.
- La comparación por $=$ está en $\mathcal{C}$

□

9.2. Cuantificadores acotados (con $<$)

Teorema

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado perteneciente a una clase PRC $\mathcal{C}$. Los siguientes predicados también están en $\mathcal{C}$:

$$(\forall t)_{< y}\ p(t, x_1, \ldots, x_n)$$
$$(\exists t)_{< y}\ p(t, x_1, \ldots, x_n)$$

Demostración.

$$(\forall t)_{< y}\ p(t, x_1, \ldots, x_n) \iff (\forall t)_{< y}\ (t = y \vee p(t, x_1, \ldots, x_n))$$
$$(\exists t)_{< y}\ p(t, x_1, \ldots, x_n) \iff (\exists t)_{< y}\ (t \neq y \wedge p(t, x_1, \ldots, x_n))$$

□

10. Minimización acotada

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado de una clase PRC $\mathcal{C}$

$$g(y, x_1, \ldots, x_n) = \sum_{u=0}^{y} \prod_{t=0}^{u} \alpha(p(t, x_1, \ldots, x_n))$$

Para entender que hace g se tiene que primero pensar en lo siguiente

- Supongamos que existe un $t \leq y$ tal que $p(t, x_1, \ldots, x_n)$ es verdadero
  - Sea t_0 el mínimo t
  - Se tiene que $p(t, x_1, \ldots, x_n) = 0$ para todo $t < t_0$
  - y $p(t_0, x_1, \ldots, x_n) = 1$.
  - Luego con esto, $\prod_{t=0}^{u} \alpha(p(t, x_1, \ldots, x_n)) = \begin{cases} 1 & \text{si } u < t_0 \\ 0 & \text{sino} \end{cases}$
  - y entonces $g(y, x_1, \ldots, x_n) = \overbrace{\underbrace{1 + 1 + \cdots + 1}_{t_0 \text{ veces}} + 0 + 0 + \cdots + 0}^{y+1 \text{ veces}} = t_0$.
  - Esto quiere decir que $g(y, x_1, \ldots, x_n)$ es el mínimo $t \leq y$ tal que $g(t, x_1, \ldots, x_n)$ es verdadero.
- Si no existe tal t, $g(y, x_1, \ldots, x_n) = y + 1$.

Notamos

$$\min_{t \leq y} \; p(t, x_1, \ldots, x_n) = \begin{cases} \text{mínimo } t \leq y \text{ tal que } p(t, x_1, \ldots, x_n) = 1 & \exists t \\ 0 & \text{sino} \end{cases}$$

Teorema

Sea $p : \mathbb{N}^{n+1} \to \{0, 1\}$ un predicado de una clase PRC $\mathcal{C}$. La función

$$\min_{t \leq y} \; p(t, x_1, \ldots, x_n)$$

también está en $\mathcal{C}$.

10.1. Funciones p.r. con minimización acotada

- $x \; div \; y$ es la divisiónentera de x por y

$$\min_{t \leq x}((t + 1) \cdot y > x)$$

(notar que con esta definición $0 \; div \; 0$ es 0)

- $x \; mod \; y$ es el resto de dividir a x por y
- p_n es el n-ésimo primo ($n > 0$). Se define $p_0 = 0, p_2 = 2, p_2 = 3, \ldots$

$$\begin{aligned} p_0 &= 0 \\ p_{n+1} &= \min_{t \leq K(n)} (primo(t) \wedge t > p_n) \end{aligned}$$

Se necesita una cota $K(n)$ que sea lo suficientemente grande y a su vez que sea p.r.; para esto $K(n) = p_n! + 1$ funciona (pues $p_{n+1} \leq p_n! + 1$).

11. Codificación de pares y secuencias

Definimos la función p.r.

$$\langle x, y\rangle = 2^x(2 \cdot y + 1)\dot{-}1$$

(notar que $2^x(2 \cdot y + 1) \neq 0$)

Proposición

Hay una única solución (x, y) a la ecuación $\langle x, y\rangle = z$.

Demostración.

- x es el máximo número tal que $2^x|(z + 1)$.
- $y = ((z + 1)/2^x - 1)/2$

□

11.1. Observadores de pares

Los observadores del par $z = \langle x, y\rangle$ son

- $l(z) = x$
- $r(z) = y$

Proposición

Los observadores de pares son funciones p.r.

Demostración. Como $x, y, < z + 1$ tenemos que

- $l(z) = \text{mín}_{x\leq z}((\exists y)_{\leq z} z = \langle x, y\rangle)$
- $r(z) = \text{mín}_{y\leq z}((\exists x)_{\leq z} z = \langle x, y\rangle)$

□

11.1.1. Ejemplo

- $\langle 2, 5\rangle = 2^2(2 \cdot 5 + 1)\dot{-}1 = 43$
- $l(43) = 2$
- $r(43) = 5$

11.2. Codificación de secuencias

El número de Gödel de la secuencia $a_1, \ldots, a_n$ es el número

$$[a_1, \ldots, a_n] = \prod_{i=1}^{n} p_i^{a_i}$$

donde p_i es el i-ésimo primo ($i \geq 1$).

Por ejemplo, el número de Gödel de la secuencia $1, 3, 3, 2, 2$ es

$$[1, 3, 3, 2, 2] = 2^1 \cdot 3^3 \cdot 5^3 \cdot 7^2 \cdot 11^2 = 40020750.$$

11.2.1. Propiedades de la codificación de secuencias

Teorema

Si $[a_1, \ldots, a_n] = [b_1, \ldots, b_n]$ entonces $a_i = b_i$ para todo $i \in \{1, \ldots, n\}$

Demostración. Por la factorización única en primos, se cumple trivialmente. □

Notar que $[a_1, \ldots, a_n] = [a_1, \ldots, a_n, 0] = [a_1, \ldots, a_n, 0, 0] = \ldots$, pero $[a_1, \ldots, a_n] \neq [0, a_1, \ldots, a_n]$.

11.3. Observadores de secuencias

Los observadores de la secuencia $x = [a_1, \ldots, a_n]$ son

- $x[i] = a_i$
- $|x|$ = longitud de x

Proposición

Los observadores de secuencias son funciones p.r.

Demostración.

- $x[i] = \text{mín}_{t \leq x}(\neg p_i^{t+1} | x)$
- $|x| = \text{mín}_{i \leq x}(x[i] \neq 0 \wedge (\forall j)_{\leq x}(j \leq i \vee x[j] = 0))$

□

11.4. Resumen de codificación de pares y secuencias

Teorema: Codificación de pares

- $l(\langle x, y \rangle) = x$
- $r(\langle x, y \rangle) = y$
- $z = \langle l(z), r(z) \rangle$
- $l(z), r(z) \leq z$
- La codificación de pares y sus observadores son funciones p.r.

Teorema: Codificación de secuencias

- $[a_1, \ldots, a_n][i] = \begin{cases} a_i & \text{si } 1 \leq i \leq n \\ 0 & \text{sino} \end{cases}$
- Si $n \geq |x|$ entonces $[x[1], \ldots, x[n]] = n$
- La codificación de secuencias y sus observadores son funciones p.r.

12. Lenguaje $\mathcal{S}$

El lenguaje de programación $\mathcal{S}$ es igual de poderoso que las máquinas de Turing, pero resulta más fácil de programar, es imperativo y consiste de lo siguiente:

- Variables que almacenan números naturales
  - De entrada: $X_1, X_2, \ldots$
  - Única variable de salida: Y, que comienza inicializada en 0.
  - Temporales: $Z_1, Z_2, \ldots$, comienzan inicializadas en 0.
- Instrucciones, las cuales pueden o no estar etiquetadas:
  - Incremento, la variable V se incrementa en 1:

    $$V \leftarrow V + 1$$

  - Decremento, la variable V se decrementa en 1 si antes era > 0, sino queda en 0:

    $$V \leftarrow V - 1$$

    (se comporta como $V \dot{-} 1$)
  - Salto condicional, donde A es una etiqueta que denota una instrucción del programa.

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

    Si el valor de V es distinto de 0, la ejecución sigue con la primera instrucción que tenga etiqueta A; si el valor de V es 0 sigue con la próxima isntrucción.

Luego, un programa escrito en $\mathcal{S}$ es una sucesión finita de las instrucciones dadas.

12.1. Ejemplos

12.1.1. Ejemplo 1

Programa P

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

El programa P termina cuando $X = 0$ porque no hay siguiente instrucción, y el mismo computa la función $f : \mathbb{N} \to \mathbb{N}$,

$$f(x) = \begin{cases} x & \text{si } x \neq 0 \\ 1 & \text{sino} \end{cases}$$

12.1.2. Ejemplo 2

Este programa computa la función $f : \mathbb{N} \to \mathbb{N},\ f(x) = x$. Cuando intenta ir a E, termina.

$$\begin{array}{ll} [A] & \text{IF } X_1 \neq 0 \text{ GOTO } B \\ & Z_1 \leftarrow Z_1 + 1 \\ & \text{IF } Z_1 \neq 0 \text{ GOTO } E \\ [B] & X_1 \leftarrow X_1 - 1 \\ & Y \leftarrow Y + 1 \\ & Z_1 \leftarrow Z_1 + 1 \\ & \text{IF } Z_1 \neq 0 \text{ GOTO } A \end{array}$$

En el ejemplo, Z_1 solo sirve para un salto incondicional.

12.2. Macros

Dado que $\mathcal{S}$ no tiene saltos incondicionales, se los puede simular con GOTO L, y se lo puede usar como si fuera parte del lenguaje. Cada vez que aparece GOTO L en un programa P, lo reemplazamos con

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

donde V es una variable nueva que no aparece previamente en P. De esta misma forma se puede simular muchas otras pseudoinstrucciones, esto es lo que se llama macro, y se usan como si fueran propias de $\mathcal{S}$; el segmento de programa que la macro abrevia se llama expansión del macro.

12.2.1. Ejemplo 1

En un programa P, la pseudoinstrucicón $V \leftarrow 0$ se expande como

$$\begin{array}{ll} [L] & V \leftarrow V - 1 \\ & \text{IF } V \neq 0 \text{ GOTO } L \end{array}$$

donde L es una etiqueta que no se encontraba en P.

12.2.2. Ejemplo 2

Asignación de variables: $V \leftarrow V'$ ($Y \leftarrow X_1$ en este caso)

$$\begin{array}{ll} & Y \leftarrow 0 \\ [A] & \text{IF } X_1 \neq 0 \text{ GOTO } B \\ & \text{GOTO } C \\ [B] & X_1 \leftarrow X_1 - 1 \\ & Y \leftarrow Y + 1 \\ & Z_1 \leftarrow Z_1 + 1 \\ & \text{GOTO } A \\ [C] & \text{IF } Z_1 \neq 0 \text{ GOTO } D \\ & \text{GOTO } E \\ [D] & Z_1 \leftarrow Z_1 - 1 \\ & X_1 \leftarrow X_1 + 1 \\ & \text{GOTO } C \end{array}$$

- En el primer ciclo se copia el valor de x_1 en Y y en Z_1, hasta dejar a X en cero.
- En el segundo ciclo se pone en X_1 el valor que tenía originalmente hasta que Z_1 es cero.
- Cuando se utiliza la macro GOTO A ,
  no debe expandiser como:

$$\begin{array}{l} Z_1 \leftarrow Z_1 + 1 \\ \text{IF } Z_1 \neq 0 \text{ GOTO } A \end{array}$$

  sino como:

$$\begin{array}{l} Z_2 \leftarrow Z_2 + 1 \\ \text{IF } Z_2 \neq 0 \text{ GOTO } A \end{array}$$

12.2.3. Ejemplo 3

Suma de dos variables:

$$\begin{array}{ll} & Y \leftarrow X_1 \\ & Z_1 \leftarrow X_2 \\ [B] & \text{IF } Z_1 \neq 0 \text{ GOTO } A \\ & \text{GOTO } E \\ [A] & Z_1 \leftarrow Z_1 - 1 \\ & Y \leftarrow Y + 1 \\ & \text{GOTO } B \end{array}$$

Computa la función $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$

$$f(x_1, x_2) = x_1 + x_2.$$

12.2.4. Ejemplo 4

Resta de dos variables:

$$\begin{array}{ll} & Y \leftarrow X_1 \\ & Z_1 \leftarrow X_2 \\ [C] & \text{IF } Z_1 \neq 0 \text{ GOTO } A \\ & \text{GOTO } E \\ [A] & \text{IF } Y \neq 0 \text{ GOTO } B \\ & \text{GOTO } A \\ [B] & Y \leftarrow Y - 1 \\ & Z_1 \leftarrow Z_1 - 1 \\ & \text{GOTO } C \end{array}$$

Computa la función $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$,

$$g(x_1, x_2) = \begin{cases} x_1 - x_2 & \text{si } x_1 \geq x_2 \\ \uparrow & \text{sino} \end{cases}$$

Notese que g es una función parcial, y se la identifica con $\uparrow$ (en el metalenguaje).

El comportamiento de un programa que se indefine es la no terminación, siendo esta la única causa de indefinición.

13. Estado y Descripción instantánea

Un estado de un programa P es una lista de ecuaciones de la forma $V = m$, donde V es una variable y m es un número, tal que hay una ecuación para cada variable que se usa en P, y no hay dos ecuaciones para la misma variable. Por ejemplo, para P:

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

Son estados de P:

- $X_1 = 3,\ Y = 1$
- $X_1 = 3,\ Y = 1,\ Z_1 = 0$
- $X_1 = 3,\ Y = 1,\ Z_1 = 8$
  - No hace falta que sean alcanzados.

No son estados de P:

- $X_1 = 3$
- $X_1 = 3,\ Z_1 = 0$
- $X_1 = 3,\ Y = 1,\ X_1 = 0$

Supongamos que el programa P tiene longitud n. Para un estado σ de P y un $i \in \{1, \ldots, n+1\}$,

- el par (i, σ) es una descripción instantánea de P,
- y el par (i, σ) se llama terminal si $i = n + 1$.

Para un (i, σ) *no terminal*, podemos definir su sucesor (j, τ) como:

1. Si la i-ésima instrucción de P es $V \leftarrow V + 1$.
   - $j = i + 1$
   - τ es identico a σ, salvo que $V = m$ se reemplaza por $V = m + 1$.
2. Si la i-ésima instrucción de P es $V \leftarrow V - 1$.
   - $j = i + 1$
   - τ es identico a σ, salvo que $V = m$ se reemplaza por $V = \text{máx}\{m - 1, 0\}$.
3. Si la i-ésima instrucción de P es IF $V \neq 0$ GOTO L.
   - τ es idéntico a σ

   3.1 Si σ tiene $V = 0$ entonces $j = i + 1$

   3.2 Si σ tiene $V = m$ para $m \neq 0$ entonces
   - Si existe en P una instrucción con etiqueda L entonces

     $$j = \text{mín}\{k \ : \ k\text{-ésima instrucción de } P \text{ que tiene etiqueta } L\}$$
   - Sino, $j = n + 1$ (estado terminal)

14. Cómputo

Un cómputo de un programa P a partir de una descripción instantánea d_1 en una lista $d_1, d_2, \ldots, d_k$ de descripciones instantáneas de P tales que

- d_{i+1} es sucesor de d_i para $i \in \{1, 2, \ldots, k-1\}$.
- d_k es terminal.

14.1. Estados y descripciones iniciales

Sea P un programa y sean $r_1, \ldots, r_m$ números dados.

- El estado inicial de P para $r_1, \ldots, r_m$ es el estado σ_1, que tiene

$$X_1 = r_1,\ X_2 = r_2,\ \ldots\ ,\ X_m = r_m,\ Y = 0$$

  junto con $V = 0$ para cada viarable V que aparezca en P y no sea $X_1, \ldots, X_m, Y$.

- La descripción inicial de P para $r_1, \ldots, r_m$ es $(1, \sigma_1)$.

14.2. Cómputos a partir del estado inicial

Sea P un programa y sean $r_1, \ldots, r_m$ números dados y σ_1 el estado inicial; se tienen dos casos:

- Hay un cómputo de P, $d_1, \ldots, d_k$, tal que $d_1 = (1, \sigma_1)$. Y notamos $\Psi_P^{(m)}(r_1, \ldots, r_m)$ al valor de Y en d_k. En particular, decimos que $\Psi_P^{(m)}(r_1, \ldots, r_m)$ está definido, y se lo nota $\Psi_P^{(m)}(r_1, \ldots, r_m)\downarrow$.

- No hay tal cómputo, i.e. esiste una secuencia infinita $d_1, d_2, d_3, \ldots$. Decimos que $\Psi_P^{(m)}(r_1, \ldots, r_m)$ está indefinido, y se lo nota $\Psi_P^{(m)}(r_1, \ldots, r_m)\uparrow$.

15. Funciones parciales computables

Una función $f : \mathbb{N}^m \to \mathbb{N}$ es $\mathcal{S}$-parcial computable (o simplemente parcial computable) si existe un programa P tal que

$$f(r_1, \ldots, r_m) = \Psi_P^{(m)}(r_1, \ldots, r_m)$$

para todo $(r_1, \ldots, r_m) \in \mathbb{N}^m$. La igualdad del metalenguaje es verdadera si los dos lados están definidos y tienen el mismo valor, o si los dos lados están indefinidos. Se dice que una función f es $\mathcal{S}$-computable (o simplemente computable) si es parcial computable y total.

Un detalle a tener en cuenta es que un mismo programa P puede servir para computar funciones de 1 variable, 2 variables, etc. Por ejemplo, si en P aparecen $X_1, \ldots, X_n$ y no aparece X_j con $j > n$:

- Si solo se especifican $m < n$ variables de entrada, $X_{m+1}, \ldots, X_n$ toman valor 0.

- Si se especifican $m > n$ variables de entrada, P ignora las variables $X_{n+1}, \ldots, X_m$.

16. Minimización no acotada

Definición

La minimización no acotada se define de la siguiente manera

$$\text{mín}_t\ p(t, x_1, \ldots, x_n) = \begin{cases} \text{mínimo } t \text{ tal que } p(t, x_1, \ldots, x_n) \text{ es verdadero} & \text{si existe tal } t \\ \uparrow & \text{sino} \end{cases}$$

Teorema

Si $P : \mathbb{N}^{n+1} \to \{0, 1\}$ es un predicado computable, entonces

$$\text{mín}_t\ p(t, x_1, \ldots, x_n)$$

es parcial computable.

Demostración. El siguiente programa computa $\text{mín}_t\ p(t, x_1, \ldots, x_n)$

$$\begin{array}{ll} [A] & \text{IF } p(Y, X_1, \ldots, X_n) = 1 \text{ GOTO } E \\ & Y \leftarrow Y + 1 \\ & \text{GOTO } A \end{array}$$

□

16.1. Clausura de funciones computables

Teorema

Si h se obtiene a partir de las funciones (parciales) computables $f, g_1, \ldots, g_k$ por composición, entonces h es (parcial) computable.

Demostración. El siguiente programa computa h:

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

Si $f, g_1, \ldots, g_k$ son totales entonces h es total. □

Teorema

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

Demostración. El siguiente programa computa h:

$$\begin{array}{ll} & Y \leftarrow k \\ [A] & \text{IF } X_1 = 0 \text{ GOTO } E \\ & Y \leftarrow g(Z_1, Y) \\ & Z_1 \leftarrow Z_1 + 1 \\ & X_1 \leftarrow X_1 - 1 \\ & \text{GOTO } A \end{array}$$

Si g es total entonces h es total. □

Teorema

La clase de funciones computables es una clase PRC.

Demostración. Por los teoremas previos se sabe que la clase de funciones computables está cerrada por composición y recursión primitiva. Falta ver que las funciones iniciales son computables:

- $s(x) = x + 1$ se computa con el programa

$$Y \leftarrow X_1 + 1$$

- $n(x) = 0$ se computa con el programa vacío.
- $u_i^n(x_1, \ldots, x_n) = x_i$ se computa con el programa

$$Y \leftarrow X_i$$

□

Corolario

Toda función primitiva recursiva es computable.

17. Codificación de programas

Si bien el único tipo de dato en $\mathcal{S}$ son los naturales, otros pueden ser simulados. Por ejemplo:

- Tipo bool: Se lo representa con el 1 (verdadero) y el 0 (falso).
- Tipo par de números naturales: La codificiación y decodificación de pares son funciones primitivas recursivas.
- Tipo entero: Podría ser codificada con un par $\langle$ bool, número natural $\rangle$.
- Tipo secuencias finitas de números naturales: La codificiación y decodificación de secuencias son funciones primitivas recursivas.

Con esto, se puede simular el tipo programa en $\mathcal{S}$. Las instrucciones de $\mathcal{S}$ son

1. $V \leftarrow V + 1$
2. $V \leftarrow V - 1$
3. IF $V \neq 0$ GOTO L'

Y por conveniencia agregamos una cuarta instrucción:

4. $V \leftarrow V$, no hace nada.

Es relevante notar que toda instrucción puede o no estar etiquetada con L, se menciona exactamente una variable V en cada una de ellas, y el IF menciona simpre una etiqueta L'. Para la correcta simulación del tipo, es necesario codificar las variables y etiquetas de la siguiente manera:

$$Y, X_1, Z_1, X_2, Z_2, X_3, Z_3, \ldots \qquad A, B, C, D, \ldots, Z, AA, AB, AC, \ldots, AZ, BA, BB, \ldots, BZ, \ldots$$

A su vez, escribimos $\#(V)$ para la posición que ocupa la variable V en la lista. Idem para $\#(L)$ con la etiqueta L; de forma que $\#(Y) = 1$, $\#(X_2) = 4$, $\#(A) = 1$, $\#(C)$, etc.

Con esto, se puede codificar la instrucción I con $\#(I) = \langle a, \langle b, c\rangle\rangle$ donde

1. Si I tiene etiqueta L, entonces $a = \#(L)$; sino $a = 0$.
2. Si la variable mencionada en I es V entonces $c = \#(V) - 1$.
3. Si la instrucción I es
   - 3.1 $V \leftarrow V$ entonces $b = 0$.
   - 3.2 $V \leftarrow V + 1$ entonces $b = 1$.
   - 3.3 $V \leftarrow V - 1$ entonces $b = 2$.
   - 3.4 IF $V \neq 0$ GOTO L', entonces $b = \#(L') + 2$.

Por ejemplo,

- $\#(X_1 \leftarrow X_1 + 1) = \langle 0, \langle 1, 1\rangle\rangle = \langle 0, 5\rangle = 10$.
- $\#([A] \quad X_1 \leftarrow X_1 + 1) = \langle 1, \langle 1, 1\rangle\rangle = \langle 1, 5\rangle = 21$.
- $\#(\text{IF } X_1 \neq 0 \text{ GOTO } A) = \langle 0, \langle 3, 1\rangle\rangle = \langle 0, 23\rangle = 46$.
- $\#(Y \leftarrow Y) = \langle 0, \langle 0, 0\rangle\rangle = \langle 0, 0\rangle = 0$.

De esta forma todo número representa una única instrucción I. Luego, un programa P no es más que una lista (finita) de instrucciones $I_1, \ldots, I_k$, y codificamos a dicho programa P con

$$\#(P) = [\#(I_1), \ldots, \#(I_k)] - 1$$

Por ejemplo, para el programa P

$$\begin{array}{ll} [A] & X_1 \leftarrow X_1 + 1 \\ & \text{IF } X \neq 0 \text{ GOTO } A \end{array}$$

se tiene

$$\#(P) = [\#(I_1), \#(I_2)] - 1 = [21, 46] - 1 = 2^{21} \cdot 3^{46} - 1$$

17.1. Ambigüedades

Se mencionó que P

$$\begin{array}{ll}[A] & X_1 \leftarrow X_1 + 1 \\ & \text{IF } X \neq 0 \text{ GOTO } A\end{array}$$

tiene número $[21, 46] - 1$. Pero $[21, 46] = [21, 46, 0]$.

Lo cual resultaría en que un mismo número podría representar a más de un programa. En realidad, el programa $[21, 46, 0]$ es

$$\begin{array}{ll}[A] & X_1 \leftarrow X_1 + 1 \\ & \text{IF } X \neq 0 \text{ GOTO } A \\ & Y \leftarrow Y\end{array}$$

que resulta equivalente a P. La forma de eliminar esta ambigüedad es estipulando que la instrucción final de un programa no puede ser $Y \leftarrow Y$. Con esto, cada número representa a un único programa.

Teorema: (Cantor)

El conjunto de las funciones (totales) $\mathbb{N} \to \mathbb{N}$ no es numerable.

Demostración. Supongamos que lo fuera. Procedo a enumerarlas: $f_0, f_1, f_2, \ldots$

$$\begin{array}{lcccccc}
\text{valores de } f_0 & = & f_0(0) & f_0(1) & f_0(2) & f_0(3) & \ldots \\
\text{valores de } f_1 & = & f_1(0) & f_1(1) & f_1(2) & f_1(3) & \ldots \\
\text{valores de } f_2 & = & f_2(0) & f_2(1) & f_2(2) & f_2(3) & \ldots \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \ddots \\
\text{valores de } f_k & = & f_k(0) & f_k(1) & f_k(2) & f_k(3) & \ldots \\
\vdots & & \vdots & \vdots & \vdots & \vdots & \ddots
\end{array}$$

Luego, definiendo la funcíon $g : \mathbb{N} \to \mathbb{N}$, $g(x) = f_x(x) + 1$, se tiene que para todo $f_k \neq g$, lo cual significa que g no está listada. Esto resulta en un absurdo, pues esto era una enumeración de todas las funciones de naturales en naturales, y aunque g se una función de la índole, no está presente. □

Con esto se puede concluir que hay una cantidad no numerable de funciones $\mathbb{N} \to \mathbb{N}$ (i.e. hay más funciones $\mathbb{N} \to \mathbb{N}$ que número naturales). Adicionalmente, se sabe que hay tantos programas como números anturales, y como hay tantas funciones *computables* como números naturales, tiene que haber funciones $\mathbb{N} \to \mathbb{N}$ no computables.

18. Halting problem

El problema de la detención es conocido como el *Halting problem*, y trata de un predicado $HALT(x, y) : \mathbb{N}^2 \to \{0, 1\}$ donde el mismo es verdadero si y solo si el programa con número y, y entrada x no se indefine, i.e.

$$HALT(x, y) = \begin{cases} 1 & \text{si } \Psi_P^{(1)}(x) \downarrow \\ o & \text{sino} \end{cases}$$

donde P es el único programa tal que $\#(P) = y$. Por ejemplo, se P el siguiente *pseudo*programa

$$\begin{array}{ll} & Y \leftarrow 2 \\ [A] & Y \leftarrow Y + 2 \\ & \text{IF } (\exists a)_{\leq Y}(\exists b)_{\leq Y} \ [Primo(a) \wedge Primo(b) \wedge a + b = Y] \text{ GOTO } A \end{array}$$

Supongamos que $\#(P) = e$, se tiene entonces que

$$HALT(x,e)=1 \iff \Psi_P(x)\downarrow \iff \text{la conjetura de Golbach es falsa}$$

Teorema: (Turing, 1936)

$HALT$ no es computable.

Demostración. Probemnos esto por el absurdo, supongamos que $HALT$ es computable. Luego, construimos el siguiente programa Q:

$$[A] \quad \text{IF } HALT(X_1,X_1) \text{ GOTO } A$$

y supongamos que $\#(Q)=e$. Entonces

$$\Psi_Q(x)=\begin{cases}\uparrow & \text{si } HALT(x,x)=1\\ 0 & \text{sino}\end{cases}$$

Entonces se tiene que

$$HALT(x,e)=1 \iff \Psi_Q(x)\downarrow \iff HALT(x,x)\neq 1$$

y se llega a un absurdo con $x=e$. □

19. Diagonalización

En general, la diagonalización sirve para definir una función distinta a muchas otras.
En el caso de $HALT(x,y)$,

- Sea P_i el programa con número i.
- Supongo que $HALT(x,y)$ es computable.
- Defino un función f computable, y la utilizo de la siguiente manera:
  - Veo que $f \notin \{\Psi_{P_0},\Psi_{P_1},\Psi_{P_2},\dots\}$. Para ello me aseguro de que $f(x)\neq\Psi_{P_x}(x)$, y con esto tengo en particular $f(x)\downarrow \iff \Psi_{P_x}(x)\uparrow$ (i.e. no puede ser $\Psi_{P_0}(0)$, no puede ser $\Psi_{P_1}(1)$, no puede ser $\Psi_{P_2}(2)$, etc.); lo cual es absurdo pues se suponía que f era computable.

20. Tesis de Church

Hay muchos modelos de cómputo tales que tienen el mismo poder de cómputo que $\mathcal{S}$ (C, Haskell, máquinas de Turing, etc.). Segúno la tesis de Church, todos los algoritmos para computar en los naturales se puede programar en $\mathcal{S}$. Entonces el *Halting problem* estipula que no hay algoritmo para decidir la verdad o falsedad de $HALT(x,y)$.

21. Programa universal

Para cada $n>0$ definimos

$$\Phi^{(n)}(x_1,\dots,x_n,e)$$

como la salida del programa e con entrada $x_1,\dots,x_n$, lo cual es igual a

$$\Psi_P^{(n)}(x_1,\dots,x_n)$$

donde $\#(P)=e$.

Teorema

Para cada $n>0$ la función $\Phi^{(n)}$ es parcial computable.

Observar que el programa para $\Phi^{(n)}$ es un intérprete de programas. Se trata de un programa que interpreta programas (representados por números).

Demostración. Para demostrar el teorema, es necesario construir el programa U_n que computa $\Phi^{(n)}$.

U_n necesita decodificar e (i.e. averiguar de qué programa P se trata), y llevar una cuenta de los estados de P en cada paso, partiendo del estado inicial de P cuando la entrada es $x_1, \ldots, x_n$.
Dichos estados van a ser codificados con listas, por ejemplo $Y = 0, X_1 = 2, X_2 = 1$ se codifica como $[0, 2, 0, 1] = 63$. En el código de U_n, K indica el número de instrucción que se está por ejecutar (en la simulación de P), y S describe el estado de P en cada momento.

Inicialización

// entrada $= x_1, \ldots, x, e$
// $\#(P) = e = [i_1, \ldots, i_m] - 1$
$Z \leftarrow X_{n+1} + 1$
// $Z = [i_1, \ldots, i_m]$
$S \leftarrow \prod_{j=1}^{n} (p_{2j})^{X_j}$
// $S = [0, X_1, 0, X_2, \ldots, 0, X_n]$ es el estado inicial
$K \leftarrow 1$
// la primera instrucción de P a analizar es la 1

Ciclo principal

// S codifica el estado, K es el número de instrucción
// $Z = [i_1, \ldots, i_m]$
[C] IF $K = |Z| + 1 \vee K = 0$ GOTO F
// si llegó al final, terminar
// sino, sea $Z[K] = i_K 0 \langle a \langle b, c \rangle \rangle$
$U \leftarrow r(Z[K])$
// $U = \langle b, c \rangle$
$P \leftarrow p_{r(U)+1}$
// la variable que aparece en i_K es la $c + 1$-ésima
// P es el primo para la variable que aparece en i_K
IF $l(U) = 0$ GOTO N
// si se trata de una instrucción $V \leftarrow V$ salta a N
IF $l(U) = 1$ GOTO S
// si se trata de una instrucción $V \leftarrow V + 1$ salta a S
// sino, es de la forma $V \leftarrow V - 1$ o IF $V \neq 0$ GOTO L
IF $\neg(P|S)$ GOTO N
// si P no divide a S (i.e. $V = 0$), salta a N
IF $l(U) = 2$ GOTO R
// $V \neq 0$. Si se trata de una isntrucción $V \leftarrow V - 1$, salta a R

Caso IF $V \neq 0$ GOTO L y $V \neq 0$

// sino, continua y se trata de una instrucción IF $V \neq 0$ GOTO L
// $b \geq 2$, por lo tanto L es la $(b-2)$-ésima etiqueta
$K \leftarrow \min_{j \leq |Z|}(l(Z[j]) + 2 = l(U))$
// K pasa a ser la primera instrucción con etiqueta L
// si no hay tal instrucción, $K = 0$ (sale del ciclo)
GOTO C
// vuelve a la primera instrucción del ciclo principal

Caso R (Resta)

$$
\begin{array}{ll}
// & \text{se trata de } V \leftarrow V - 1 \text{ con } V \neq 0 \\
[R] & S \leftarrow S \; div \; P \\
 & \text{GOTO } N \\
// & S = \text{nuevo estado de } P \text{ (resta 1 a } V) \text{ y salta a } N
\end{array}
$$

Caso S (Suma)

$$
\begin{array}{ll}
// & \text{se trata de } V \leftarrow V + 1 \\
[S] & S \leftarrow S \cdot P \\
 & \text{GOTO } N \\
// & S = \text{nuevo estado de } P \text{ (suma 1 a } V) \text{ y salta a } N
\end{array}
$$

Caso N (Nada)

$$
\begin{array}{ll}
// & \text{la instrucción no cambia el estado} \\
[N] & K \leftarrow K + 1 \\
 & \text{GOTO } C \\
// & S \text{ no cambia} \\
// & K \text{ pasa a la siguiente instrucción} \\
// & \text{vuelve al ciclo principal}
\end{array}
$$

Devolución del resultado

$$
\begin{array}{ll}
// & \text{sale del ciclo principal} \\
[F] & Y \leftarrow S[1] \\
// & Y = \text{el valor que toma la variable } Y \text{ de } P \text{ al terminar}
\end{array}
$$

Todo junto

$$
\begin{array}{ll}
 & Z \leftarrow X_{n+1} + 1 \\
 & S \leftarrow \prod_{j=1}^{n} (p_{2j})^{X_j} \\
 & K \leftarrow 1 \\
[C] & \text{IF } K = |Z| + 1 \vee K = 0 \text{ GOTO } F \\
 & U \leftarrow r(Z[K]) \\
 & P \leftarrow p_{r(U)+1} \\
 & \text{IF } l(U) = 0 \text{ GOTO } N \\
 & \text{IF } l(U) = 1 \text{ GOTO } S \\
 & \text{IF } \neg(P|S) \text{ GOTO } N \\
 & \text{IF } l(U) = 2 \text{ GOTO } R \\
 & K \leftarrow \text{mín}_{j \leq |Z|}(l(Z[j]) + 2 = l(U)) \\
 & \text{GOTO } C \\
[R] & S \leftarrow S \; div \; P \\
 & \text{GOTO } N \\
[S] & S \leftarrow S \cdot P \\
 & \text{GOTO } N \\
[N] & K \leftarrow K + 1 \\
 & \text{GOTO } C \\
[F] & Y \leftarrow S[1]
\end{array}
$$

De esta manera, teniendo esta descripción de U_n, para cada $n > 0$, la secuencia

$$\Phi^{(n)}(x_1, \ldots, x_n, 0), \Phi^{(n)}(x_1, \ldots, x_n, 1), \ldots$$

enumera todas las posibles funciones computables de n variables. □

21.1. Notación

A veces se escribe

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = \Phi^{(n)}(x_1, \ldots, x_n, e)$$

y a veces se omite el superíndice cuando $n = 1$

$$\Phi_e(x) = \Phi(x, e) = \Phi^{(1)}(x, e)$$

22. Step Counter y Snapshot

Se define el predicado step counter como

$STP^{(n)}(x_1, \ldots, x_n, e, t)$ $\iff$
el programa e termina en t o menos pasos con entrada $x_1, \ldots, x_n$ $\iff$
hay un cómputo del programa e de longitud $\leq t + 1$, comenzando en con la entrada $x_1, \ldots, x_n$

Teorema

Para cada $n > 0$, el predicado $STP^{(n)}(x_1, \ldots, x_n, e, t)$ es p.r.

Se define la función snapshot como

$SNAP^{(n)}(x_1, \ldots, x_n, e, t)$ $=$
representación de la descripción instantánea del programa e con entrada $x_1, \ldots, x_n$ en el paso t

Teorema

Para cada $n > 0$, la función $SNAP^{(n)}(x_1, \ldots, x_n, e, t)$ es p.r.

22.1. Funciones computables que no son p.r.

Para entender esto es primero remarcar algunos detalles de los temas previos. Los programas de $\mathcal{S}$ pueden ser codificados con constructores y observadores p.r., y su vez se pueden codificar definiciones de funciones p.r. con constructores y observadores p.r.. Por otro lado, existe $\Phi_e^{(n)}(x_1, \ldots, x_n)$ parcial computable que simula al e-ésimo programa con entrada $x_1, \ldots, x_n$, y existe $\tilde{\Phi}_e^{(n)}(x_1, \ldots, x_n)$ computable que simula a la e-ésima función p.r. con entrada $x_1, \ldots, x_n$.

Luego, analicemos $g : \mathbb{N} \to \mathbb{N}$ definida como $g(x) = \tilde{\Phi}_x^{(1)}(x)$. g es trivialmente computable, supongamos entonces que g es p.r.

- Con esto entonces también es p.r. la función $f(x) = g(x) + 1 = \tilde{\Phi}_x^{(1)}(x) + 1$
- Y existe un programa e tal que $\tilde{\Phi}_e = f$.
- De forma que tendríamos $\tilde{\Phi}_e(x) = f(x) = \tilde{\Phi}_x(x) + 1$
- Observemos que el e es fijo, pero la x es variable.
- Y si instanciamos $x = e$, tenemos $\tilde{\Phi}_e(e) = f(e) = \tilde{\Phi}_e(e) + 1$ y se llega a un absurdo.

23. Teorema de la forma normal

Teorema

Sea $f : \mathbb{N}^n \to \mathbb{N}$ una función parcial computable. Entonces existe un predicado p.r. $R : \mathbb{N}^{n+1} \to \mathbb{N}$ tal que

$$f(x_1, \ldots, x_n) = l(\min_z \ R(x_1, \ldots, x_n, z))$$

Demostración. Sea e el número de algún programa para $f(x_1, \ldots, x_n)$. Teniendo en cuenta la representación de una descripción instantanea, el siguiente predicado $R(x_1, \ldots, x_n, z)$ es el buscado:

$$STP^{(n)}(x_1, \ldots, x_n, e, r(z)) \wedge l(z) = r(SNAP^{(n)}(x_1, \ldots, x_n, e, r(z)))[1]$$

Notese que $r(SNAP^{(n)}(x_1, \ldots, x_n, e, r(z)))$ es el estado final de e con entrada $x_1, \ldots, x_n$, y al tomar [1] se está tomando el valor de la variable Y en ese estado final. □

23.1. Caracterización de funciones computables

Teorema

Una función es parcial computable si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de

- composición,
- recursión primitiva y
- minimización

Teorema

Una función es computable si se puede obtener a partir de las funciones iniciales por un número finito de aplicaciones de

- composición,
- recursión primitiva y
- minimización propia (del tipo $\min_t \ q(x_1, \ldots, x_n, t)$ donde siempre existe al menos un t tal que $q(x_1, \ldots, x_n, t)$ es verdadero)

23.2. Eliminación de variables de entrada

Consideremos un programa P que usa la entrada X_1 y X_2, el cual computa la función $f : \mathbb{N}^2 \to \mathbb{N}$, de forma que $f(x, y) = \Psi_P^{(2)}(x, y)$.

$$\begin{array}{ll} \text{INSTRUCCIÓN 1} & // \#(I_1) \\ \vdots & \\ \text{INSTRUCCIÓN k} & // \#(I_k) \end{array}$$

Y se tiene que $\#(P) = [\#(I_1), \ldots, \#(I_k)] - 1$.

Luego, busco el número de programa P_0 para $f_0 : \mathbb{N} \to \mathbb{N}$, $f_0(x) = f(x, 0)$, que computa la función f_0 tal que $f_0(x) = \Psi_{P_0}^{(1)}(x)$

$$\begin{array}{lll} [A] & X_2 \leftarrow X_2 - 1 & // \ 109 \\ & \text{IF } X_2 \neq 0 \text{ GOTO } A & // \ 110 \\ & \text{INSTRUCCIÓN 1} & // \#(I_1) \\ & \vdots & \\ & \text{INSTRUCCIÓN k} & // \#(I_k) \end{array}$$

de forma que $\#(P_0) = [109, 110, \#(I_1), \ldots, \#(I_k)] - 1$. Para esto se supone que A no aparece como etiqueta en P, caso contrario se elige otro nombre de etiqueta.

Esto se puede repetir para otros valores: Busco el número de programa P_1 para $f_1 : \mathbb{N} \to \mathbb{N}$, $f_1(x) = f(x, 1)$, que computa la función f_1 tral que $f_1(x) = \Psi_{P_1}^{(1)}(x)$

$$\begin{array}{lll} [A] & X_2 \leftarrow X_2 - 1 & // \ 109 \\ & \text{IF } X_2 \neq 0 \text{ GOTO } A & // \ 110 \\ & X_2 \leftarrow X_2 + 1 & // \ 26 \\ & \text{INSTRUCCIÓN 1} & // \#(I_1) \\ & \vdots & \\ & \text{INSTRUCCIÓN k} & // \#(I_k) \end{array}$$

de forma que $\#(P_1) = [109, 110, 26, \#(I_1), \ldots, \#(I_k)] - 1$.
O para 2, busco el número de programa P_2 para $f_2 : \mathbb{N} \to \mathbb{N}$, $f_2(x) = f(x, 2)$, que computa la función f_2 tral que $f_2(x) = \Psi_{P_2}^{(1)}(x)$

$$\begin{array}{lll} [A] & X_2 \leftarrow X_2 - 1 & // \ 109 \\ & \text{IF } X_2 \neq 0 \text{ GOTO } A & // \ 110 \\ & X_2 \leftarrow X_2 + 1 & // \ 26 \\ & X_2 \leftarrow X_2 + 1 & // \ 26 \\ & \text{INSTRUCCIÓN 1} & // \#(I_1) \\ & \vdots & \\ & \text{INSTRUCCIÓN k} & // \#(I_k) \end{array}$$

de forma que $\#(P_2) = [109, 110, 26, 26, \#(I_1), \ldots, \#(I_k)] - 1$.

24. Teorema del parámetro

La idea en custión, gira en torno a que hay un programa P_{x_2} para la función $f_{x_2}(x_1) = f(x_1, x_2)$. Y la transformación $(x_2, \#(P)) \mapsto \#(P_{x_2})$ es p.r., es decir, existe una función $S : \mathbb{N}^2 \to \mathbb{N}$ p.r. tal que dado x_2 e

$y = \#(P)$ calcula $\#(P_{x_2})$. Dicha función, es la siguiente

$$S(x_2, y) = (2^{109} \cdot 3^{110} \cdot \prod_{j=1}^{x_2} p_{j+2}^{26} \cdot \prod_{j=1}^{|y+1|} p_{j+x_2+2}^{(y+1)[j]}) - 1$$

Teorema

Hay una función p.r. $S : \mathbb{N}^2 \to \mathbb{N}$ tal que

$$\Phi_y^{(2)}(x_1, x_2) = \Phi_{S(x_2,y)}^{(1)}(x_1)$$

Teorema

Para cada $n, m > 0$ hay una función p.r. inyectiva $S_m^n : \mathbb{N}^{n+1} \to \mathbb{N}$ tal que

$$\Phi^{(m+n)}(x_1, \ldots, x_m, u_1, \ldots, u_n, y) = \Phi_y^{(m+n)}(x_1, \ldots, x_m, u_1, \ldots, u_n) =$$

$$\Phi_{S_m^n(u_1,\ldots,u_n,y)}^{(m)}(x_1, \ldots, x_m) = \Phi^{(m)}(x_1, \ldots, x_m, S_m^n(u_1, \ldots, u_n, y))$$

25. Teorema de la recursión y aplicaciones

Teorema

Si $g : \mathbb{N}^{n+1} \to \mathbb{N}$ es parcial computable, exite un e tal que

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Demostración. Sea S_n^1 la función del Teorema del Parámetro:

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

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

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

(siendo d el programa que computa tal g). Como d está fijo y v es variable, elegimos $v = d$ y $e = S_n^1(d, d)$. □

Corolario

Si $g : \mathbb{N}^{n+1} \to \mathbb{N}$ es parcial computable, existen infinitos e tal que

$$\Phi_e^{(n)}(x_1, \ldots, x_n) = g(e, x_1, \ldots, x_n)$$

Demostración. En la demostración del teorema anterior, existen infinitos d tal que

$$\Phi_d^{(n+1)} = g(S_n^1(v, v), x_1, \ldots, x_n).$$

$v \mapsto S_n^1(v, v)$ es inyectiva de modo que existen infinitos $e = S_n^1(d, d)$. □

25.1. Quines

Un quine es un programa que cuando se ejecuta, devuelve como salida el mismo programa. Por ejemplo

```
char* f = "char* f = %c%s%c;main(){printf(f,34,f,34,10);}%c";
main(){printf(f,34,f,34,10);}
```

Proposición

Hay infinitos e tal que $\Phi_e(x) = e$.

Demostración. Considerar la función $g : \mathbb{N}^2 \to \mathbb{N}$, $g(z, x) = z$. Aplicando el Teorema de la Recursión, existen infinitos e tales que

$$\Phi_e(x) = g(e, x) = e.$$

□

Proposición

Hay infinitos e tal que $\Phi_e(x) = h(e)$.

Demostración. Considerar la función $g : \mathbb{N}^2 \to \mathbb{N}$, $g(z, x) = h(z)$. Aplicando el Teorema de la Recursión, existen infinitos e tal que

$$\Phi_e(x) = g(e, x) = h(e).$$

□

26. Teorema del punto fijo

Teorema

Si $f : \mathbb{N} \to \mathbb{N}$ es computable, existe un e tal que $\Phi_{f(e)} = \Phi_e$

Demostración. Considerar la función $g : \mathbb{N}^2 \to \mathbb{N}$, $g(z, x) = \Phi_{f(z)}(x)$. Aplicando el Teorema de la Recursión, existe un e tal que para todo x,

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

□

27. Conjuntos computablemente enumerables (c.e.)

Cuando se habla de un conjuntos de naturales A, se piensa siempre en la función característica de ese conjunto:

$$A(x) = \begin{cases} 1 & \text{si } x \in A \\ 0 & \text{sino} \end{cases}$$

Así, un conjunto puede ser:

- computable
- primitivo recursivo

Teorema

Sean A, B conjuntos de una clase PRC $\mathcal{C}$. Entonces $A \cup B$, $A \cap B$ y $\overline{A}$ están en $\mathcal{C}$.

De la misma forma que sucede con las funciones, se tienen conjuntos computables, por ejemplo

$$\emptyset,\ \mathbb{N},\ \{p\ :\ p \text{ es primo}\}$$

y hay conjuntos no computables, por ejemplo

$$\{\langle x, y\rangle\ :\ HALT(x, y)\},\ \{\langle x, \langle y, z\rangle\rangle\ :\ \Phi_x(y) = z\}$$

Definición

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

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

es decir,

- Podemos decidir algoritmicamente si un elemento sí pertenece a A, pero dicho algoritmo se indefine para elementos que no pertenecen a A.
- Dicho algoritmos se llaman de semi-decisión: resuelven una aproximación al problema de decidir la pertenencia de un elemento al cojnjunto A.

27.1. Propiedades de los conjuntos c.e.

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

Teorema

Si A es computable entonces A es c.e.

Demostración. Sea P_A un programa para (la función característica de) A. Consideremeos el siguiente programa P

$$[C] \quad \text{IF } P_A(X_1) = 0 \text{ GOTO } C$$

Tenemos entonces que

$$\Psi_P(x) = \begin{cases} 0 & \text{si } x \in A \\ \uparrow & \text{sino} \end{cases}$$

y por lo tanto

$$A = \{x \ : \ \Psi_P(x) \downarrow\}$$

□

Teorema

Si A y B son c.e. entonces $A \cup B$ y $A \cap B$ también son c.e.

Demostración. Sean $A = \{x \ : \ \Phi_p(x)\downarrow\}$ y $B = \{x \ : \ \Phi_q(x)\downarrow\}$
donde A y B son los dominios de las p-ésimas y q-ésimas funciones computables respectivamente.

- $(A \cap B)$: El siguiente programa R tiene como dominio a $A \cap B$:

$$\begin{array}{l} Y \leftarrow \Phi_p(x) \\ Y \leftarrow \Phi_q(x) \end{array}$$

  De esta forma, $\Psi_R(x)\downarrow \iff \Phi_p(x)\downarrow \wedge \Phi_q(x)\downarrow$

- $(A \cup B)$: El siguiente programa R' tiene como dominio a $A \cup B$:

$$\begin{array}{ll} [C] & \text{IF } STP^{(1)}(X_1, p, T) = 1 \text{ GOTO } E \\ & \text{IF } STP^{(1)}(X_1, q, T) = 1 \text{ GOTO } E \\ & T \leftarrow T + 1 \\ & \text{GOTO } C \end{array}$$

  De esta forma, $\Psi_{R'}(x)\downarrow \iff \Phi_p(x)\downarrow \vee \Phi_q(x)\downarrow$

□

Teorema

A es computable $\iff$ A y $\overline{A}$ son c.e.

Demostración.

($\Rightarrow$) Si A es computable, es c.e., y como A es computable entonces $\overline{A}$ es computable, con lo que $\overline{A}$ es c.e.

($\Leftarrow$) Tenemos que A y $\overline{A}$ son c.e., definidos como

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

consideremos el programa P:

$$\begin{array}{ll} [C] & \text{IF } STP^{(1)}(X_1, p, T) = 1 \text{ GOTO } F \\ & \text{IF } STP^{(1)}(X_1, q, T) = 1 \text{ GOTO } E \\ & T \leftarrow T + 1 \\ & \text{GOTO } C \\ [F] & Y \leftarrow 1 \end{array}$$

Con esto tenemos que para cada x, $x \in A$ o bien $x \in \overline{A}$. Entonces Ψ_P computa A.

□

28. Teorema de la enumeración

Definimos $W_n = \{x \ : \ \Psi_n(x)\downarrow\}$ como el dominio del n-ésimo programa

Teorema

Un conjunto A es c.e. $\iff$ existe un n tal que $A = W_n$

A su vez, existe una numeración de todos los conjuntos c.e. $W_1,\ W_2,\ \dots$

28.1. Problema de la detención (visto como conjunto)

Definimos $K = \{n \ : \ n \in W_n\}$, con esto se puede observar

$$n \in W_n \iff \Psi_n(n)\downarrow \iff HALT(n,n)$$

Teorema

K es c.e. pero no computable.

Demostración. Por un lado, dado que la función $n \mapsto \Phi(n,n)$ es parcial computable, se tiene que K es c.e. Por otro lado, supongamos que K fuera computable. Con esto se tiene que $\overline{K}$ también lo sería, y por ser computable, también es c.e. De forma que por el teorema previo, existe e tal que $\overline{K} = W_e$. Por lo tanto:

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

Con lo que se llega a un absurdo pues $W_e = \overline{K}$.

□

Teorema

Si A es c.e., existe un predicado p.r. $R : \mathbb{N}^2 \to \mathbb{N}$ tal que $A = \{x \ : \ (\exists t)\ R(x,t)\}$

Demostración. Sea $A = W_e$. Es decir, $A = \{x \ : \ \Phi_e(x)\downarrow\}$.
Entonces $x \in A$ cuando para algún valor de t (en este caso representando algún tiempo t), el programa e con entrada x termina, i.e.

$$A = \{x \ : \ (\exists t)\ STP^{(1)}(x,e,t)\}$$

siendo $STP^{(1)}(x,e,t) = R(x,t)$.

□

Teorema

Si $A \neq \emptyset$ es c.e., existe una función p.r. $f : \mathbb{N} \to \mathbb{N}$ tal que

$$A = \{f(0), f(1), f(2), \dots\}$$

Demostración. Por el teorema anterior, existe P p.r. tal que $A = \{x \ : \ (\exists t)\ P(x,t)\}$.
Sea $a \in A$, y defino

$$f(u) = \begin{cases} l(u) & \text{si } P(l(u), r(u)) \\ a & \text{sino} \end{cases}$$

Si $x \in A$, existe un t tal que $P(x,t)$, de forma que con esta definición $f(\langle x,t\rangle) = x$. Teniendo esto en cuenta, sea x tal que $f(u) = x$ para algún u; con ello $x = a$ o bien u es de la forma $u = \langle x,t\rangle$, con $P(x,t)$. Luego $x \in A$. □

Teorema

Si $f : \mathbb{N} \to \mathbb{N}$ es parcial computable, $A = \{f(x) \ : \ f(x)\downarrow\}$ es c.e.

Demostración. Sea $\Phi_p = f$, y definamos el programa Q de la forma

$$
\begin{array}{ll}
[A] & \text{IF } STP^{(1)}(Z,p,T) = 0 \text{ GOTO } B \\
 & \text{IF } \Phi_p(Z) = X \text{ GOTO } E \\
[B] & Z \leftarrow Z + 1 \\
 & \text{IF } Z \leq T \text{ GOTO } A \\
 & T \leftarrow T + 1 \\
 & Z \leftarrow 0 \\
 & \text{GOTO } A
\end{array}
$$

Notar que $\Psi_Q(X)\downarrow$ si existen Z, T tales que $Z \leq T$, $X = f(Z)$, y $STP^{(1)}(Z,p,T)$ sea verdadero (i.e. el programa para f termina en T o menos pasos con entrada Z). Con ello

$$\Psi_Q(x) = \begin{cases} 0 & \text{si } x \in A \\ \uparrow & \text{sino} \end{cases}$$

Luego, A es c.e.

□

28.2. Caracterización de los conjuntos c.e.

Teorema

Si $A \neq \emptyset$, son equivalentes:

1. A es c.e.
2. A es el rango de una función primitiva recursiva.
3. A es el rango de una función computable.
4. A es el rango de una función parcial compuable.

Demostración.

- $(1 \Rightarrow 2)$ penúltimo teorema visto.
- $(2 \Rightarrow 3)$ trivial.
- $(3 \Rightarrow 4)$ trivial.
- $(4 \Rightarrow 1)$ último teorema visto.

□

29. Teorema de Rice y aplicaciones

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

Teorema

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

Demostración. Supongamos $\mathcal{C}$ tal que $A = \{x \ : \ \Phi_x \in \mathcal{C}\}$ computable. Sean

- $f \in \mathcal{C}$
- $g \notin \mathcal{C}$

funciones parciales computables. Y sea $h : \mathbb{N}^2 \to \mathbb{N}$ la siguiente función parcial computable:

$$h(t,x) = \begin{cases} g(x) & \text{si } t \in A \\ f(x) & \text{sino} \end{cases}$$

Por el teorema de la recursión, existe e tal que tomando $t = e$, se tiene $\Phi_e(x) = h(e,x)$, luego

- $e \in A \Rightarrow \Phi_e = g \Rightarrow \Phi_e \notin \mathcal{C} \Rightarrow e \notin A$
- $e \notin A \Rightarrow \Phi_e = f \Rightarrow \Phi_e \in \mathcal{C} \Rightarrow e \in A$

□

Este teorema da una fuente de conjuntos no computables:

- $\{x \ : \ \Phi_x$ es total$\}$
- $\{x \ : \ \Phi_x$ es creciente$\}$
- $\{x \ : \ \Phi_x$ tiene dominio infinito$\}$
- $\{x \ : \ \Phi_x$ es primitiva recursiva$\}$

29.1. Ejemplos de conjuntos no c.e.

- $\overline{K} = \{x \ : \ \Phi_x(x) \uparrow\}$ no es c.e.
  - K es c.e., de modo que si $\overline{K}$ lo fuera, K sería computable.
- $Tot = \{x \ : \ \Phi_x$ es total$\}$ no es c.e.
  - Esto se puede ver por una diagonalización simple. Supongamos que Tot es c.e. De esta forma existe f computable tal que $Tot = \{f(0), f(1), f(2), \dots\}$, y existe e tal que $\Phi_e(x) = \Phi_{f(x)}(x) + 1$; lo cual está definido, pues $f(x)$ computa un índice que está en Tot, de modo que Φ_e es total, y $e \in Tot$.
  - Esto implica que existe u tal que $f(u) = e$; de forma que $\Phi_{f(u)}(x) = \Phi_{f(x)}(x) + 1$, y tomando $x = u$ se llega a que $\Phi_{f(u)}(u) = \Phi_{f(u)}(u) + 1$, lo cual es absurdo.
- $\overline{Tot} = \{x \ : \ \Phi_x$ no es total$\}$ no es c.e.
  - Para esto supongamos que $\overline{Tot}$ es c.e.; con esto existe d tal que $\overline{Tot} = dom(\Phi_d)$. Luego, definimos el siguiente programa P:

$$\begin{array}{ll} [C] & \text{IF } STP^{(1)}(X,d,T) = 1 \text{ GOTO } B \\ & T \leftarrow T + 1 \\ & \text{GOTO } C \end{array}$$

tal que

$$\Psi_P^{(2)}(x,y) = g(x,y) = \begin{cases} \uparrow & x \notin dom(\Phi_d) \text{ i.e. } \Phi_x \text{ es total} \\ 0 & \text{sino} \end{cases}$$

Función que se indefine si $x \notin dom(d)$, es decir, no está en $\overline{Tot}$.

- Luego, por el teorema de la recursión, sea e tal que $\Phi_e(y) = g(e,y)$, con esto se tiene:
  - Φ_e es total $\Rightarrow g(e,y) \uparrow$ para todo y y como $\Phi_e(y) = g(e,y)$, esto implica que $\Rightarrow \Phi_e(y) \uparrow$ para todo $y \Rightarrow \Phi_e$ no es total.
  - Φ_e no es total $\Rightarrow g(e,y) = 0$ para todo y y como $\Phi_e(y) = g(e,y)$, esto implica que $\Phi_e(y) = 0$ para todo $y \Rightarrow \Phi_e$ es total.

30. Lenguaje de la lógica proposicional

El lenguaje P se compone de

- Símbolos: $p\ '\ \neg\ \rightarrow\ (\ )$
  - donde $p, p', p'', p''', \ldots$ son símbolos proposicionales.
- Fórmulas:
  1. Todo símbolo proposicional es una fórmula.
  2. Si φ es una fórmula entonces $\neg\varphi$ es una fórmula.
  3. Si φ y ψ son fórmulas entonces $(\varphi \rightarrow \psi)$ es una fórmula.
  4. Nada más es una fórmula.
- Convenciones:
  - Escribimos q por p', r por p'', s por p'', $\ldots$
  - Escribimos $(\varphi \vee \psi)$ en lugar de $(\neg\varphi \rightarrow \psi)$
  - Escribimos $(\varphi \wedge \psi)$ en lugar de $\neg(\varphi \rightarrow \neg\psi)$
  - Escribimos φ en lugar de (φ) cuando convenga
- Llamamos PROP al conjunto de todos los símbolos proposicionales.
- Llamamos FORM al conjunto de todas las fórmulas.

31. Semántica

Una interpretación es una función $v : PROP \rightarrow \{0,1\}$, donde a v se la llama también valuación. Con esto, definimos la noción de verdad de una fórmula para una valuación.

Si $\varphi \in FORM$ y v es una valuación, notamos:

- $v \vDash$ si φ es verdadera para v
- $v \nvDash$ si φ es falsa para v

La definición de $\vDash$ es recursiva:

1. Si $p \in PROP$, $v \vDash p \iff v(p) = 1$
2. $v \vDash \neg\psi \iff v \nvDash \psi$
3. $v \vDash (\psi \rightarrow \rho) \iff v \nvDash \psi \vee v \vDash \rho$

Observar que por esta convención se tiene también:

4 $v \vDash (\psi \wedge \rho) \iff v \vDash \psi \wedge v \vDash \rho$

5 $v \vDash (\psi \vee \rho) \iff v \vDash \psi \vee v \vDash \rho$

31.1. Ejemplos

Por ejemplo, si $v(p) = 1$, $v(q) = 0$, $v(r) = 1$, se tiene que

- $v \vDash (p \to r)$
- $v \vDash (q \to r)$
- $v \nvDash \neg p$
- $v \nvDash (p \wedge q)$

32. Tautologías y métodos de decisión

Una fórmula φ es una tautología ($\vDash \varphi$) si φ es verdadera para toda interpretación, i.e. para toda valuación v, $v \vDash \varphi$.

Proposición

Sea $\varphi \in FORM$ y sean v y w son dos valuaciones tales que $v(p) = w(p)$ para toda variable proposicional que aparece en φ. Entonces $v \vDash \varphi \iff w \vDash \varphi$.

A su vez, decimos que φ es una contradicción si $v \nvDash \varphi$ para toda valuación v, y que φ es una contingencia si existen valuaciones v_1 y v_2 tales que $v_1 \vDash \varphi$ pero $v_2 \nvDash \varphi$.

Existe un método de decisión para saber si φ es tautología o no:

- Supongamos que φ tiene variables proposcionales $p_1, \ldots, p_n$.
- Sea $\mathcal{P}(\{p_1, \ldots, p_n\}) = \{V_1, \ldots, V_{2^n}\}$
- donde para $i \in \{1, \ldots, 2^n\}$ definimos $v_i(p) = \begin{cases} 1 & \text{si } p \in V_i \\ 0 & \text{sino} \end{cases}$
- Con esto, φ es tautología $\iff$ $v_i \vDash \varphi$ para todo $i \in \{1, \ldots, 2^n\}$

32.1. Conjuntos adecuados

Una función booleana es una función $f : \{0,1\}^n \to \{0,1\}$, donde n es algún número natural. Las funciones booleanas son una manera de formalizar la idea de tabla de verdad.

Decimos que un conjunto de conectivos es adecuado si, dada cualquier función booleana f, podemos escribir una fórmula que use sólo esos conectivos y cuya tabla de verdad se corresponda con la tabla de f.

Proposición

El conjunto de conectivo $\{\vee, \wedge, \neg\}$ es adecuado.

33. Consecuencia semántica y conjunto satisfacible

Sea $\Gamma \subseteq FORM$ y $\varphi \in FORM$. Decimos que φ es consecuencia semántica de Γ ($\Gamma \vDash \varphi$) si para toda interpretación v:

$$\underbrace{\text{si } v \vDash \psi \text{ para toda } \psi \in \Gamma,}_{\text{lo notamos } v \vDash \Gamma} \text{ entonces } v \vDash \varphi$$

Decimos que Γ es satisfacible si existe una interpretación v tal que $v \vDash \psi$ para toda $\psi \in \Gamma$ (i.e. tal que $v \vDash \Gamma$)

33.1. Ejemplos

- $\{q\} \vDash q$
- $\{q\} \vDash p \to q$
- $\{r, p \vee q\} \nvDash p$
- $\{r, p \vee q\} \nvDash s$
- $\emptyset$ es satisfacible
- $\{p, q\}$ es satisfacible
- $\{\neg p, p \wedge q\}$ no es satisfacible
- $\{p, p \to q, \neg q\}$ no es satisfacible

33.2. Algunos resutlados sobre $\vDash$

Proposición

1. $\emptyset \vDash \varphi \iff \vDash \varphi$ (i.e. φ es tautología)
2. si $\vDash \varphi$ entonces $\Gamma \vDash \varphi$
3. $\{\varphi\} \vDash \varphi$
4. si $\Gamma \subseteq \Delta$ y $\Gamma \vDash \varphi$ entonces $\Delta \vDash \varphi$
5. si $\Gamma \vDash \varphi$ y $\Gamma \vDash \varphi \to \psi$ entonces $\Gamma \vDash \psi$

Demostración. Item 5.

- Sea v una interpretación tal que $v \vDash \Gamma$
- Sabemos que $v \vDash \varphi$
- y sabemos que $v \vDash \varphi \to \psi$
- Con esto conlucimos que $v \vDash \psi$

□

33.3. Conjunto independiente

Un conjunto de fórmulas Γ se dice independiente si para toda $\varphi \in \Gamma$, $\varphi \notin con(\Gamma \setminus \{\varphi\})$.

34. Sistema axiomático *SP*

SP es un mecanismo deductivo que se compone de:

- Axiomas: Sean $\varphi, \psi, \rho \in FORM$
  - SP1 $\varphi \to (\psi \to \varphi)$
  - SP2 $(\varphi \to (\psi \to \rho)) \to ((\varphi \to \psi) \to (\varphi \to \rho))$
  - SP3 $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$
- Una regla de inferencia
  - MP Sean $\varphi, \psi \in FORM$. ψ es una consecuencia inmediata de $\varphi \to \psi$ y φ.

Una demostración de φ en *SP* es una cadena finita y no vacía $\varphi_1, \ldots, \varphi_n$ de fórmulas de P tal que $\varphi_n = \varphi$ y

- φ_i es un axioma o
- φ_i es una consecuencia inmediata de φ_k y φ_l, donde $k, l < i$.

En este caso, decimos que φ es un teorema, y lo notamos $\vdash \varphi$.

34.1. Ejemplo: Demostración de $p \to p$

Utilizando los axiomas y reglas de inferencia de SP, se puede realizar una demostración de $p \to p$ en P de la siguiente manera:

1. $p \to ((p \to p) \to p)$ — $SP1$
2. $(p \to ((p \to p) \to p)) \to ((p \to (p \to p)) \to (p \to p))$ — $SP2$
3. $(p \to (p \to p)) \to (p \to p)$ — MP 1 y 2
4. $p \to (p \to p)$ — $SP1$
5. $p \to p$ — MP 3 y 4

De esta manera se concluye que $\vdash p \to p$ (i.e. $p \to p$ es un teorema).

35. Consecuencia sintáctica

Sea $\Gamma \subseteq FORM$ y $\varphi \in FORM$. φ es una consecuencia sintáctica de Γ ($\Gamma \vdash \varphi$) si existe una cadena finita y no vacía $\varphi_1, \ldots, \varphi_n$ de fórmulas de P tal que $\varphi_n = \varphi$ y

- φ_i es un axioma o
- $\varphi_i \in \Gamma$ o
- φ_i es una consecuencia inmediata de φ_k y φ_l con $k, l < i$.

$\varphi_1, \ldots, \varphi_n$ se llama derivación de φ a partir de Γ, donde Γ se llama una teoría. De esta forma, decimos que φ es un teorema de la teoría Γ.

35.1. Correctitud de SP

Teorema

Si $\Gamma \vdash \varphi$ entonces $\Gamma \vDash \varphi$ (i.e. si φ es teorema de la teoría Γ, es válido en toda interpretación de Γ).

Demostración. Supongamos $\Gamma \vdash \varphi$. Es decir, existe una cadena finita y no vacía $\varphi_1, \ldots, \varphi_n$ de fórmulas de P tal que $\varphi_n = \varphi$ y

- φ_i es un axioma o
- $\varphi_i \in \Gamma$ o
- φ_i es una consecuencia inmediata de φ_k y φ_l con $k, l < i$.

Con esto, se puede demostrar que $\Gamma \vDash \varphi$ por inducción en n, donde la propiedad a demostrar es la siguiente:

$$P(n) = \text{“si } \varphi_1, \ldots, \varphi_n \text{ es una derivación de } \varphi \text{ a partir de } \Gamma, \text{ entonces } v \vDash \Gamma \implies v \vDash \varphi\text{”}$$

Demuestro que vale $P(n)$ por inducción en n:

- Caso base: Veo que vale para $P(1)$. Supongo que existe v tal que $v \vDash \Gamma$. Quiero ver que $v \vDash \Gamma \implies v \vDash \varphi$. De forma que hay 2 posibilidades:
  1. φ es axioma de SP: En este caso, $v \vDash \varphi$
  2. $\varphi \in \Gamma$: En este caso, también $v \vDash \varphi$.
- Paso inductivo: Supongo que existe v tal que $v \vDash \Gamma$, y supongo que vale $P(m)$ para todo $m \leq n$. Quiero ver que vale para $P(n+1)$. Luego, sea $\varphi_1, \ldots, \varphi_n, \varphi_{n+1} = \varphi$ una derivación de φ a partir de Γ. Hay 3 posibilidades:
  1. φ es axioma de SP: En este caso, es igual al caso base.
  2. $\varphi \in \Gamma$: También igual al caso base.
  3. φ es consecuencia inmediata de φ_i y $\varphi_j = \varphi_i \to \varphi$ (con $i, j \leq n$). Luego, por hipótesis inductiva ($P(i)$ y $P(j)$), se sabe que $v \vDash \varphi_i$ y $v \vDash \varphi_i \to \varphi$; con esto $v \vDash \varphi$.

□

35.2. Ejemplos

- $\Gamma_1 = \{p\} \vdash p$

  1. p — $p \in \Gamma_1$

- $\Gamma_2 = \{p\} \vdash \varphi \to p$

  1. p — $p \in \Gamma_2$
  2. $p \to (\varphi \to p)$ — $SP1$
  3. $\varphi \to p$ — MP 1 y 2

- $\Gamma_3 = \{p\} \nvdash q$

  pues $\Gamma_3 \nvDash q$ (considerar $v(p) = 1$ y $v(q) = 0$)

- $\Gamma_4 = \{p, \neg p\} \vdash \varphi$

  1. $\neg p \to (\neg\varphi \to \neg p)$ — $SP1$
  2. $\neg p$ — $\neg p \in \Gamma_4$
  3. $\neg\varphi \to \neg p$ — MP 1 y 2
  4. $(\neg\varphi \to \neg p) \to (p \to \varphi)$ — $SP3$
  5. $p \to \varphi$ — MP 3 y 4
  6. p — $p \in \Gamma_4$
  7. φ — MP 5 y 6

35.3. Algunos resultados sobre $\vdash$

Proposición

1. $\emptyset \vdash \varphi \iff \vdash \varphi$ (i.e. φ es teorema)
2. Si $\vdash \varphi$ entonces $\Gamma \vdash \varphi$
3. $\{\varphi\} \vdash \varphi$
4. Si $\Gamma \subseteq \Delta$ y $\Gamma \vdash \varphi$ entonces $\Delta \vdash \varphi$
5. Si $\Gamma \vdash \varphi$ y $\Gamma \vdash \varphi \to \psi$ entonces $\Gamma \vdash \psi$

Si se reemplaza $\vdash$ por $\vDash$ se pueden obtener resultados similares.

36. Conjuntos y sistemas consistentes

Se dice que $\Gamma \subseteq FORM$ es consistente si no existe $\varphi \in FORM$ tal que

$$\Gamma \vdash \varphi \quad \wedge \quad \Gamma \vdash \neg\varphi$$

Se dice que un sistema S es consistente si no existe $\varphi \in FORM$ tal que

$$\vdash_S \varphi \quad \wedge \quad \vdash_S \neg\varphi$$

Teorema

El sistema SP es consistente

Demostración. Sea v una valuación cualquiera. Por correctitud, todo teorema de SP es verdadero para v

$$\vdash \varphi \implies v \vDash \varphi \implies v \nvDash \neg\varphi \implies \nvdash \neg\varphi$$

Es decir, dado un teorema φ de SP, sucede que cualquier valuación lo hace verdadero, de forma que toda valuación hace falso a $\neg\varphi$, con lo que $\neg\varphi$ no es un teorema. De forma que no puede pasar que ambos φ y $\neg\varphi$ sean teoremas a la vez. □

36.1. Notas sobre computabilidad

Se pueden codificar las fórmulas de P con números naturales, donde

- a cada fórmula φ se le asigna un número $\#\varphi > 0$
- cada número positivo representa una única fórmula

Con esto se puede decidir algorítmicamente si una fórmula es un axioma o no. Por ejemplo, se puede computar la función

$$ax(x) = \begin{cases} 1 & \text{si la fórmula de número } x \text{ es un axioma de } SP \\ 0 & \text{sino} \end{cases}$$

Y se puede decidir algorítmicamente si una fórmula es consecuencia inmediata de otras dos, computando la fución

$$mp(x,y,z) = \begin{cases} 1 & \text{si la fórmula de número } z \text{ es consecuencia inmediata de las fórmulas de números } x \text{ e } y \\ 0 & \text{sino} \end{cases}$$

A su vez, las demostraciones al ser listas finitas de fórmulas, se pueden codificar como $[\#\varphi_1, \ldots, \#\varphi_n]$
Con ello se puede decidir se una lista de fñormulas es una demostración válida o no, computando

$$dem(x) = \begin{cases} 1 & x \text{ es una demostración válida} \\ 0 & \text{sino} \end{cases}$$

donde efectivamente:

$$\begin{aligned} dem(x) &= (\forall k \in \{1, \ldots, |x|\})\ [ax(x[k]) \vee cons(x,k)] \\ cons(x,k) &= (\exists i \in \{1, \ldots, k-1\})\ [mp(x[i], x[j], x[k])] \end{aligned}$$

Por último, se puede considerar el siguiente programa P:

$$\begin{array}{ll} [A] & \text{IF } dem(D) = 1 \wedge D[|D|] = X \text{ GOTO } E \\ & D \leftarrow D + 1 \\ & \text{GOTO } A \end{array}$$

P busca un demostración para la fórmula con número X.

- Si la encuentra, se detiene
- caso contrario, se indefine

es decir, $\vdash \varphi \iff \Psi_P(\#\varphi)\downarrow$, o equivalentemente φ es teorema $\iff \#\varphi \in dom(\Psi_P)$

Se tiene entonces también que (en general), cualquier sistema de deducción con un conjunto computable de axiomas y reglas de inferencia computables, tiene un conjunto de teoremas c.e.

37. Teorema de la deducción

Teorema

Si $\Gamma \cup \{\varphi\} \vdash \psi$ entonces $\Gamma \vdash \varphi \to \psi$

Demostración. Esto se puede demostrar por inducción en la longitud de la demostración de $\Gamma \cup \{\varphi\} \vdash \psi$. Supongo que $\varphi_1, \ldots, \varphi_n$ es una derivación de $\psi (= \varphi_n)$ a partir de $\Gamma \cup \{\varphi\}$. Luego,

- Caso base: Sea $\varphi_1, \ldots, \varphi_n$ una derivación de ψ a partir de $\Gamma \cup \{\varphi\}$, y sea $n = 1$ (i.e. la derivación es una sola fórmula $\varphi_1 = \psi$). Con esto, se puede observar que hay 3 posibilidades:

  1. ψ es un axioma de SP:

1.1 ψ	ψ es axioma
1.2 $\psi \to (\varphi \to \psi)$	$SP1$
1.3 $\varphi \to \psi$	$MP1$ y 2

  2. $\psi \in \Gamma$:

2.1 ψ	$\psi \in \Gamma$
2.2 $\psi \to (\varphi \to \psi)$	$SP1$
2.3 $\varphi \to \psi$	$MP1$ y 2

  3. $\psi = \varphi$:
     Dado que $\vdash p \to p$ (probado previamente), se puede demostrar $\varphi \to \varphi$.

- Paso inductivo: Sea $\varphi_1, \ldots, \varphi_n$ una derivación de ψ a partir de $\Gamma \cup \{\varphi\}$, y tomo como hipótesis inductiva el que para toda derivación de ψ' a partir de $\Gamma \cup \{\varphi\}$ de longitud menor a n, se tiene que $\Gamma \vdash \varphi \to \psi'$. Luego, pruebo que vale para n. Hay 4 posibilidades:

  1. ψ es un axioma de SP: Idem caso base.
  2. $\psi \in \Gamma$: Idem caso base.
  3. $\psi = \varphi$: Idem caso base.
  4. ψ se infiere por MP de φ_i y φ_j (con $i, j < n$).
     - Tomamos entonces que $\varphi_j = \varphi_i \to \psi$
     - Luego, $\Gamma \cup \{\varphi\} \vdash \varphi_i$, donde su derivación tiene longitud menor a n. De forma que por hipótesis inductiva, $\Gamma \vdash \varphi \to \varphi_i$.
     - A su vez, $\Gamma \cup \{\varphi\} \vdash \varphi_j$, donde su derivación tiene longitud menor a n. De forma que por hipótesis inductiva, $\Gamma \vdash \varphi \to \varphi_j$.
     - Es decir, $\Gamma \vdash \varphi \to (\varphi_i \to \psi)$.
     - Luego, por $SP2$ se tiene que $\vdash (\varphi \to (\varphi_i \to \psi)) \to ((\varphi \to \varphi_i) \to (\varphi \to \psi))$
     - Aplicando MP entre ■ y luego entre ■ se obtiene que $\Gamma \vdash \varphi \to \psi$.

□

37.1. Conjuntos consistentes

Proposición

1. $\Gamma \cup \{\neg\varphi\}$ es inconsistente $\iff \Gamma \vdash \varphi$
2. $\Gamma \cup \{\varphi\}$ es inconsistente $\iff \Gamma \vdash \neg\varphi$

Demostración. Se demuestra 1, la demostración de 2 es análoga.

($\Leftarrow$) $\Gamma \vdash \varphi \implies \Gamma \cup \{\neg\varphi\} \vdash \varphi$, y a su vez, trivialmente $\Gamma \cup \{\neg\varphi\} \vdash \neg\varphi$, de forma que $\Gamma \cup \{\neg\varphi\}$ es inconsistente.

($\Rightarrow$) Como $\Gamma \cup \{\neg\varphi\}$ es inconsistente, existe ψ tal que $\Gamma \cup \{\neg\varphi\} \vdash \psi$ y $\Gamma \cup \{\neg\varphi\} \vdash \neg\psi$. De forma que por el teorema de la deducción, se tiene que $\Gamma \vdash \neg\varphi \to \psi$ y $\Gamma \vdash \neg\varphi \to \neg\psi$.

Luego, utilizando el teorema $\vdash (\neg\varphi \to \psi) \to ((\neg\varphi \to \neg\psi) \to \varphi)$, y aplicando MP primero entre ■ y luego entre ■, se obtiene que $\Gamma \vdash \varphi$.

□

37.2. Satisfacible $\implies$ Consistente

Teorema

Si $\Gamma \subseteq FORM$ es satisfacible entonces Γ es consistente.

Demostración. Esto se puede probar por el absurdo:
Supongamos que existe una interpretación v tal que $v \vDash \Gamma$ pero Γ es inconsistente, de forma que existe ψ tal que $\Gamma \vdash \psi$ y $\Gamma \vdash \neg\psi$. Por correctitud de SP, consecuencia sintáctica implica consecuencia semántica, es decir $\Gamma \vDash \psi$ y $\Gamma \vDash \neg\psi$. Pero como por hipótesis $v \vDash \Gamma$, se tiene que $v \vDash \psi$ y $v \vDash \neg\psi$.

De forma que se llega a un absurdo que provino de suponer que Γ era inconsistente. □

38. Lema de Lindenbaum

Se dice que $\Gamma \subseteq FORM$ es maximal consistente (m.c.) en SP si

- Γ es consistente y
- para toda fórmula φ
  - $\varphi \in \Gamma$ o
  - existe ψ tal que $\Gamma \cup \{\varphi\} \vdash \psi$ y $\Gamma \cup \{\varphi\} \vdash \neg\psi$

Lema

Si $\Gamma \subseteq FORM$ es consistente, existe Γ' m.c. tal que $\Gamma \subseteq \Gamma'$

Demostración. Esta demostración se basa en obtener Γ' (superconjunto m.c. de Γ).
Se enumeran todas las fórmulas de forma $\varphi_1, \varphi_2, \dots$ y se define

- $\Gamma_0 = \Gamma$
- $\Gamma_{n+1} = \begin{cases} \Gamma_n \cup \{\varphi_{n+1}\} & \text{si } \Gamma_n \cup \{\varphi_{n+1}\} \text{ es consistente} \\ \Gamma_n & \text{sino} \end{cases}$
- $\Gamma' = \bigcup_{i \geq 0} \Gamma_i$

Y se tiene que

1. $\Gamma' \supseteq \Gamma$
2. Cada Γ_i es consistente.
3. Γ' es consistente. Caso contrario, existe ψ tal que $\Gamma' \vdash \psi$ y $\Gamma' \vdash \neg\psi$, donde en ambas derivaciones aparecen únicamente $\{\gamma_1, \dots, \gamma_k\} \subseteq \Gamma'$. Luego, sea j suficientemente grande tal que $\{\gamma_1, \dots, \gamma_k\} \subseteq \Gamma_j$, en este caso se tendría que $\Gamma_j \vdash \psi$ y $\Gamma_j \vdash \neg\psi$, lo cual es absurdo pues contradice el punto 2.
4. Por último, Γ' es maximal. Por ejemplo, supongamos que existe $\varphi \notin \Gamma'$. Debe existir n tal que $\varphi_{n+1} = \varphi$, de forma que $\varphi_{n+1} \notin \Gamma_{n+1}$ y entonces $\Gamma_n \cup \{\varphi_{n+1}\}$ resultaría inconsistente. Luego, $\Gamma' \cup \{\varphi_{n+1}\}$ sería inconsistente, pues $\Gamma_n \subseteq \Gamma'$.

□

38.1. Conjuntos maximales consistentes

Proposición

Si Γ' es m.c. entonces para toda $\varphi \in FORM$, o bien $\varphi \in \Gamma'$ o bien $\neg\varphi \in \Gamma'$.

Demostración. Para empezar, no puede ser que φ y $\neg\varphi$ estén en Γ' porque sería inconsistente, asi que para ver esto partamos de pensar que ninguna de ellas está. Como Γ' es maximal,

- $\Gamma' \cup \{\varphi\}$ es inconsistente, lo cual implica que $\Gamma' \vdash \neg\varphi$, resultando en Γ' inconsistente.

- $\Gamma' \cup \{\neg\varphi\}$ es inconsistente, lo cual implica que $\Gamma' \vdash \varphi$, resultando en Γ' inconsistente.

De forma que o bien $\varphi \in \Gamma'$ o bien $\neg\varphi \in \Gamma'$. □

Proposición

Sea Γ' m.c., $\Gamma' \vdash \varphi \iff \varphi \in \Gamma'$.

38.2. Consistente $\implies$ Satisfacible

Teorema

Si $\Gamma \subseteq FORM$ es consistente, entonces Γ es satisfacible.

Demostración.
Dado Γ consistente, construimos $\Gamma' \supseteq \Gamma$ m.c. (por el lema de Lindenbaum), y definimos la interpretación v tal que $v(p) = 1 \iff p \in \Gamma'$. Veamos que $v \vDash \varphi \iff \varphi \in \Gamma'$ por inducción en la complejidad de φ (i.e. cantidad de $\neg$ o $\to$ que aparecen en φ).

- Caso base: $\varphi = p$, esto resulta trivial por la definición de v.
- Paso inductivo:

  Tomo como hipótesis inductiva $v \vDash \varphi \iff \varphi \in \Gamma'$ para toda φ de complejidad menor a m.

  Luego, sea φ de complejidad m, hay 2 casos para analizar:

  1. $\varphi = \neg\psi$:

     ($\Rightarrow$) $v \vDash \varphi \implies v \nvDash \psi$ donde ψ tiene complejidad $m-1$ (tiene una negación menos que φ), lo cual, por hipótesis inductiva implica que $\psi \notin \Gamma' \implies \neg\psi \in \Gamma' \implies \varphi \in \Gamma'$

     ($\Leftarrow$) $\varphi \in \Gamma' \implies \psi \notin \Gamma'$ lo cual, por hipótesis inductiva implica que $v \nvDash \psi \implies v \vDash \neg\psi \implies v \vDash \varphi$
  2. $\varphi = \psi \to \rho$

     ($\Rightarrow$) Se tiene que $v \vDash \varphi \implies v \vDash (\psi \to \rho)$ lo cual implica que $v \nvDash \psi$ o $v \vDash \rho$.
     - Si $v \nvDash \psi$, por hipótesis inductiva $\psi \notin \Gamma' \implies \neg\psi \in \Gamma' \implies \Gamma' \vdash \neg\psi$. Usando el teorema $\vdash \neg\psi \to (\psi \to \rho)$ y MP, se obtiene que $\Gamma' \vdash \psi \to \rho$. Es decir, $\psi \to \rho \in \Gamma'$.
     - Si $v \vDash \rho$, por hipótesis inductiva $\rho \in \Gamma' \implies \Gamma' \vdash \rho$. Usando $\vdash \rho \to (\psi \to \rho)$ $(SP1)$, junto con MP, se obtiene que $\Gamma' \vdash \psi \to \rho$. Es decir, $\psi \to \rho \in \Gamma'$.

     ($\Leftarrow$) La vuelta se la puede probar por el contrareciproco, de forma que se tiene que $v \nvDash \varphi \implies v \vDash \psi$ y $v \nvDash \rho$, donde ψ y ρ tienen complejidad $m-1$ (tienen una implicación menos que φ), y por hipótesis inductiva, $\psi \in \Gamma'$ y $\rho \notin \Gamma' \implies \neg\rho \in \Gamma'$. Esto implica que $\Gamma' \vdash \psi$ y $\Gamma' \vdash \neg\rho$. Luego, utilizando el teorema $\vdash \psi \to (\neg\rho \to \neg(\psi \to \rho))$ y aplicando MP dos veces sobre los antecedentes, se obtiene que $\Gamma' \vdash \neg(\psi \to \rho)$. Por lo tanto, $\neg(\psi \to \rho) \in \Gamma'$ y entonces $\psi \to \rho \notin \Gamma'$.

□

39. Teorema de Completitud (fuerte) en *SP*

Ya se probo que Γ consisitente $\iff$ Γ satisfacible y que $\Gamma \cup \{\neg\varphi\}$ es inconsistente $\iff \Gamma \vdash \varphi$. De esto surge:

Teorema

Si $\Gamma \vDash \varphi$ entonces $\Gamma \vdash \varphi$ (i.e. consecuencia semántica implica consecuencia sintáctica).

Demostración. Supongamos $\Gamma \vDash \varphi$. Luego $\Gamma \cup \{\neg\varphi\}$ es insatisfacible, y como consistente $\implies$ satisfacible, tenemos el contrarecíproco y $\Gamma \cup \{\neg\varphi\}$ sería inconsistente, de forma que $\Gamma \vdash \varphi$. □

40. Teorema de Compacidad

Teniendo en cuenta el teorema previo, se pueden obtener los siguientes corolarios:

Corolario

$\Gamma \vdash \varphi \iff \Gamma \vDash \varphi$

Corolario

$\vdash \varphi \iff \vDash \varphi$ (i.e. φ es un teorema de $SP \iff$ es tautología).

Con esto se puede enunciar el siguiente teorema (Compacidad):

Teorema

Sea $\Gamma \subseteq FORM$. Si todo subconjunto finito de Γ es satisfacible, entonces Γ es satisfacible.

Demostración. Esto se puede probar por el absurdo.

Supongamos que Γ es insatisfacible, lo que implica que Γ es inconsistente y existe ψ tal que $\Gamma \vdash \psi$ y $\Gamma \vdash \neg\psi$. Para esto, se utilizan solo finitos axiomas de la teoría Γ (porque dichas derivaciones son finitas), de forma que existe $\Delta \subseteq \Gamma$ finito tal que $\Delta \vdash \psi$ y $\Delta \vdash \neg\psi$.

Esto quiere decir que Δ es inconsistente, lo cual implica que Δ es insatisfacible, pero esto contradice la hipótesis de que todo subconjunto finito de Γ es satisfacible, llegando a un absurdo. □

40.1. Resumen del lenguaje P

Semántica	Método deductivo
■ Tautología (verdadera en toda interpretación)	■ Teorema (tiene demostración en SP)
■ Consecuencia semántica $\vDash$	■ Consecuencia sintáctica $\vdash$
■ Conjunto satisfacible (existe un modelo para todos sus elementos)	■ Conjunto consistente (no perimite probar φ y $\neg\varphi$)

40.2. Notas sobre computabiliiad

Se mencionó previamente que el conjunto de teoremas de SP es c.e., y de hecho, el mismo resulta ser computable. Dado que se puede por ejemplo trabajar con un método de decisión como las tablas de verdad, de forma que $\vdash \varphi \iff \vDash \varphi$ se puede interpretar como $\vdash \varphi \iff$ en la tabla de verdad de φ solo hay $1s$ en la última columna.

41. Lenguaje de lógica de primer orden

Los lenguajes de primer orden se componen de símbolos lógicos: $x\ '\ \forall\ \neg\ \rightarrow\ (\ )$, donde $x,\ x',\ x'',\ x''',\ldots$ son variables pertenecientes a VAR (el conjunto de variables), y $\forall$ se llama cuantificador universal.

Se tienen distintos símbolos en cada lenguaje particular $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$, donde:

- $\mathcal{C}$ es un conjunto de símbolos de constantes (puede ser $\mathcal{C} = \emptyset$).
- $\mathcal{F}$ es un conjunto de símbolos de funciones (puede ser $\mathcal{F} = \emptyset$).
- $\mathcal{P}$ es un conjunto de símbolos de predicados ($\mathcal{P} \neq \emptyset$).

42. Términos, fórmulas y variables de un lenguaje

42.1. Términos de $\mathcal{L}$

Para un lenguaje fijo $\mathcal{L}$ se pueden definir los términos de $\mathcal{L}$ como:

1. Toda variable es un término.
2. Todo símbolo de constante de $\mathcal{L}$ es un término.
3. Si f es un símbolo de función n-ádico de $\mathcal{L}$ y $t_1, t_2, t_3, \ldots, t_n$ son término de $\mathcal{L}$, entonces $f(t_1, \ldots, t_n)$ es un término de $\mathcal{L}$.
4. Nada más es un término de $\mathcal{L}$.

TERM($\mathcal{L}$) es el conjunto de todos los términos del lenguaje $\mathcal{L}$.

Se dice que un término es cerrado si no tiene variables. Por ejemplo, para $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$, con $\mathcal{C} = \{c, d\}$, $\mathcal{F} = \{f\}$ y $\mathcal{P} = \{R\}$ (f de aridad 3, R binario) son términos:

$$c, \quad d, \quad x, \quad f(c, d, x'), \quad f(c, f(x''', x'', x''), x')$$

42.2. Fórmulas de $\mathcal{L}$

Para un lenguaje fijo $\mathcal{L}$, se pueden defirnir las fórmulas de $\mathcal{L}$ como:

1. Si P es un símbolo de predicado n-ádico de $\mathcal{L}$ y $t_1, \ldots, t_n$ son términos de $\mathcal{L}$, entonces $P(t_1, \ldots, t_n)$ es una fórmula (atómica) de $\mathcal{L}$.
2. Si φ es una fórmula de $\mathcal{L}$ entonces $\neg\varphi$ es una fórmula de $\mathcal{L}$.
3. Si φ y ψ son fórmulas de $\mathcal{L}$ entonces $(\varphi \rightarrow \psi)$ es una fórmula de $\mathcal{L}$.ç
4. Si φ es una fórmula de $\mathcal{L}$ y x una variable entonces $(\forall x)\varphi$ es una fórmula de $\mathcal{L}$.
5. Nada más es una fórmula de $\mathcal{L}$.

FORM($\mathcal{L}$) es el conjunto de todas las fórmulas del lenguaje $\mathcal{L}$.
Por ejemplo, para $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$, con $\mathcal{C} = \{c, d\}$, $\mathcal{F} = \{f\}$ y $\mathcal{P} = \{R\}$ (f de aridad 3, R binario) son fórmulas:

$$R(d, x'), \quad (\forall x')R(d, x''), \quad (\forall x'')R(f(x'', x', x'''), d)$$

42.3. Convenciones

Algunas convenciones a utilizar para los lenguajes de primer orden a tratar:

- Usamos $x, y, z, \ldots$ para variables.
- Usamos $a, b, c, d, \ldots$ para símbolos de constante.
- Usamos $f, g, h, \ldots$ para símbolos de función (la aridad simbre va a quedar clara a partir del contexto).
- Usamos $P, Q, R, \ldots$ para símbolos de predicado (la aridad simbre va a quedar clara a partir del contexto).
- Escribimos $(\exists x)\varphi$ en lugar de $\neg(\forall x)\neg\varphi$.
- Escribimos $(\varphi \vee \psi)$ en lugar de $(\neg\varphi \to \psi)$.
- Escribimos $(\varphi \wedge \psi)$ en lugar de $\neg(\varphi \to \neg\psi)$.
- Escribimos φ en lugar de (φ) caundo convenga.

42.4. Variables libres y ligadas

Una aparición deuna variables x en una fórmula está ligada si está dentro del alcance de un cuantificador. En cado contrario, dicha aparición está libre; a su vez una variable está libre/ligada en una fórmula si todas sus apariciones están libres/ligadas. Adicionalmente, se dice que una fórmula es una sentencia si todas las variables son ligadas (no hay apariciones libres de variables).

Por ejemplo, para un lenguaje con un símbolo de predicado binario P se tiene que

- En $P(x, y)$, x está libre.
- En $(\forall y)P(x, y)$, x está libre.
- En $(\forall x)P(x, y)$, x está ligada.
- En $(\forall x)(\forall y)P(x, y)$, x está ligada.
- En $P(x, y) \to (\forall x)(\forall y)P(x, y)$
  - La primera aparición de x está libre.
  - La segunda aparición de x está ligada.
  - Entonces, x no está ni libre ni ligada.

43. Interpretación y valuación

43.1. Estructuras de un lenguaje

Una $\mathcal{L}$-estructura $\mathcal{A}$ (o una interpretación de $\mathcal{L}$), de un lenguaje $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$ está compuesta de un conjunto A no vacío (llamado universo o dominio), y las siguientes asignaciones:

- Para cada símbolos de constante $c \in \mathcal{C}$, un elemento fijo $c_{\mathcal{A}} \in A$.
- Para cada símbolos de función $f \in \mathcal{F}$, una función $f_{\mathcal{A}} : A^n \to A$.
- Para cada símbolos de predicado n-ario $P \in \mathcal{P}$, una relación $P_{\mathcal{A}} \subseteq A^n$.

Las funciones $f_{\mathcal{A}}$ y los predicados $P_{\mathcal{A}}$ son siempre totales.

43.1.1. Ejemplos

Para $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$, con $\mathcal{C} = \{c, d\}$, $\mathcal{F} = f, g$ y $\mathcal{P} = P$ (f unaria, g binaria, P binario)

$\mathcal{L}$-estructura $\mathcal{A}$

- $A = \mathbb{Z}$
- $c_{\mathcal{A}} = 0$
- $d_{\mathcal{A}} = 1$
- $f_{\mathcal{A}}(x) = -x$
- $g_{\mathcal{A}}(x, y) = x + y$
- $P_{\mathcal{A}}(x, y) \iff x|y$

$\mathcal{L}$-estructura $\mathcal{B}$

- $B = \mathcal{P}(\mathbb{N})$
- $c_{\mathcal{B}} = \emptyset$
- $d_{\mathcal{B}} = \mathbb{N}$
- $f_{\mathcal{B}}(x) = \overline{x}$
- $g_{\mathcal{B}}(x, y) = x \cup y$
- $P_{\mathcal{B}}(x, y) \iff x \subseteq y$

43.1.2. No ejemplos

Para $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$, con $\mathcal{C} = \{c, d\}$, $\mathcal{F} = f, g$ y $\mathcal{P} = P$ (f unaria, g binaria, P binario)

$\mathcal{L}$-estructura $\mathcal{M}$

- $M = \mathbb{Z}$
- $c_{\mathcal{M}} = 0$
- $d_{\mathcal{M}} = 1$
- $f_{\mathcal{M}}(x) = 1/x$
- $g_{\mathcal{M}}(x, y) = x^y$
- $P_{\mathcal{M}}(x, y) \iff x|y$

En general,

- $1/x \notin \mathbb{Z}$
- $x^y \notin \mathbb{Z}$.

$\mathcal{L}$-estructura $\mathcal{N}$

- $N = \mathcal{P}(\mathbb{N})$
- $c_{\mathcal{N}} =$ función identidad
- $d_{\mathcal{N}} =$ función cero (nula)
- $f_{\mathcal{N}}(x) =$ derivada de x
- $g_{\mathcal{N}}(x, y) = x \circ y$
- $P_{\mathcal{N}}(x, y) \iff x = y$

Una función $\mathbb{R} \to \mathbb{R}$ puede no ser derivable.

43.2. Valuaciones

Si fijasemos una $\mathcal{L}$-estructura $\mathcal{A}$ con dominio A, una valuación para $\mathcal{A}$ es una función $v : VAR \to A$. Si extendemos v a $\tilde{v} : TERM(\mathcal{L}) \to A$, que interpreta un término t en una $\mathcal{L}$-estructura $\mathcal{A}$ de la forma:

- Si $t = x$ (variable), entonces $\tilde{v}(t) = v(x)$
- Si $t = c$ (constante), entonces $\tilde{v}(t) = c_{\mathcal{A}}$
- Si $t = f(t_1, \ldots, t_n)$ (función), entonces $\tilde{v}(t) = f_{\mathcal{A}}(\tilde{v}(t_1), \ldots, \tilde{v}(t_n))$

Adicionalmente, sea v una valuación de $\mathcal{A}$ y sea $a \in A$. Se define la valuación $v(x = a)$ de la siguiente manera

$$v(x = a)(y) = \begin{cases} v(y) & x \neq y \\ a & x = y \end{cases}$$

(escribimos v en vez de $\tilde{v}$).

43.2.1. Ejemplos

Para $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$, con $\mathcal{C} = \{c, d\}$, $\mathcal{F} = f, g$ y $\mathcal{P} = P$ (f unaria, g binaria, P binario)

$\mathcal{L}$-estructura $\mathcal{A}$

- $A = \mathbb{Z}$
- $c_{\mathcal{A}} = 0$
- $d_{\mathcal{A}} = 1$
- $f_{\mathcal{A}}(x) = -x$
- $g_{\mathcal{A}}(x, y) = x + y$
- $P_{\mathcal{A}}(x, y) \iff x|y$

Tenemos

- Si $v(x) = 2$
  - $\tilde{v}(g(x, f(d))) = 2 + (-1) = 1$
- Para cualquier v
  - $\tilde{v}(g(c, f(d))) = 0 + (-1) = -1$

$\mathcal{L}$-estructura $\mathcal{B}$

- $B = \mathcal{P}(\mathbb{N})$
- $c_{\mathcal{B}} = \emptyset$
- $d_{\mathcal{B}} = \mathbb{N}$
- $f_{\mathcal{B}}(x) = \overline{x}$
- $g_{\mathcal{B}}(x, y) = x \cup y$
- $P_{\mathcal{B}}(x, y) \iff x \subseteq y$

Tenemos

- Si $v(x) = \{1, 2\}$
  - $\tilde{v}(g(x, f(d))) = \{1, 2\} \cup \overline{\mathbb{N}} = \{1, 2\}$
- Para cualquier v
  - $\tilde{v}(g(c, f(d))) = \emptyset \cup \overline{\mathbb{N}} = \emptyset$

43.3. Interpretación de una fórmula

Sea $\mathcal{A}$ una $\mathcal{L}$-estructrura con dominio A y v una valuación de $\mathcal{A}$. Se propone la siguiente definición, cuando φ es verdadera en $\mathcal{A}$ bajo la valuación v (lo cual notamos $\mathcal{A} \vDash \varphi[v]$) y

1. φ es de la forma $P(t_1, \ldots, t_n)$ (atómica)
$$\mathcal{A} \vDash P(t_1, \ldots, t_n)[v] \iff (\tilde{v}(t_1), \ldots, \tilde{v}(t_n)) \in P_{\mathcal{A}}$$
2. φ es de la forma $\neg\psi$
$$\mathcal{A} \vDash \neg\psi[v] \iff \mathcal{A} \nvDash \psi[v]$$
3. φ es de la forma $(\psi \to \rho)$
$$\mathcal{A} \vDash (\psi \to \rho)[v] \iff \mathcal{A} \nvDash \psi[v] \vee \mathcal{A} \vDash \rho[v]$$
4. φ es de la forma $(\forall x)\psi$
$$\mathcal{A} \vDash (\forall x)\psi[v] \iff \forall a \in A,\ \mathcal{A} \vDash \psi[v(x = a)]$$

43.3.1. Ejemplos

Para $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$, con $\mathcal{C} = \{c, d\}$, $\mathcal{F} = f, g$ y $\mathcal{P} = P$ (f unaria, g binaria, P binario)

$\mathcal{L}$-estructura $\mathcal{A}$

- $A = \mathbb{Z}$
- $c_{\mathcal{A}} = 0$
- $d_{\mathcal{A}} = 1$
- $f_{\mathcal{A}}(x) = -x$
- $g_{\mathcal{A}}(x, y) = x + y$
- $P_{\mathcal{A}}(x, y) \iff x|y$

Tenemos

- Para $v(x) = 1$
  - $\mathcal{A} \vDash P(x, d)[v]$
- Para $v(x) = 0$
  - $\mathcal{A} \nvDash P(x, c)[v]$
- Para cualquier v
  - $\mathcal{A} \nvDash (\forall y)P(y, g(y, d))[v]$

$\mathcal{L}$-estructura $\mathcal{B}$

- $B = \mathcal{P}(\mathbb{N})$
- $c_{\mathcal{B}} = \emptyset$
- $d_{\mathcal{B}} = \mathbb{N}$
- $f_{\mathcal{B}}(x) = \overline{x}$
- $g_{\mathcal{B}}(x, y) = x \cup y$
- $P_{\mathcal{B}}(x, y) \iff x \subseteq y$

Tenemos

- Para $v(x) = \emptyset$
  - $\mathcal{B} \vDash P(x, d)[v]$
- Para $v(x) = \{1, 2, 3\}$
  - $\mathcal{B} \nvDash P(x, c)[v]$
- Para cualquier v
  - $\mathcal{B} \vDash (\forall y)P(y, g(y, d))[v]$

43.3.2. Notación ($\wedge$, $\vee$, $\exists$)

Sea $\mathcal{A}$ una $\mathcal{L}$-estructura y v una valuación de $\mathcal{A}$. Se deduce:

5. φ es de la forma $\psi \vee \rho$
$$\mathcal{A} \vDash (\psi \vee \rho)[v] \iff \mathcal{A} \vDash \psi[v] \vee \mathcal{A} \vDash \rho[v]$$
6. φ es de la forma $\psi \wedge \rho$
$$\mathcal{A} \vDash (\psi \wedge \rho)[v] \iff \mathcal{A} \vDash \psi[v] \wedge \mathcal{A} \vDash \rho[v]$$
7. φ es de la forma $(\exists x)\psi$
$$\mathcal{A} \vDash (\exists x)\psi[v] \iff \exists a \in A,\ \mathcal{A} \vDash \psi[v(x = a)]$$

44. Niveles de verdad

Para un lenguaje $\mathcal{L}$ fijo se dice que una $\mathcal{L}$-fórmula φ:

1. Es satisfacible si existe una $\mathcal{L}$-estructura $\mathcal{A}$ y una valuación v de $\mathcal{A}$ tal que $\mathcal{A} \vDash \varphi[v]$.
2. Es verdadera (o válida) en una $\mathcal{L}$-estructura $\mathcal{L}$ (es decir, $\mathcal{A} \vDash \varphi$) si $\mathcal{A} \vDash \varphi[v]$ para toda valuación v de $\mathcal{A}$.
   - En este caso se dice que $\mathcal{A}$ es un modelo de φ.
3. Es universalmente válida ($\vDash \varphi$) si $\mathcal{A} \vDash \varphi[v]$ (a veces escrito $\mathcal{A}, v \vDash \varphi$) para toda $\mathcal{L}$-estructura $\mathcal{A}$ y toda valuación v de $\mathcal{A}$.

44.1. Ejemplos

- $\mathcal{A} = \langle \mathbb{Z}; <, 0 \rangle$ con la interpretación usual
  - $\mathcal{A} \vDash (\forall x)(\exists y)x < y$
  - $\mathcal{A} \vDash (\exists y)x < y$
  - $\mathcal{A} \nvDash x < y \to (\exists z)(x < z \wedge z < y)$
  - $\mathcal{A} \vDash (\exists x)x < 0$
- $\mathcal{B} = \langle \mathbb{N}; <, 0 \rangle$ con la interpretación usual
  - $\mathcal{B} \nvDash x < y \to (\exists z)(x < z \wedge z < y)$
  - $\mathcal{B} \nvDash (\exists x)x < 0$
- $\mathcal{C} = \langle \mathbb{Q}; <, 0 \rangle$ con la interpretación usual
  - $\mathcal{C} \vDash x < y \to (\exists z)(x < z \wedge z < y)$
  - $\mathcal{C} \vDash (\exists x)x < 0$
- $(\exists x)(\forall y)P(x, y)$ es satisfacible
  - $\mathcal{D} = \langle \{0\}; = \rangle$ con la interpretación usual
  - $\mathcal{E} = \langle \mathbb{N}; \leq \rangle$ con la interpretación usual

- $\vDash (\forall x)P(x) \to P(x)$ se entiende $((\forall x)P(x)) \to P(x)$
- $\nvDash P(x) \to (\forall x)P(x)$
  - $\mathcal{F} = \langle \mathbb{N}; par \rangle$ con la interpretación usual, $v(x) = 0$

44.2. Resultados sobre satisfacibilidad y validez

- Si φ es una sentencia, $\mathcal{A} \vDash \varphi \iff \mathcal{A} \vDash \varphi[v]$.
- φ es universalmente válida $\iff \neg\varphi$ es insatisfacible.
- Preservación de validez del Modus Ponens:
  - $\mathcal{A} \vDash \varphi[v] \land \mathcal{A} \vDash (\varphi \to \psi)[v] \implies \mathcal{A} \vDash \psi[v]$
  - $\mathcal{A} \vDash \varphi \land \mathcal{A} \vDash (\varphi \to \psi) \implies \mathcal{A} \vDash \psi$
  - $\vDash \varphi \land \vDash (\varphi \to \psi) \implies \vDash \psi$
- Clausura universal
  - $\mathcal{A} \vDash \varphi \iff \mathcal{A} \vDash (\forall x)\varphi$
  - $\vDash \varphi \iff \vDash (\forall x)\varphi$

Decimos que un elemento e del universo de una $\mathcal{L}$-estructura $\mathcal{A}$ es distinguible si existe una $\mathcal{L}$-fórmula φ con una sola variable libre x tal que $\mathcal{A} \vDash \varphi[v] \iff v(x) = e$.

45. Consecuencia semántica en lógica de primer orden

Sea $\Gamma \subseteq FORM(\mathcal{L})$ y $\varphi \in FORM(\mathcal{L})$. Se dice que φ es consecuencia semántica de Γ ($\Gamma \vDash \varphi$) si para toda $\mathcal{L}$-estructura $\mathcal{A}$ y toda valuación v de $\mathcal{A}$

$$\mathcal{A} \vDash \Gamma[v] \implies \mathcal{A} \vDash \varphi[v]$$

Y Notamos $\mathcal{A} \vDash \Gamma[v]$ como, para toda $\psi \in \Gamma$, $\mathcal{A} \vDash \psi[v]$.

Dada una $\mathcal{L}$-estructura $\mathcal{A}$ con universo A, decimos que una relación $R \subseteq A^n$ es expresable si existe una $\mathcal{L}$-fórmula φ con n variables $x_1, \ldots, x_n$ tal que para toda valuación v, vale que $\mathcal{A} \vDash \varphi[v] \iff (v(x_1), \ldots, v(x_n)) \in R$.

A su vez, decimos que una clase de $\mathcal{L}$-estructuras $\mathcal{K}$ es definible si existe una sentencia φ tal que para toda $\mathcal{L}$-interpretación $\mathcal{A}$, vale que $\mathcal{A} \vDash \varphi \iff \mathcal{A} \in \mathcal{K}$.

45.1. Ejemplos

Sea $L = \{P, Q\}$, con P y Q símbolos de predicado unarios.

- $\Gamma_1 = \{(\forall x)(P(x) \to Q(x))\}$
  - $\Gamma_1 \nvDash (\exists x)P(x)$
  - $\Gamma_1 \vDash (\exists x)P(x) \to^{(} \exists x)Q(x)$
  - $\Gamma_1 \vDash (\forall x)P(x) \to (\forall x)Q(x)$
- $\Gamma_2 = \{(\forall x)(P(x) \to Q(x)),\ (\exists x)P(x)\}$
  - $\Gamma_2 \vDash (\exists x)Q(x)$
  - $\Gamma_2 \vDash (\exists x)(P(x) \land Q(x))$
  - $\Gamma_2 \nvDash (\exists x)(\neg P(x) \land Q(x)$
- $\Gamma_3 = \{(\forall x)(P(x) \to Q(x)),\ (\exists x)(P(x) \land \neg Q(x))\}$
  - $\Gamma_3 \vDash \varphi$ para cualquier φ

45.2. Lenguajes con igualdad

$\mathcal{L}$ es un lenguaje con igualdad si tiene un símbolo proposicional binario especial (el $=$) que sólo se interpreta como la igualdad. Para ver algunos ejemplos, fijemos un lenguaje $\mathcal{L}$ con igualdad y con ningún otro símbolo. Buscamos $\varphi \in FORM(\mathcal{L})$ tal que $\{\mathcal{A} : \mathcal{A} \vDash \varphi\}$ sea la clase de modelos con

- exactamente 1 elemento
$$\varphi = (\exists x)(\forall y)x = y$$
- exactamente 2 elementos
$$\varphi = (\exists x)(\exists y)(x \neq y \wedge (\forall z)(z = x \vee z = y))$$
- exactamente 3 elementos
$$\varphi = (\exists x)(\exists y)(\exists z)(x \neq y \wedge x \neq z \wedge y \neq z)$$

46. Sustitución de variables

46.1. Reemplazo de variables libres por términos

Sea $\mathcal{L} = \mathcal{C} \cup \mathcal{F} \cup \mathcal{P}$ un lenguaje fijo. Sea $\varphi \in FORM(\mathcal{L})$, $t \in TERM(\mathcal{L})$ y $x \in VAR$. $\varphi[x/t]$ es la fórmula obtenida a partir de φ sustituyendo todas las apariciones libres de la variables x por t. Por ejemplo, para un lenguaje con símbolo de predicado binario P y símbolo de función unario f se tiene

- $P(x, y)[x/f(z)] = P(f(z), y)$
- $P(x, y)[x/f(x)] = P(f(x), y)$
- $((\forall x)(\forall y)P(x, y))[x/f(z)] = (\forall x)(\forall y)P(x, y)$
- $(P(x, y) \to (\forall x)(\forall y)P(x, y))[x/f(z)] = P(f(z), y) \to (\forall x)(\forall y)P(x, y)$

Si $c \in \mathcal{C}$, $\varphi[c/t]$ es la fórmula obtenida a partir de φ sustituyendo todas las apariciones de la constante c por t.

46.1.1. Variable reemplazable por un término

Sea $t \in TERM(\mathcal{L})$, $x \in VAR$ y $\varphi \in FORM(\mathcal{L})$. Decimos que x es remplazable por t en φ cuando

1. t es un término cerrado (i.e. no tiene variables) o
2. t tiene variables pero ninguna de ellas queda atrapada por un cuantificador en el reemplazo $\varphi[x/t]$.

Por ejemplo, para un lenguaje con símbolo de predicado unario P y símbolo de función binaria f, en

$$(\forall y)((\forall x)P(x) \to P(x))$$

- x es reemplazable por z: $(\forall y)((\forall x)P(x) \to P(z))$
- x es reemplazable por $f(x, z)$: $(\forall y)((\forall x)P(x) \to P(f(x, z)))$
- x no es reemplazable por $f(x, y)$: $(\forall y)((\forall x)P(x) \to P(f(x, y)))$

46.2. Lema de sustitución

Lema

Si x es reemplazable por t en φ entonces

$$\mathcal{A} \vDash (\varphi[x/t])[v] \iff \mathcal{A} \vDash \varphi[v(x = \tilde{v}(t))]$$

Demostración. Esto se puede probar por inducción en la complejidad de φ. Veamos caso por caso

- $\varphi = P(u)$ es atómica (u es término; el caso n-ario es similar). Utilizando que $v(u[x/t]) = v(x = v(t))(u)$, se tiene

$$\mathcal{A} \vDash P(u[x/t])[v] \iff \tilde{v}(u[x/t]) \in P_{\mathcal{A}} \iff v[x = \tilde{v}(t)](u) \in P_{\mathcal{A}} \iff \mathcal{A} \vDash \varphi[v(x = \tilde{v}(t))]$$

- φ es de la forma $\neg\psi$ o de la forma $\psi \to \rho$ resulta trivial.
- φ es de la forma $(\forall y)\psi$. Hay dos posibles situaciones:
  - Supongo que x no aparece libre en φ. Tengo que v y $v[x = v(t)]$ coinciden en todas las variables que aparecen libres en φ, además con esto resulta inmediato ver que $\varphi = \varphi[x/t]$.
  - Supongo que x sí aparece libre en φ. Como x es reemplazable por t en φ, y no ocurre en t. Luego, para todo $d \in A$

$$v(t) = v(y = d)(t) \tag{1}$$

Además, se tiene que x es reemplazable por t en ψ, y como $x \neq y$,

$$\varphi[x/t] = ((\forall y)\psi)[x/t] = (\forall y)(\psi[x/t]).$$

Luego con esto se tiene

$$\begin{aligned}
\mathcal{A} \vDash \varphi[x/t][v] &\underset{def}{\iff} \forall d \in A,\ \mathcal{A} \vDash \psi[x/t][v(y = d)] \\
&\underset{h.i.}{\iff} \forall d \in A,\ \mathcal{A} \vDash \psi[v(y = d)(x = v(y = d)(t))] \\
&\underset{(1)}{\iff} \forall d \in A,\ \mathcal{A} \vDash \psi[v(y = d)(x = v(t))] \\
&\underset{x \neq y}{\iff} \forall d \in A,\ \mathcal{A} \vDash \psi[v(x = v(t))(y = d)] \\
&\underset{def}{\iff} \mathcal{A} \vDash \varphi[v(x = v(t))]
\end{aligned}$$

Con esto queda probado el lema. □

47. Sistema axiomático *SQ*

EL mecanismo deductivo SQ para un lenguaje fijo $\mathcal{L}$ se compone de

- Axiomas. Sean $\varphi,\ \psi,\ \rho \in FORM(\mathcal{L}),\ x \in VAR,\ t \in TERM(\mathcal{L})$

  SQ1 $\varphi \to (\psi \to \rho)$
  SQ2 $(\varphi \to (\psi \to \rho)) \to ((\varphi \to \psi) \to (\varphi \to \rho)$
  SQ3 $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$
  SQ4 $(\forall x)\varphi \to \varphi[x/t]$ si x es reemplazable por t en φ
  SQ5 $\varphi \to (\forall x)\varphi$ si x no aparece libre en φ
  SQ6 $(\forall x)(\varphi \to \psi) \to ((\forall x)\varphi \to (\forall x)\psi)$
  SQ7 Si φ es un axioma entonces $(\forall x)\varphi$ también es un axioma

- Regla de inferencia.

  MP Sean $\varphi,\ \psi \in FORM(\mathcal{L})$. ψ es una consecuencia inmediata de $\varphi \to \psi$ y φ

48. Consecuencia sintáctica en lógica de primer orden

48.1. Consecuencia sintáctica, demostraciones, teoremas y teorías

Fijamos un lenguaje $\mathcal{L}$. Sea $\Gamma \subseteq FORM(\mathcal{L})$ y $\varphi \in FORM(\mathcal{L})$.

1. Una demostración de φ en SQ es una cadena finita y no vacía $\varphi_1, \ldots, \varphi_n$ de fórmulas de $\mathcal{L}$ tal que $\varphi_n = \varphi$ y
   - φ_i es un axioma o

- φ_i es una consecuencia inmediata de φ_k, φ_l (con $k, l < i$)

Donde en este caso, decimos que φ es un teorema ($\vdash \varphi$)

2. φ es una consecuencia sintñáctica de Γ ($\Gamma \vdash \varphi$) si existe una cadena finita y no vacía $\varphi_1, \ldots, \varphi_n$ de fórmulas de $\mathcal{L}$ tal que $\varphi_n = \varphi$ y

   - φ_i es un axioma o
   - $\varphi_i \in \Gamma$ o
   - φ_i es una consecuencia inmediata de φ_k, φ_l (con $k, l < i$)

   En este caso, $\varphi_1, \ldots, \varphi_n$ se llama derivación de φ a partir de Γ. A su vez Γ se llama teoría y decimos que φ es un teorema de la teoría Γ.

48.1.1. Ejemplo

Tomemos como ejemplo $\Gamma = \{(\forall x)(\varphi[z/x])\} \vdash (\forall z)\varphi$, donde x no aparece en φ.

1. $(\forall z)((\forall x)(\varphi[z/x]) \to \varphi)$	$SQ4 + SQ7$
2. $(\forall z)((\forall x)(\varphi[z/x]) \to \varphi) \to ((\forall z)(\forall x)(\varphi[z/x]) \to (\forall z)\varphi)$	$SQ6$
3. $(\forall z)(\forall x)(\varphi[z/x]) \to (\forall z)\varphi$	MP 1, 2
4. $(\forall x)(\varphi[z/x]) \to (\forall z)(\forall x)(\varphi[z/x])$	$SQ5$
5. $(\forall x)(\varphi[z/x])$	Γ
6. $(\forall z)(\forall x)(\varphi[z/x])$	MP 4, 5
7. $(\forall z)\varphi$	MP 3, 6

Observar que

- Paso 1: x es reemplazable por z en $\varphi[z/x]$
- Paso 1: $\varphi[z/x][x/z] = \varphi$
- Paso 4: z no aparece libre en $(\forall x)(\varphi[z/x])$

48.2. Correctitud y consistencia

Teorema: (Correctitud)

El sistema SQ es correcto, i.e. $\Gamma \vdash \varphi \implies \Gamma \vDash \varphi$.

Teorema: (Consistencia)

El sistema SQ es consistente, i.e. no existe $\varphi \in FORM(\mathcal{L})$ tal que $\vdash \varphi$ y $\vdash \neg\varphi$.

48.3. Resultados similares a los de SP

Teorema: (de la Deducción)

Si $\Gamma \cup \{\varphi\} \vdash \psi$ entonces $\Gamma \vdash \varphi \to \psi$

Decimos que $\Gamma \subseteq FORM(\mathcal{L})$ es consistente si no existe $\varphi \in FORM(\mathcal{L})$ tal que $\Gamma \vdash \varphi$ y $\Gamma \vdash \neg\varphi$.

Proposición

1. $\Gamma \cup \{\neg\varphi\}$ es inconsisitente $\iff \Gamma \vdash \varphi$
2. $\Gamma \cup \{\varphi\}$ es inconsisitente $\iff \Gamma \vdash \neg\varphi$

Teorema

Si Γ es satisfacible, entonces Γ es consistente.

Teorema

Si Γ es inconsistente, entonces existe un subconjunto finito de Γ que es inconsistente.

48.4. Instancias de esquemas tautológicos

Sea $\varphi(p_1, \ldots, p_n)$ una tautología de P con variables proposicionales $p_1, \ldots, p_n$, y sean $\psi_1, \ldots, \psi_n$ fórmulas cualesquiera de primer orden. Deicmos que $\varphi(\psi_1, \ldots, \psi_n)$ es una instancia de un esquema tautológico (reemplazando p_i por ψ_i en la fórmula original φ).

Proposición: S

i φ es una instancia de un esquema tautológico entonces $\vdash \varphi$.

Por ejemplo, la fórmula $P(p, q) = (p \wedge q) \to p$ es una tautología. Luego, utilizando $\psi = (\forall x)R(x)$ y $\rho = (\exists y)Q(y)$ se tiene que $P(\psi, \rho)$ es una instancia de un esquema tautológico y entonces

$$\vdash ((\forall x)R(x) \wedge (\exists y)Q(y)) \to (\forall x)R(x)$$

48.5. Variantes alfabéticas

Sea $\mathcal{L} = \{0, S\}$ con igualdad y $\varphi \in FORM(\mathcal{L})$ definida como $\varphi = x \neq 0 \to (\exists y)x = S(y)$.

En φ la variable x es reemplazable por z:

$$\varphi[x/z] = z \neq 0 \to (\exists y)z = S(y)$$

Sin embargo, la variable x no es reemplazable por y:

$$\varphi[x/y] = y \neq 0 \to (\exists y)y = S(y)$$

No habría habido problema si la fórmula original hubiese sido:

$$\varphi' = x \neq 0 \to (\exists w)x = S(w)$$

Esto es para ilustrar que a φ' se la llama variante alfabética de φ.

Lema

Sea $\varphi \in FORM(\mathcal{L})$. Dados $x \in VAR$ y $t \in TERM(\mathcal{L})$ podemos encontrar φ' (variante alfabética de φ) tal que

- $\{\varphi\} \vdash \varphi'$ y $\{\varphi'\} \vdash \varphi$
- x es reemplazable por t en φ'

49. Teorema de la generalización (TG)

Teorema

Si $\Gamma \vdash \varphi$ y x no aaprece libre en ninguna fórmula de Γ, entonces $\Gamma \vdash (\forall x)\varphi$.

Observar el por qué es necesario pedir que x no aparezca libre en ninguna fórmula de Γ:

- $\Gamma = \{R(x)\} \vDash (\forall x)R(x)$
- entonces por correctitud, $\{R(x)\} \nvdash (\forall x)R(x)$

Demostración. Para demostrar esto, se plantea lo siguiente

$P(n) =$ para toda ψ, Γ, x tal que $\Gamma \vdash \psi$ y x no aparece libre en ninguna fórmula de Γ, si $\psi_1, \dots, \psi_n$ es una derivación de ψ a partir de Γ entonces $\Gamma \vdash (\forall x)\psi$.

Luego, esto se puede demostrar por inducción en n.

- *Caso base*: $P(1)$
  - Sea φ, Γ y x tal que x no aparece libre en Γ
  - Sea φ una derivación de φ a partir de Γ
  - Queremos ver que $\Gamma \vdash (\forall x)\varphi$

  Se tienen 2 posibilidades:

  1. Si φ es un axioma de SQ, por $SQ7$, $\vdash \varphi \implies \Gamma \vdash (\forall x)\varphi$
  2. Si $\varphi \in \Gamma$ entonces
     - $\Gamma \vdash \varphi$
     - Por hipótesis, x no aparece libre en φ
     - Y por $SQ5$, $\varphi \to (\forall x)\varphi$
     - Luego por MP, $\Gamma \vdash (\forall x)\varphi$
- *Paso inductivo*: $P(n)$
  - Sean φ, Γ, x tal que x no aparece libre en Γ
  - Sean $\varphi_1, \dots, \varphi_n$ una derivación de *varphi* a apartir de Γ
  - Queremos ver que $\Gamma \vdash (\forall x)\varphi$
  - Tomamos por hipótesis inductiva vale $P(m)$ para todo $m < n$

  Nos encontramos con 3 posibilidades:

  1. φ es axioma de SQ, igual que el caso base.
  2. $\varphi \in \Gamma$, igual que el caso base.
  3. φ se obtiene por MP de φ_i y φ_j (con $i, j < n$). Supongamos que $\varphi_j = \varphi_i \to \varphi$. Usamos HI 2 veces:
     - Como $i < n$, vale $P(i)$, y en particular $\Gamma \vdash (\forall x)\varphi_i$
     - Como $j < n$, vale $P(j)$, y en particular $\Gamma \vdash (\forall x)(\varphi_i \to \varphi)$

     Luego por $SQ6$, se tiene $\vdash (\forall x)(\varphi_i \to \varphi) \to ((\forall x)\varphi_i \to (\forall x)\varphi)$. Y aplicando MP 2 veces (primero con ■ y luego con ■) se obtiene que $\Gamma \vdash (\forall x)\varphi$.

  Con esto queda probado lo pedido.

□

50. Teorema de la generalización en constantes (TGC)

Teorema

Supongamos que $\Gamma \vdash \varphi$ y c es un símbolo de constante que no aparece en Γ. Encontes existe una variable x que no aparece en φ tal que $\Gamma \vdash (\forall x)(\varphi[c/x])$. Más aun, hay una derivación de $(\forall x)(\varphi[c/x])$ a partir de Γ en donde c no aparece.

50.1. Consecuencias de TGC

Corolario

Supongamos que $\Gamma \vdash \varphi[z/c]$ y c es un símbolo de constante que no paraece en Γ ni en φ. Entonces $\varphi \vdash (\forall z)\varphi$. Más aun, hay una derivación de $(\forall z)\varphi$ a partir de Γ en donde c no aparece.

Demostración. Por el TGC, existe x, tal que x no aparece en $\varphi[z/c]$ y $\Gamma \vdash (\forall x)(\varphi[z/c][c/z])$ (y en esta última derivación no aparece c).
Como c no aparece en φ, $\varphi[z/c][c/x] = \varphi[z/x]$. De forma que $\Gamma \vdash (\forall x)(\varphi[z/x])$, y por el teorema de la deducción, sabemos $\vdash (\forall x)(\varphi[z/x]) \to (\forall z)\varphi$.
Por último haciendo MP concluimos que $\Gamma(\forall z)\varphi$ (notar que en está última derivación no aparece c). □

50.2. Lenguajes con igualdad en lógica de primer orden

Fijamos un lenguaje $\mathcal{L}$ con igualdad. Para los lenguajes con igualdad, se considera el sistema $SQ^=$ con los axiomas y la regla de inferencia de SQ, sumando estos dos axiomas.

- Axiomas adicionales para $SQ^=$. Sea $\varphi, \psi \in FORM(L)$, $x, y \in VAR$

  SQ$^=$1 $x = x$
  SQ$^=$2 $x = y \to (\varphi \to \psi)$ donde φ es atómica y ψ se obtiene de φ reemplazando x por y en cero o más lugares.

Se puede probar que

- $SQ^=$ es consistente
- Si hay una derivación de φ en $SQ^=$ entonces φ es verdadera en toda $\mathcal{L}$-estructura en donde el $=$ se interpreta como la igualdad

50.3. Notas sobre computabilidad y lógica de primer orden

Fijemos un lenguaje numerable $\mathcal{L}$. Se pueden codificar las fórmulas de $FORM(\mathcal{L})$ con números naturales. Igual que para la lógica proposicional:

- Es computable la función:

$$dem_{\mathcal{L}}(x) \begin{cases} 1 & x \text{ es una demostración válida en } SQ \text{ para } \mathcal{L} \\ 0 & cc \end{cases}$$

- Considerar el siguiente programa P:

$$\begin{array}{ll} [A] & \text{IF } dem_{\mathcal{L}}(D) = 1 \wedge D[|D|] = X \text{ GOTO } E \\ & D \leftarrow D + 1 \\ & \text{GOTO } A \end{array}$$

- φ es teorema de SQ para $\mathcal{L} \iff \#\varphi \in dom(\Psi_P)$
- El conjunto de teoremas del sistema SQ para $\mathcal{L}$ es c.e.

51. Completitud de *SQ*

Para probar dicho teorema, primero es necesaria ver que *Consistente* $\implies$ *Satisfacible*. Esto va a ser probado por una larga demostración, para luego abordar el teorema de completitud y sus corolarios.

51.1. Consistente $\implies$ Satisfacible

Sea $\mathcal{L}$ un lenguaje fijo. Sea $\Gamma \subseteq FORM(\mathcal{L})$ consistente. Se quiere construir un modelo canónico $\mathcal{B}$ y una valuación v de $\mathcal{B}$ tal que:

$$\mathcal{B} \vDash \varphi[v] \text{ para toda } \varphi \in \Gamma$$

Demostración en 5 pasos:

Demostración.

- Paso 1. Expandir $\mathcal{L}$ a $\mathcal{L}'$ con nuevas constantes. $\mathcal{L}' = \mathcal{L} \cup \mathcal{C}$. En $\mathcal{C}$ hay una cantidad infinita numerable de nuevas constantes ("nuevas"porque no aparecen en $\mathcal{L}$).
- Paso 2. Agregar testigos a Γ. Trabajamos con $\Gamma \cup \Theta$, donde Θ es un conjunto de fórmulas especiales que usan las constantes nuevas de $\mathcal{L}'$.
- Paso 3. Aplicar el lema de Lindenbaum para $\Gamma \cup \Theta$. Obtener $\Delta \supseteq \Gamma \cup \Theta$ maximal consistente.
- Paso 4. Construir el modelo canónico $\mathcal{A}$ y valuación v (para $\mathcal{L}'$) tal que $\mathcal{A} \vDash \varphi[v] \iff \varphi \in \Delta$.
- Paso 5. Restringir $\mathcal{A}$ y v al lenguaje original $\mathcal{L}$ y obtener $\mathcal{B}$.

□

51.1.1. Paso 1: Expandir $\mathcal{L}$ a $\mathcal{L}'$ con nuevas constantes

Teorema

Sea $\Gamma \subseteq FORM(\mathcal{L})$ consistente. Sea $\mathcal{C}$ un conjutno de nuevas constantes que no aparecen en $\mathcal{L}$. Si $\mathcal{L}' = \mathcal{L} \cup \mathcal{C}$ entonces Γ es consistente en el lenguaje $\mathcal{L}'$.

Demostración. Para probar esto, se puede probar por el absurdo. Supongo que Γ es inconsistente en el nuevo lenguaje $\mathcal{L}'$. Entonces $\exists \varphi \in FORM(\mathcal{L}')$ tal que $\Gamma \vdash \varphi \wedge \Gamma \vdash \neg\varphi$. Con esto se tiene:

- Cada una de estas derivaciones usa fórmulas en $FORM(\mathcal{L}')$ donde solo aparece finitas constantes nuevas.
- Por el TGC, cada cosntante nueva utilizada (que por hipótesis no aparece en Γ) puede reemplazarse por una variable nueva.
- De esta forma se obtiene una derivación de

  $$\Gamma \vdash \varphi[c_1, \ldots, c_n/x_1, \ldots, x_n] \quad \text{y} \quad \Gamma \vdash \neg\varphi[c_1, \ldots, c_n/x_1, \ldots, x_n]$$

  en el lenguaje original $\mathcal{L}$ (donde c_i son nuevas constantes y x_i son nuevas variables).

Luego, Γ es inconsistente en el lenguaje $\mathcal{L}$ llegando a un absurdo.

□

51.1.2. Paso 2: Agregar testigos a Γ

Sean Γ y $\mathcal{C}$ como en el paso 1. Sea $\langle \varphi_1, x_1 \rangle, \langle \varphi_2, x_2 \rangle, \ldots$ una enumeración de $FORM(\mathcal{L}') \times VAR$. Definimos

$$\theta_n = \neg(\forall x_n)\varphi_n \to \neg(\varphi_n[x_n/c_n])$$

donde c_n es la primera constante de $\mathcal{C}$ que

- No aparece en φ_n y
- no aparece en $\theta_1, \ldots, \theta_{n-1}$

Con esto definimos $\Theta = \{\theta_1, \theta_2, \ldots\}$

Teorema

$\Gamma \cup \Theta \subseteq FORM(\mathcal{L}')$ es consistente.

Observar que Θ agrega testigos a Γ. Si ocurre $\neg(\forall x)\varphi$ entonces hay una constante c que atestigua que φ no vale para todo x, i.e. $\neg(\varphi[x/c])$.

Demostración. Supongamos Γ consistente. Recordemos $\Theta = \{\theta_1, \theta_n, \ldots\}$

- Supongamos que $\Gamma \cup \Theta$ inconsistente, debe existir i tal que $\Gamma \cup \{\theta_1, \ldots, \theta_{i+1}\}$ es inconsistente; sea n el mínimo tal i.
- Es importante observar que $\Gamma \cup \{\theta_1, \ldots, \theta_n\}$ es consistente y se tiene que

$$\Gamma \cup \{\theta_1, \ldots, \theta_n\} \vdash \neg \underbrace{(\neg(\forall x)\varphi \to \neg(\varphi[x/x]))}_{\theta_{n+1}}$$

  donde c no aparece en $\{\theta_1, \ldots, \theta_n\}$ ni en φ.
- De forma que las siguientes fórmulas son instancias de esquemas tautológicos:
  - $\neg\theta_{n+1} \to \neg(\forall x)\varphi$
  - $\neg\theta_{n+1} \to (\varphi[x/c])$

  por lo tanto
  - $\vdash \neg\theta_{n+1} \to \neg(\forall x)\varphi \underset{MP}{\Longrightarrow} \Gamma \cup \{\theta_1, \ldots, \theta_n\} \vdash \neg(\forall x)\varphi$
  - $\vdash \neg\theta_{n+1} \to (\varphi[x/c]) \underset{MP}{\Longrightarrow} \Gamma \cup \{\theta_1, \ldots, \theta_n\} \vdash \varphi[x/c]$
- Pero luego por el corolario del TGC se tiene que $\Gamma \cup \{\theta_1, \ldots, \theta_n\} \vdash (\forall x)\varphi$ (notar que c no aparece en $\Gamma \cup \{\theta_1, \ldots, \theta_n\}$ ni en φ).
- Resultando en $\Gamma \cup \{\theta_1, \ldots, \theta_n\}$ inconsistente, llegando a un absurdo.

□

51.1.3. Paso 3: Lema de Lindenbaum para $\Gamma \cup \Theta$

Teorema

Sean Γ y Θ como en los pasos 1 y 2. Existe un conjunto $\Delta \supseteq \Gamma \cup \Theta$ tal que Δ es maximal consistente.

Demostración. La construcción de dicho Δ se da de manera similar al caso de la lógia proposicional del lema de Lindenbaum. □

Como en el caso proposicional, para toda $\varphi \in FORM(\mathcal{L}')$

- $\varphi \in \Delta$ o bien $\neg\varphi \in \Delta$
- $\varphi \in \Delta \iff \Delta \vdash \varphi$

51.1.4. Paso 4: Construcción del modelo canónico $\mathcal{A}$

Primero vamos a definir el modelon canónico $\mathcal{A}$:

- $A = TERM(\mathcal{L}')$
- Para cada símbolo de función n-aria $f \in \mathcal{L}'$, $f_{\mathcal{A}}\underbrace{(t_1, \ldots, t_n)}_{\in A^n} = f(t_1, \ldots, t_n) \in A$
- Para cada símbolo de constante $c \in \mathcal{L}'$, $c_{\mathcal{A}} = c \in A$
- Para cada símbolo de predicado n-ario $P \in \mathcal{L}'$, $\underbrace{(t_1, \ldots, t_n)}_{\in A^n} \in P^{\mathcal{A}} \iff P(t_1, \ldots, t_n) \in \Delta$

Luego vamos a definir la valuación $v : VAR \to \underbrace{TERM(\mathcal{L}')}_{A}$ como $v(x) = x$

Lema

Para todo $t \in TERM(\mathcal{L}')$, $\tilde{v}(t) = t$

Demostración. Se puede probar por inducción en la complejidad de t. □

Teorema

Para toda $\varphi \in FORM(\mathcal{L}')$, $\mathcal{A} \vDash \varphi[v] \iff \varphi \in \Delta$

Demostración. Esto se puede probar por inducción en la complejidad de φ

- *Caso base*: Si φ es una fórmula atómica $P(t_1, \ldots, t_n)$:

$$\begin{aligned} \mathcal{A} \vDash P(t_1, \ldots, t_n)[v] &\iff v(\tilde{v}(t_1), \ldots, \tilde{v}(t_n)) \in P^{\mathcal{A}} \\ &\iff (t_1, \ldots, t_n) \in P^{\mathcal{A}} && \text{pues } \tilde{v}(t) = t \\ &\iff P(t_1, \ldots, t_n) \in \Delta && \text{por definición de } \mathcal{A} \end{aligned}$$

- *Paso inductivo*: Para el paso inductivo hay 3 posibilidades, tomando por HI que la propiedad se cumple para φ con menor complejidad:
  - $\varphi = \neg\psi$

$$\begin{aligned} \mathcal{A} \vDash \varphi[v] &\iff \mathcal{A} \nvDash \psi[v] \\ &\iff \psi \notin \Delta && \text{por } HI \\ &\iff \neg\psi \in \Delta && \text{por propiedad de } \Delta \end{aligned}$$

  - $\varphi = \psi \to \rho$.

(⇒)

$$\begin{aligned} \mathcal{A} \vDash \varphi[v] &\iff \mathcal{A} \nvDash \psi[v] \vee \mathcal{A} \vDash \rho[v] \\ &\iff \psi \notin \Delta \vee \rho \in \Delta && \text{por } HI \\ &\iff \neg\psi \in \Delta \vee \rho \in \Delta && \text{por propiedad de } \Delta \\ &\implies \Delta \vdash \psi \to \rho \\ &\implies \psi \to \rho \in \Delta && \text{por propiedad de } \Delta \end{aligned}$$

(⇐)

$$\begin{aligned} \varphi \in \Delta &\implies \psi \notin \Delta \vee (\psi \in \Delta \wedge \Delta \vdash \rho) && MP \text{ en 2do caso} \\ &\implies \psi \notin \Delta \vee (\psi \in \Delta \wedge \rho \in \Delta) && \text{por propiedad de } \Delta \\ &\implies \psi \notin \Delta \vee \rho \in \Delta \\ &\iff \mathcal{A} \nvDash \psi[v] \vee \mathcal{A} \vDash \rho[v] && \text{por } HI \\ &\iff \mathcal{A} \vDash (\psi \to \rho)[v] \end{aligned}$$

- $\varphi = (\forall x)\psi$

  ($\Rightarrow$) Supongamos que $\mathcal{A} \vDash (\forall x)\psi[v]$, luego para todo $t \in A$, $\mathcal{A} \vDash \psi[v(x = t)]$. Y supongamos que $\neg(\forall x)\psi \to \neg(\psi[x/c]) \in \Theta$.

  En particular se tiene que $\mathcal{A} \vDash \psi[v(x = c)]$, y por la definición de v dada, $\mathcal{A} \vDash [v(x = \tilde{v}(c))]$. Luego por el lema de sustitución $\mathcal{A} \vDash (\psi[x/c])[v]$. Y por HI, $\psi[x/c] \in \Delta$ (con lo que por propiedad de Δ, $\neg(\psi[x/c]) \notin \Delta$).

  Ahora veamos que $\neg(\forall x)\psi \in \Delta$. Supongamos que $\neg(\forall x)\psi \in \Delta$, con lo que $\Delta \vdash \neg(\forall x)\psi$. Como $\Delta \supseteq \Theta$, $\neg(\forall x)\psi \to \neg(\psi[x/c]) \in \Delta$, de forma que $\Delta \vdash \neg(\forall x)\psi \to \neg(\psi[x/c])$. Luego por MP tenemos $\Delta \vdash \neg(\psi[x/c])$, y por propiedad de Δ, $\neg(\psi[x/c]) \in \Delta$.
  Con esto se concluye que $(\forall x)\psi \in \Delta$.

  ($\Leftarrow$) Supongamos que $\mathcal{A} \nvDash \varphi[v]$, luego existe $t \in A$, $\mathcal{A} \nvDash \psi[v(x = t)]$. Luego, sea ψ' una variable alfabética de ψ tal que x sea reemplazable por t en ψ', de forma que $\mathcal{A} \nvDash \psi'[v(x = t)]$.

  Como $\tilde{v}(t) = t$, $\mathcal{A} \nvDash \psi'[v(x = \tilde{v}(t))]$. Y por el lema de sustitución se tiene que $\mathcal{A} \nvDash (\psi'[x/t])[v]$, con lo que por HI, $\psi'[x/t] \notin \Delta$.

  Ahora veamos que $(\forall x)\psi' \notin \Delta$. Supongamos que $(\forall x)\psi' \in \Delta$, con lo que $\Delta \vdash (\forall x)\psi'$. Sabemos por $SQ4$ que $\vdash (\forall x)\psi' \to \psi'[x/t]$. Por MP concluimos que $\Delta \vdash \psi'[x/t]$, y por propiedad de Δ, $\psi'[x/t] \in \Delta$.
  Y por equivalencia de variantes alfabéticas, $(\forall x)\psi \notin \Delta$ (llegando a un absurdo).

□

51.1.5. Paso 5: Restringir $\mathcal{A}$ y v al lenguaje original $\mathcal{L}$

Volviendo al lenguaje original $\mathcal{L}$, definimos $\mathcal{B}$ como la restricción de $\mathcal{A}$ a $\mathcal{L}$ (i.e. ya no interpreto las nuevas constantes).

Del paso 4 sabemos que para toda $\varphi \in FORM(\mathcal{L}')$, $\mathcal{A} \vDash \varphi[v] \iff \varphi \in \Delta$ (recordar que $\Delta \supseteq \Gamma$). Adicionalmente entonces, si $\varphi \in \Gamma \subseteq FORM(\mathcal{L})$ tenemos que $\mathcal{A} \vDash \varphi[v] \iff \mathcal{B} \vDash \varphi[v]$.

Luego, para Γ consistente, enconrtramos una $\mathcal{L}$-estructura $\mathcal{B}$ tal que $\mathcal{B} \vDash \varphi[v]$ para toda $\varphi \in \Gamma$.
Con esto concluimos que Γ es satisfacible.

Corolario

Γ es consistente $\iff$ Γ es satisfacible.

51.2. Teorema de Löwenheim-Skolem

Teorema: (sin igualdad)

Sea $\mathcal{L}$ numerable y sin igualdad. Si $\Gamma \subseteq FORM(\mathcal{L})$ es satisfacible, es satisfacible en un modelo infinito numerable.

Demostración. Es lo visto en el teorema previo, si $\mathcal{L}$ es numerable, $A = FORM(\mathcal{L})$ es infinito numerable. □

Teorema: (con igualdad)

Sea $\mathcal{L}$ numerable y con igualdad. Si $\Gamma \subseteq FORM(\mathcal{L})$ es satisfacible, es satisfacible en un modelo finito o infinito numerable.

Teorema: (ascendente)

Si $\mathcal{L}$ numerable y $\Gamma \subseteq FORM(\mathcal{L})$ tiene modelo infinito, tiene modelo en cualquier cardinalidad.

52. Compacidad

Teorema: (Completitud fuerte, Gödel)

$\Gamma \vDash \varphi \implies \Gamma \vdash \varphi$

Demostración. Igual que para la lógica proposicional. □

Corolario

$\Gamma \vDash \varphi \iff \Gamma \vdash \varphi$

Teorema: (Compacidad)

Sea $\Gamma \subseteq FORM(\mathcal{L})$. Si todo Δ finito tal que $\Delta \subseteq \Gamma$ es satisfacible, entonces Γ es satisfacible.

Demostración. Igual que para la lógica proposicional. □

53. Aplicaciones de la compacidad

53.1. No expresividad

Teorema

Si Γ tiene modelos arbitrariamente grandes, tiene modelo infinito.

Demostración. Definimos (en el lenguaje con solo la igualdad)

$$\begin{aligned}\varphi_2 &= (\exists x)(\exists y)x \neq y\\ \varphi_3 &= (\exists x)(\exists y)(\exists z)(x \neq y \wedge x \neq z \wedge y \neq z)\\ &\vdots\\ \varphi_n &= \text{“hay al menos n elementos”}\end{aligned}$$

- Por hipótesis, todo subconjunto finito de $\Gamma \cup \{\varphi_i \mid i \geq 2\}$ tiene modelo.
- Por compacidad, $\Gamma \cup \{\varphi_i \mid i \geq 2\}$ tiene algún modelo $\mathcal{M}$.
- $\mathcal{M}$ tiene que ser infinito.

Con ello se puede concluir

- $\mathcal{A}$ es infinito $\iff \mathcal{A} \vDash \{\varphi_i \mid i \geq 2\}$
- y no existe Γ tal que $\mathcal{A}$ finito $\iff \mathcal{A} \vDash \Gamma$.

□

53.2. Modelos no estándar

Consideremos un lenguaje $\mathcal{L} = \{0, S, <, +, \cdot\}$ con igualdad. Consideremos la estructura $\mathcal{N} = \langle \mathbb{N}; 0, S, <, +, \cdot \rangle$ con interpretación usual. Luego, sea

$$Teo(\mathcal{N}) = \{\varphi \in FORM(\mathcal{L}) \ : \ \varphi \text{ es sentencia y } \mathcal{N} \vDash \varphi\}$$

Expandimos el lenguaje con una nueva constante c y definimos

$$\Gamma = \{0 < c,\ S(0) < c,\ S(S(0)) < c,\ S(S(S(0))) < c, \ldots\}$$

Se puede observar que

- cada subconjunto finito de $\Gamma \cup Teo(\mathcal{N})$ tiene modelo,
- y por compacidad, $\Gamma \cup Teo(\mathcal{N})$ tiene modelo,
- con lo que por el teorema de Löwenheim-Skolem $\Gamma \cup Teo(\mathcal{N})$ tiene un modelo numerable de la forma

$$\mathcal{M} = \langle M; 0^{\mathcal{M}}, S^{\mathcal{M}}, <^{\mathcal{M}}, +^{\mathcal{M}}, \cdot^{\mathcal{M}}, c^{\mathcal{M}} \rangle$$

Luego, sea $\mathcal{M}'$ la restricción de $\mathcal{M}$ al lenguaje original $\mathcal{L}$. Se tiene que

- $\mathcal{N} \vDash \varphi \iff \mathcal{M}' \vDash \varphi$ para toda sentencia $\varphi \in FORM(\mathcal{L})$, Luego
  - $\mathcal{N} \vDash \varphi \implies \varphi \in Teo(\mathcal{N}) \implies \mathcal{M} \vDash \varphi \implies \mathcal{M}' \vDash \varphi$
  - $\mathcal{N} \nvDash \varphi \implies \mathcal{N} \vDash \neg\varphi \implies \neg\varphi \in Teo(\mathcal{N}) \implies \mathcal{M} \vDash \neg\varphi \implies \mathcal{M}' \vDash \neg\varphi \implies \mathcal{M}' \nvDash \varphi$
- Pero se tiene que $\mathcal{N}$ y $\mathcal{M}'$ no son isomorfos: $c^{\mathcal{M}}$ es inalcanzable en $\mathcal{M}'$.

54. Repaso de Máquinas de Turing

Recordar que una máquina de Turing es una tupla

$$M = (\Sigma, Q, T, q_0, q_f)$$

donde

- Σ (finito) es el conjunto de símbolos que puede escribir en la cinta
- Q (finito) es el conjunto de estados, de los cuales se distinguen:
  - $q_0 \in Q$ es el estado inicial
  - $q_f \in Q$ es el estado final
- $T \subseteq Q \times \Sigma \times \Sigma \cup \{L, R\} \times Q$ es la tabla finita de instrucciones‘

Dicha máquina, fijando $\Sigma = \{1, *\}$ con esntrada $w \in \{1\}^+$ término (notado $M(w)\downarrow$) sí y sólo si partiendo de w en la cinta de entrada y la cabeza leyendo el primer caracter después de w,

$$\ldots\ *\ *\ 1\ 1\ \ldots\ 1\ \underset{q_0}{*}\ \ldots$$

llega al estado q_f después de una cantidad finita de pasos.
No es computable determinar si una máquina de Turing termina o no.

55. Indecidibilidad de la lógica de primer orden

La idea de la demostración de queprimer orden es incedidible consiste en fijar un lenguaje adecuado $\mathcal{L}$, y dada un máquina M y $w \in \{1\}^+$, construir (uniformemente) una sentencia $\varphi_{M,w} \in FORM(\mathcal{L})$ tal que

$$M(w)\downarrow \iff \vdash \varphi_{M,w}$$

Si el problema de determinar si vale $\vdash \psi$ o $\nvdash \psi$ para $\psi \in FORM(\mathcal{L})$ fuese computable, en particular sería computable determinar si $\vdash \varphi_{M,w}$ o $\nvdash \varphi_{M,w}$ para cualquier máquina M y entrada w. Como esto último no escomputable, tampoco es computable determinar si vale $\vdash \psi$ o $\nvdash \psi$ para cualquier $\psi \in FORM(\mathcal{L})$.

Dados M y w, $\varphi_{M,w}$ es una fórmula de $\mathcal{L}$ que se construye computablemente a partir de M y w. Donde $\varphi_{M,w}$ describe el comportamiento de M con entrada w en una cierta interpretación $\mathcal{A}$. Se dice que $\varphi_{M,w}$ es una fórmula programa.

55.1. El lenguaje $\mathcal{L}$

El lenguaje $\mathcal{L}$ consiste de

- Símbolos de constante:
  - Uno solo: ϵ
- Símbolos de función
  - La función 1 unaria
  - La función * unaria
- Símbolos de relación
  - Infinitos (tantas como necesitemos) símbolos de relaciones binarias
  - Sea $E = \{q_0, q_f, p, q, r, \ldots\}$ un conjunto infinito de estados que podemos llegar a usar en máquinas de Turing.
    - Cada máquina particular usará solo una cantidad finita de estados de E.
  - Los símbolos de relación son: $R_{q_0}, R_{q_f}, R_p, R_q, R_r, \ldots$

Donde se utilizará como notación:

- Si t es un término de $\mathcal{L}$, $1(t)$ lo notamos $1t$
- Si t es un término de $\mathcal{L}$, $*(t)$ lo notamos $*t$

Por ejemplo:

- $1(x)$ lo notamos $1x$
- $1(1(\epsilon))$ lo notamos 11ϵ
- $1(1(*(1(y))))$ lo notamos $11 * 1y$

55.1.1. La interpretación $\mathcal{A}$

Dada una máquina $M = (\Sigma, Q, T, q_0, q_f)$ y una entrada $w \in \{1\}^+$, fijamos una interpretación $\mathcal{A} = \mathcal{A}_{M,w}$

- El universo: $A = \{1, *\}^*$ = cadenas finitas sobre $\{1, *\}$, lo cual va a representar datos en la cinta de M.
- $\epsilon_{\mathcal{A}}$ = cadena vacía. La cinta es infinita, pero infinitos * se representa como ϵ.
- Las funciones $1_{\mathcal{A}} : A \to A$ y $*_{\mathcal{A}} : A \to A$ se interpretan así:
  - $1_{\mathcal{A}}(x) = 1x$, o sea la cadena que empieza por 1 y sigue con x
  - $*_{\mathcal{A}}(x) = *x$, o sea la cadena que empieza por $*$ y sigue con x
- Para $q \in Q$, $(R_q)_{\mathcal{A}}(x, y)$ es verdadero sí y sólo si la máquina M con entrada w llega a una configuración en la que:
  - El estado es q
  - En la cinta está escrito x en orden inverso y a continuación y
  - La cabeza de M apunta al primer caracter de y

55.2. Definición de la fórmula-programa $\varphi_{M,w}$

Dada un máquina $M = (\Sigma, Q, T, q_0, q_f)$ y una entrada $w = \overbrace{1 \dots 1}^{k}$ fijamos la interpretación $\mathcal{A} = \mathcal{A}_{M,w}$ que acabamos de ver. Luego, definimos las sguientes fórmulas:

- $\varphi_0 := R_{q_0}(\overbrace{1 \dots 1}^{k}\epsilon, \epsilon)$
  - i.e. "el estado inicial es alcanzable"
  - $\mathcal{A} \vDash \varphi_0$
- $\varphi_f := (\exists x)(\exists y)R_{q_f}(x, y)$
  - i.e. "el estado final es alcanzable"
  - $\mathcal{A} \vDash \varphi_f \iff M(w)\downarrow$

Se tiene para cada instrucción $I \in T$:

- Si I sice si M está en el estado q y lee un 1, escribir b y pasar al estado r, se define
$$\psi_I := (\forall x)(\forall y)(R_q(x, 1y) \to R_r(x, by))$$
- Si I sice si M está en el estado q y lee un $*$, escribir b y pasar al estado r, se define
$$\begin{aligned}\psi_I := &(\forall x)(\forall y)(R_q(x, *y) \to R_r(x, by)) \wedge \\ &(\forall x)(R_q(x, \epsilon) \to R_r(x, b\epsilon))\end{aligned}$$
- Si I sice si M está en el estado q y lee un 1, moverse a la izquierda y pasar al estado r, se define
$$\begin{aligned}\psi_I := &(\forall x)(\forall y)(R_q(1x, 1y) \to R_r(x, 11y)) \wedge \\ &(\forall x)(\forall y)(R_q(*x, 1y) \to R_r(x, *1y)) \wedge \\ &(\forall y)(R_q(\epsilon, 1y) \to R_r(\epsilon, *1y))\end{aligned}$$
- Si I sice si M está en el estado q y lee un $*$, moverse a la izquierda y pasar al estado r, se define
$$\begin{aligned}\psi_I := &(\forall x)(\forall y)(R_q(1x, *y) \to R_r(x, 1 * y)) \wedge \\ &(\forall x)(\forall y)(R_q(*x, *y) \to R_r(x, * * y)) \wedge \\ &(\forall y)(R_q(\epsilon, *y) \to R_r(\epsilon, * * y)) \wedge \\ &(\forall x)(R_q(*x, \epsilon) \to R_r(x, \epsilon)) \wedge \\ &(\forall x)(R_q(1x, \epsilon) \to R_r(x, 1\epsilon)) \wedge \\ &(R_q(\epsilon, \epsilon) \to R_r(\epsilon, \epsilon))\end{aligned}$$

(Análogo para moverse a la derecha).

Recordar que la máquina $M = (\Sigma, Q, T, q_0, q_f)$ tiene siempre un conjunto finito de instrucciones T. Definimos luego

$$\varphi_{M,w} := (\varphi_0 \wedge \bigwedge_{I \in T} \psi_I) \to \varphi_f$$

Proposición

Si $\mathcal{A} \vDash \varphi_{M,w} \iff M(w)\downarrow$

Demostración. Sabemos que $\mathcal{A} \vDash \varphi_0$, y sabemos que $\mathcal{A} \vDash \varphi_f \iff M(w)\downarrow$. Luego, se puede ver que $\mathcal{A} \vDash \psi_I$ para cada $I \in T$. Con lo que $\mathcal{A} \vDash \varphi_{M,w} \iff \mathcal{A} \vDash \varphi_f \iff M(w)\downarrow$. □

56. Entscheidungsproblem

Teorema

$\vdash \varphi_{M,w} \iff M(w)\downarrow$

Demostración.

($\Rightarrow$) Si $\vdash \varphi_{M,w}$ entonces $\vDash \varphi_{M,w}$, es decir, $\varphi_{M,w}$ es verdadera en toda interpretación. En particular, $\mathcal{A} \vDash \varphi_{M,w}$. Luego $M(w)\downarrow$.

($\Leftarrow$) Se va a proponer una idea de la demostración. Si $M(w)\downarrow$ entonces existe un cómputo de $M(w)\downarrow$:

$$(x_1, r_1, y_1) \rightsquigarrow (x_2, r_2, y_2) \rightsquigarrow \ldots \rightsquigarrow (x_n, r_n, y_n)$$

con $x_i, y_i \in \{1, *\}^*$, $r_i \in Q$, $x1 = w$, $r_1 = q_0$, $y_1 = \epsilon$, $r_n = q_f$.
Cada (x_i, r_i, y_i) representa una configuración del cómputo $M(w)$:

- El estado es r_i
- La cinta contiene $\cdots * * * [x_i][y_i] * * * \cdots$
- La cabeza está apuntando al primer caracter de y_i‘

Cada paso de la ejecución coincide con una sustitución de una de las fórmulas ψ_I.

- Cómputo de $M(w)$ = demostración de $\varphi_{M,w}$
- Fórmula $\varphi_{M,w}$ = programa de M

Recordemos que

$$\varphi_{M,w} := (\varphi_0 \wedge \bigwedge_{I \in T} \psi_I) \rightarrow \varphi_f$$

Supongamos que M con entrada 111 da el siguiente cómputo:

$$(111, q_0, *) \overset{(q_0,*,L,q_1)\in T}{\rightsquigarrow} (11, q_1, 1) \overset{(q_1,1,L,q_f)\in T}{\rightsquigarrow} (1, q_f, 11)$$

Veamos que $\{q_0\} \cup \{\psi_I \mid I \in T\} \vdash \varphi_f$. Esto pruba que $\vdash \varphi_{M,w}$.

- Recordemos que $\psi_{(q_0,*,L,q_1)} := \ldots \wedge (\forall x)(R_{q_0}(1x, \epsilon) \rightarrow R_{q_1}(x, 1\epsilon)) \wedge \ldots$, donde por $SQ4$, $\{q_0\} \cup \{\psi_I \mid I \in T\} \vdash (R_{q_0}(111\epsilon, \epsilon) \rightarrow R_{q_1}(11\epsilon, 1\epsilon))$.
- Recordemos que $\psi_{(q_1,1,L,q_f)} := (\forall x)(\forall y)(R_{q_1}(1x, 1y) \rightarrow R_{q_f}(x, 11y)) \wedge \ldots$, donde por $SQ4$, $\{q_0\} \cup \{\psi_I \mid I \in T\} \vdash (R_{q_1}(11\epsilon, 1\epsilon) \rightarrow R_{q_f}(1\epsilon, 11\epsilon))$.
- Recordemos que $\varphi_0 := R_{q_0}(111\epsilon, \epsilon)$
- Por MP, concluimos $\{q_0\} \cup \{\psi_I \mid I \in T\} \vdash R_{q_f}(1\epsilon, 11\epsilon)$
- De esto, se puede concluir $\{q_0\} \cup \{\psi_I \mid I \in T\} \vdash \underbrace{(\exists x)(\exists y)R_{q_f}(x, y)}_{\varphi_f}$

□

Teorema

Sea $\mathcal{L}$ el lenguaje descripto y sea $\psi \in FORM(\mathcal{L})$. El problema de decidir si $\vdash \psi$ o $\nvdash \psi$ no es computable.

Demostración.
Supongamos que hay un programa que dada $\psi \in FORM(\mathcal{L})$ devuelve verdadero sí y sólo si $\vdash \psi$.
Dada M y w, habría un procedimiento para decidir si $M(w)\downarrow$ o $M(w)\uparrow$:

1. Construir $\varphi_{M,w}$ (esto se have computablemente)
2. Si $\vdash \varphi_{M,w}$ entonces $M(w)\downarrow$; si no $M(w)\uparrow$

□

57. Teorema de incompletitud de Gödel

Pendiente.