Lógica y Computabilidad
Apunte 1er Parcial

Sebastián Taboh

28 de septiembre de 2017

Este es un apunte de los temas del primer parcial de la cursada de Lógica y Computabilidad de verano de 2017, y contiene extractos de apuntes hechos por docentes de la materia así como agregados personales.

Cabe aclarar que las referencias a las guías de ejercicios son las de ese cuatrimestre y que puede haber cambios en los temas que se aborden en distintas cursadas. Además, el fin del apunte **no** es explicar los temas, es un apunte útil para llevar al parcial.

En caso de encontrar errores, por favor comunicarlo por mail a sebi_282@hotmail.com.

Índice

1. Funciones primitivas recursivas y clases PRC

1.1. Lista de funciones p.r.

- **Aritméticas**
  - $x + y$
  - $x * y$
  - x^y
  - $x!$
  - $x \dot{-} y$
  - $p(x)$
  - $|x - y|$
  - $\alpha(x)$
  - $máx\{x, y\}$
  - $mín\{x, y\}$
  - $\left\lfloor \frac{x}{y} \right\rfloor$
  - $resto(x, y)$
  - $raíz(x, y)$
  - $nprimo(i)$
- **Pares**
  - $\langle x, y \rangle$
  - $l(z)$
  - $r(z)$
- **Sobre predicados**: las últimas 6 funciones son p.r. si el predicado P es p.r.
  - $x = y$
  - $x \leq y$
  - $x < y$
  - $\neg P$
  - $P \vee Q$
  - $P \wedge Q$
  - $y|x$
  - $primo(x)$
  - $par(x)$
  - $cantidad_P(x_1, ..., x_n, y, z)$
  - $todos_P(x_1, ..., x_n, y, z)$
  - $alguno_P(x_1, ..., x_n, y, z)$

- $mínimo_P(x_1, ..., x_n, y, z)$
- $máximo_P(x_1, ..., x_n, y, z)$
- $único_P(x_1, ..., x_n, y, z)$

- **Secuencias**
  - Codificación de secuencias
  - $|xs|$
  - $xs[i]$
  - $[x]$: crea la lista con el elemento x
  - $máx(xs)$: devuelve el máximo elemento de la lista
  - $divisoresComunes(xs, ys)$: da la lista de divisores comunes
  - $pertenece(x, xs)$
  - $subsecuencia(xs, d, h)$: da la subsecuencia desde el índice d hasta h
  - $insertar(i, x, xs)$: insertar en la posición i el elemento x en la lista xs y desplazar los elementos desde la posición i un lugar
  - $concatenar(xs, ys)$
  - $filtrar_P(xs)$: da la lista de los elementos de xs que cumplen el predicado P
  - $suma(xs)$
  - $producto(xs)$
  - $cuántos(x, xs)$: cuenta las apariciones de x en xs
  - $mismos(xs, ys)$: indica si están los mismos elementos en las mismas cantidades
  - $invertir(xs)$: invierte la lista
  - $ordenar(xs)$

1.2. Demostraciones de algunas funciones p.r

- $p(x)$
- $|x - y|$
- $[x]$: crea la lista con el elemento x
- $máx(xs)$: devuelve el máximo elemento de la lista

$$máx(xs) = \min_{t \leq xs} \Big((\forall\ j)_{\leq |xs|} (xs[j] \leq t) \Big)$$

- $pertenece(x, xs) = (\exists i)_{\leq |xs|}\ (1 \leq i \wedge xs[i] = x)$
- $subsecuencia(xs, d, h)$: da la subsecuencia desde el índice d hasta h

$$subsecuencia(xs, d, h) = (d \leq h) * \prod_{i=0}^{h \dot{-} d} {p_{d+i}}^{xs[d+i]} + (d > h) * []$$

- $insertar(i, x, xs)$: insertar en la posición i (i en rango) el elemento x en la lista xs y desplazar los elementos desde la posición i un lugar

$$insertar(i, x, xs) = (1 \leq i \leq |xs|) * \left(\prod_{j=1}^{i-1} {p_j}^{xs[j]} \right) * {p_i}^{x} * \prod_{j=1}^{|xs|-i+1} {p_{i+j}}^{xs[i-1+j]}$$

- $inserciónOrdenada(xs, x)$: toma una lista ordenada y un elemento y lo inserta ordenadamente

$$inserciónOrdenada(xs, x) = insertar\Big(\min_{t \leq |xs|} (xs[t+1] \geq x), x, xs \Big)$$

- $concatenar(xs, ys)$

$$\begin{aligned} concatenar(xs, ys) &= concatenar'(ys, xs) \\ concatenar'(ys, []) &= ys \\ concatenar'(ys, [x_1, ..., x_n]) &= x_1 : (concatenar'(ys, [x_2, ..., x_n])) \end{aligned}$$

  Hay que ver que (:) (la función que agrega un elemento al principio de la lista) es p.r.

$$: (x, xs) = 2^x * \prod_{i=1}^{|xs|} p_{i+1}^{xs[i]}$$

  Otra forma:

$$\begin{aligned} concatenar(xs, ys) = \min_{t \leq \prod_{i=1}^{|xs|+|ys|} {p_i}^{máx\{máx(xs), máx(ys)\}}} \Big(& |t| = |xs| + |ys| \\ & \wedge subsecuencia(t, 1, |xs|) = xs \\ & \wedge subsecuencia(t, |xs| + 1, |t|) = ys \Big) \end{aligned}$$

- $filtrar_P(xs)$: da la lista de los elementos de xs que cumplen el predicado P en el mismo orden en el que aparecían en xs (aunque haya repetidos)

$$\begin{aligned} filtrar_P([]) &= [] \\ filtrar_P([x_1, ..., x_n]) &= P(x_1) * \Big(x_1 : \Big(filtrar_P([x_2, ..., x_n]) \Big) \Big) + \alpha(P(x_1)) * filtrar_P([x_2, ..., x_n]) \end{aligned}$$

  Otra forma:

$$\begin{aligned} filtrar_P(xs) = \min_{t \leq \text{cotaÚtil}(xs)} \Big(& todos_P(t, 1, |t|) \\ & \wedge cantidad_P(xs, 1, |xs|) = |t| \\ & \wedge (\forall\ j)_{\leq |t|} (cuántos(t[j], xs) = cuántos(t[j], t)) \Big) \end{aligned}$$

- $cotaÚtil(xs)$: devuelve una cota que generalmente es útil :P

$$cotaÚtil(xs) = \prod_{i=1}^{|xs|} p_i^{max(xs)}$$

- $suma(xs)$: devuelve la suma de los elementos de la lista

$$suma(xs) = \sum_{i=1}^{|xs|} xs[i]$$

- $producto(xs)$: devuelve el producto de los elementos de la lista

$$producto(xs) = \prod_{i=1}^{|xs|} xs[i]$$

- $cuántos(x, xs)$: cuenta las apariciones de x en xs

$$cuántos(x, xs) = cantidad_P(xs, x, 1, |xs|) \text{ donde } P(xs, z, t) = (xs[t] = z) \text{ es p.r.}$$

- $mismos(xs, ys)$: indica si están los mismos elementos en las mismas cantidades

$$mismos(xs, ys) = \Big(|xs| = |ys| \wedge \ (\forall i)_{\leq |xs|}(cuántos(xs[i], xs) = cuántos(xs[i], ys))\Big)$$

1.3. Otros resultados

Se sigue del Ejercicio 5 de la Práctica 1 que:
Sea $f : \mathbb{N} \to \mathbb{N}$ una función p.r., $A \subset \mathbb{N}$ un conjunto finito, y g una función total tal que $g(x) = f(x)\ \forall\ x \notin A$, entonces g es p.r.

2. Funciones $\mathcal{S}$-computables

Definición: Una función (parcial) $f : \mathbb{N}^m \to \mathbb{N}$ es $\mathcal{S}$-parcial computable si existe un programa P tal que $f(r_1, ..., r_m) = \Psi_P^{(m)}(r_1, ..., r_m)$ para todo $(r_1, ..., r_m) \in \mathbb{N}^m$.
La función f es $\mathcal{S}$-computable si es parcial computable y total.

Teorema: Si $p : \mathbb{N}^{n+1} \to \{0, 1\}$ es un predicado computable entonces $\min\limits_t p(x_1, ..., x_n, t)$ es parcial computable y $(\exists\ t)\ p(x_1, ..., x_n, t)$ es parcial computable.

Teorema: La clase de funciones computables es una clase PRC.

Corolario: Toda función primitiva recursiva es computable.

2.1. Codificación de variables y etiquetas en $\mathcal{S}$

Ordenamos las variables

$$Y, X_1, Z_1, X_2, Z_2, ...$$

y las etiquetas

$$A, B, C, ..., Z, AA, AB, ..., AZ, BA, ...$$

Escribimos $\#(V)$ para la posición que ocupa la variable en la lista. Ídem para $\#(L)$ con la etiqueta L. Las posiciones empiezan desde 1.

2.2. Codificación de instrucciones en $\mathcal{S}$

Codificamos a la instrucción I con $\#(I) = \langle a, \langle b, c\rangle\rangle$ donde

1. si I tiene etiqueta L, entonces $a = \#(L)$, si no $a = 0$
2. si la variable mencionada en I es V entonces $c = \#(V) - 1$
3. si la instrucción I es
   a) $V \leftarrow V$ entonces $b = 0$
   b) $V \leftarrow V + 1$ entonces $b = 1$
   c) $V \leftarrow V - 1$ entonces $b = 2$
   d) IF $V \neq 0$ GOTO L' entonces $b = \#(L') + 2$

En resumen, codificamos

- $[L]\ V \leftarrow V$ como $\Big\langle \#(L), \langle 0, \#(V) - 1\rangle\Big\rangle$
- $[L]\ V \leftarrow V + 1$ como $\Big\langle \#(L), \langle 1, \#(V) - 1\rangle\Big\rangle$
- $[L]\ V \leftarrow V \dot{-} 1$ como $\Big\langle \#(L), \langle 2, \#(V) - 1\rangle\Big\rangle$
- $[L]$ IF $V \neq 0$ GOTO L' como $\Big\langle \#(L), \langle \#(L') + 2, \#(V) - 1\rangle\Big\rangle$

donde $\#(V), \#(L)$ y $\#(L')$ indican las posiciones de V, L y L' en las enumeraciones correspondientes ($\#(L)$ es 0 si la instrucción no tiene etiqueta).

2.3. Codificación de programas en $\mathcal{S}$

Un programa P es una lista (finita) de instrucciones $I_1, ..., I_k$. Codificamos al programa P con

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

La instrucción final de un programa no puede ser $Y \leftarrow Y$.

$\textsc{Halt}(x, y) : \mathbb{N}^2 \to \{0, 1\}$ es verdadero sii el programa con número y y entrada x no se indefine.

$$\textsc{Halt}(x, y) = \begin{cases} 1 & \text{si } \Psi_P^{(1)}(x) \downarrow \\ 0 & \text{si no} \end{cases}$$

2.4. Universalidad

Para cada $n > 0$ definimos, dado un programa P tal que $\#(P) = e$,

$$\Phi^{(n)}(x_1, ..., x_n, e) = \Psi_P^{(n)}(x_1, ..., x_n) = \text{salida del programa } e \text{ con entrada } x_1, ..., x_n$$

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

2.5. Step Counter

Definimos

$$\text{STP}^{(n)}(x_1, ..., x_n, e, t) = \text{ el programa } e \text{ termina en } t \text{ o menos pasos con entrada } x_1, ..., x_n$$

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

2.6. Snapshot

Definimos

$$\begin{aligned}\text{SNAP}^{(n)}(x_1, ..., x_n, e, t) = &\text{ representación de la configuración instantánea del programa } e \\ &\text{con entrada } x_1, ..., x_n \text{ en el paso } t \\ = &\text{ ⟨número de instrucción, lista de valores de las variables⟩ en el paso } t \\ &\text{de la ejecución del programa } e \text{ con entrada } (x_1, ..., x_n) \text{ donde el} \\ &\text{orden de las variables en la lista es } Y, X_1, Z_1, X_2, Z_2, ...\end{aligned}$$

Teorema: Para cada $n > 0$, el predicado $\text{SNAP}^{(n)}(x_1, ..., x_n, e, t)$ es p.r.

2.7. Macros útiles

- $V_i \leftarrow k$
- $V_i \leftarrow V_j + k$
- IF $V_i = 0$ GOTO L
- GOTO L
- IF $r(V_1, ..., V_n)$ GOTO L donde $r : \mathbb{N}^n \to \{0, 1\}$ es un predicado p.r.
- IF $r(V_1, ..., V_n)$ THEN P ELSE Q donde P y Q son dos programas autocontenidos con etiquetas disjuntas y $r : \mathbb{N}^n \to \{0, 1\}$ es un predicado p.r.
- WHILE $r(V_1, ..., V_n)$ P donde P y Q son dos programas autocontenidos con etiquetas disjuntas y $r : \mathbb{N}^n \to \{0, 1\}$ es un predicado p.r.

2.8. Otros resultados

Se sigue del Ejercicio 5 de la Práctica 2 que:
Sean $f : \mathbb{N} \to \mathbb{N}$ una función biyectiva y $\mathcal{S}$-computable (total) y $g, h : \mathbb{N} \to \mathbb{N}$ con h $\mathcal{S}$-computable tales que

$$f \circ g = h$$

Entonces g es computable.

3. Funciones no computables y conjuntos c.e

Teorema de la Forma Normal: Sea $f : \mathbb{N}^n \to \mathbb{N}$ una función parcial computable. Entonces existe un predicado p.r. $P : \mathbb{N}^{n+1} \to \mathbb{N}$ tal que

$$f(x_1, ..., x_n) = l\Big(\min_z P(x_1, ..., x_n, z)\Big)$$

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 ($\min_t q(x_1, ..., x_n, t)$ donde siempre existe al menos un t tal que $q(x_1, ..., x_n, t)$ es verdadero)

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

$$\Phi^{(n+m)}_y(x_1, ..., x_m, u_1, ..., u_n) = \Phi^{(m)}_{S^n_m(u_1,...,u_n,y)}(x_1, ..., x_m)$$

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

$$\Phi^{(n)}_e(x_1, ..., x_n) = g(e, x_1, ..., x_n)$$

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

$$\Phi^{(n)}_e(x_1, ..., x_n) = g(e, x_1, ..., x_n)$$

Proposición: Sea h parcial computable. Hay infinitos e tales que $\Phi_e(x) = h(e)$.

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

3.1. Conjuntos en teoría de la computabilidad

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

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

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

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

Teorema: Si A es computable entonces es c.e.

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

Teorema: A es computable sii A y $\overline{A}$ son c.e.

Definición: Definimos $W_n = \{x : \Phi_n(x) \downarrow\}$ = dominio del n-ésimo programa.

Teorema: Un conjunto A es c.e. sii existe un n tal que $A = W_n$.

Teorema: $K = \{n : n \in W_n\} = \{n : \Phi_n(n) \downarrow\} = \{n : \textsc{Halt}(n, n)\}$ es c.e. pero no computable.

Teorema: Si A es c.e., existe un predicado p.r. $P : \mathbb{N}^2 \to \mathbb{N}$ tal que

$$A = \{x : (\exists\, t)P(x, t)\}$$

Teorema: Si $A \neq \emptyset$ es c.e., existe una función p.r. $f : \mathbb{N}^2 \to \mathbb{N}$ tal que

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

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

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

1. A es c.e.
2. A es el dominio de una función parcial computable.
3. A es la imagen de una función p.r.
4. A es la imagen de una función computable.
5. A es la imagen de una función parcial computable.

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

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

3.1.1. Conjuntos y funciones no computables

- $\{x : \Phi_x$ es total$\}$
- $\{x : \Phi_x$ es creciente$\}$
- $\{x : \Phi_x$ tiene dominio infinito$\}$
- $\{x : \Phi_x$ es primitiva recursiva$\}$
- $K = \{x : \textsc{Halt}(x, x)\}$
- $Tot = \{x : \Phi_x$ es total$\}$
- $\textsc{Halt}(x, y)$

3.1.2. Conjuntos no c.e.

- $\overline{K} = \{x : \Phi_x(x) \uparrow\}$
- Tot
- $\overline{Tot} = \{x : \Phi_x$ no es total$\}$

3.1.3. Conjuntos c.e.

- K