

Clase 24: Teorema de Rice y Problema de Correspondencia de Post

Aldo Berrios Valenzuela

19 de julio de 2016

1. Teorema de Rice

Antes de comenzar, recuerde que una propiedad P es un conjunto de todos aquellos lenguajes que cumplan con la propiedad (ser de contexto libre, regulares, etc.).

Teorema 1.1 (Rice). Toda propiedad no trivial de los lenguajes R.E. es no decidible

Supondremos que nos dan el lenguaje R.E. como una TM, decidir la propiedad es determinar dado M si $\mathcal{L}(M)$ tiene la propiedad ($M \in L_P$)

Nótese que el teorema de Rice no nos dice nada acerca de la máquina de Turing. **Es sólo para lenguajes, no para TM.**

Estrategia: Reducimos $L_u \leq L_P$; como L_u no es decidible, no es decidible L_P .

Demostración. Supongamos primeramente $\emptyset \notin P$ (el lenguaje vacío no tiene la propiedad P). Reducimos L_u a L_P mediante la siguiente construcción: “Sea L un lenguaje, $L \in P$, y $L = \mathcal{L}(M_L)$ para una TM M_L . Notar que M_L es fijo”. Dada una instancia $\langle M, \omega \rangle$ de U , construyo M' :

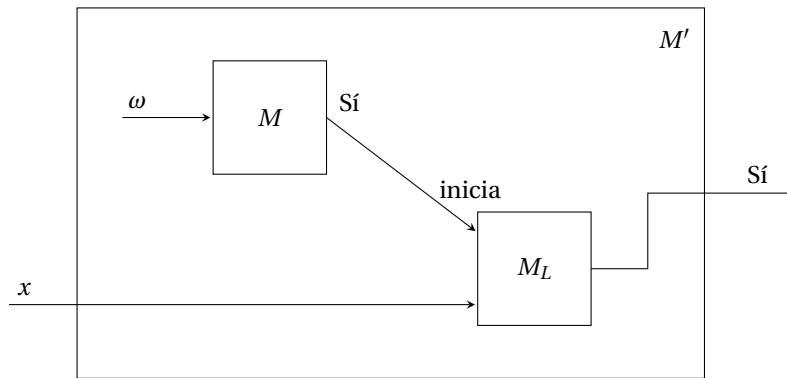


Figura 1: La máquina de Turing M' emula una máquina de Turing universal, que es no decidible. Note que si M acepta, echa a andar M_L sobre M' .

Notamos que:

$$\left. \begin{array}{l} \text{Si } \omega \in \mathcal{L}(M) \Rightarrow \mathcal{L}(M') = L \\ \text{Si } \omega \notin \mathcal{L}(M) \Rightarrow \mathcal{L}(M') = \emptyset \end{array} \right\} \begin{array}{l} \text{Si pudiera decidir si } \mathcal{L}(M) \in P, \text{ podría decidir si } \omega \in \mathcal{L}(M), \\ \text{pero } L_u \text{ no es decidible} \end{array}$$

Falta el caso $\emptyset \in P$. En tal caso, como P no es trivial, $\emptyset \notin \bar{P}$, y \bar{P} es una propiedad no trivial que no incluye \emptyset . Por el caso anterior, $L_{\bar{P}}$ no es decidible, pero $L_{\bar{P}} = \bar{L}_P$. Como \bar{L}_P no es decidible, L_P no es decidible (haciendo este truco, podríamos decidir L_u , pero L_u no es decidible, por lo tanto, se origina una contradicción). \square

Ojo, habla del lenguaje aceptado por M , no de M .

2. Problema de Correspondencia de Post (PCP)

Una instancia¹ de PCP es una colección de pares de palabras (α_i, β_i) sobre Σ , siendo la pregunta si hay una secuencia finita $\langle i_1, i_2, \dots, i_n \rangle$ tal que:

$$\alpha_{i_1} \alpha_{i_2} \cdots \alpha_{i_n} = \beta_{i_1} \beta_{i_2} \cdots \beta_{i_n} \quad (2.1)$$

Ejemplo 2.1.² Sea $\Sigma = \{0, 1\}$. Considere el Cuadro 1, que define todos los α_i 's y β_i 's de nuestro PCP. Para la secuencia finita $\langle i_1, i_2, i_3, i_4 \rangle = \langle 2, 1, 1, 3 \rangle$ vemos que este PCP sí tiene solución. Esto lo podemos verificar concatenando los strings correspondientes en orden para ambas lsisas:

$$w_2 w_1 w_1 w_3 = x_2 x_1 x_1 x_3 = 101111110 \quad (2.2)$$

Note que esta solución no es única, ya que $\langle i_1, i_2, \dots, i_8 \rangle = \langle 2, 1, 1, 3, 2, 1, 1, 3 \rangle$ es otra solución. \square

i	α_i	β_i
1	1	111
2	10111	10
3	10	0

Cuadro 1: Instancia PCP del Ejemplo 2.1. Corresponde a las definiciones de los α_i 's y β_i 's

Resulta que PCP no es decidible. Demostración es vía $L_u \leq \text{PCP}$. Idea general es que al ir construyendo la secuencia $\alpha_{i_1} \alpha_{i_2} \cdots$, vamos construyendo IDs de M , y en $\beta_{i_1} \beta_{i_2} \cdots$, vamos construyendo el ID siguiente de M , simulando una movida. Para que los α 's "alcancen" los β 's, tienen que copiar el ID actual y los β generar el siguiente, ... Si en los β 's acepta, damos la posibilidad de terminar.

Podemos usar esto para demostrar cosas más terrenales...

Teorema 2.1. No es decidible si una CFG es ambigua.

Demostración. Reducimos $\text{PCP} \leq \text{CFG ambigua}$. Sea (α_i, β_i) una instancia de PCP. La gramática:

$$S \rightarrow A \mid B, \quad A \rightarrow i A \alpha_i \mid \epsilon, \quad B \rightarrow i B \beta_i \mid \epsilon \quad (2.3)$$

es ambigua si y solo si esa instancia de PCP tiene solución. \square

¹Una instancia del problema son sus datos de entrada.

²Introduction to Automata Theory, Languages and Computation, by John E. Hopcroft, Rajeev Motwani and Jeffrey D. Ullman: Ejemplo 9.13