Clase 2 - TCVP

Resumen

Se introduce el concepto de **jerarquía de la computabilidad**: un esquema para clasificar problemas según su grado de dificultad desde la perspectiva de la computabilidad.

Clasificación de lenguajes

Sea *L* el conjunto de todos los lenguajes, representante del conjunto de todos los problemas de decisión. Y sea Σ el alfabeto de los símbolos que integran las cadenas de los lenguajes de *L* (toda cadena de un lenguaje de *L* pertenece al conjunto Σ*). Para definir la jerarquía de la computabilidad, comenzamos distinguiendo en *L* los siguientes dos conjuntos de lenguajes:

- Lenguajes **recursivamente enumerables (RE)**: existe una MT que los acepta, Si una cadena pertenece al lenguaje, la MT eventualmente se detendrá y aceptará, si la cadena no pertenece al lenguaje, la MT podría tenerse y rechazar o nunca detenerse (problemas computables decidibles y no decidibles).
  - Propiedad: sus cadenas pueden enumerarse.
- Lenguajes **recursivos (R)**: existe una MT que lo reconoce y **se detiene siempre**, ya sea qA o qR (problemas computables decidibles).

Se establece entonces que `R ⊆ RE ⊆ L`.

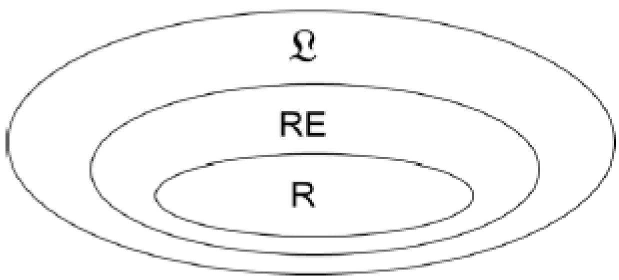


Figura 3.1. Primera versión de la jerarquía de la computabilidad.

Propiedades de los lenguajes recursivamente enumerables y los lenguajes recursivos

Lenguajes Recursivos

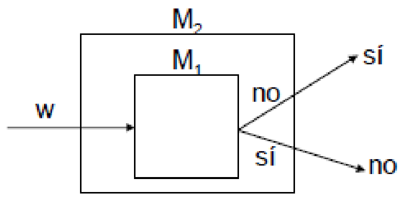
Propiedad 1: Si L ∈ R, entonces L^c ∈ R.

Es decir, si existe una MT M1 que decide L, también existe una MT M2 que decide L^c

Construcción de la MT M_2.

Si: $M_1 = (Q, \Gamma, \delta, q_0, q_A, q_R)$

entonces: $M_2 = (Q, \Gamma, \delta', q_0, q_A, q_R)$



donde las funciones de transición δ y δ´ de M_1 y M_2 son iguales salvo que los estados q_A y q_R **están permutados.**

Formalmente, para todos los estados q y q´, símbolos s y s´, y movimientos d de {L, R, S}:

- Si $\delta(q, s) = (q_A, s', d)$, entonces $\boldsymbol{\delta'(q, s) = (q_R, s', d)}$ *** se cambia q_A por q_R
- Si $\delta(q, s) = (q_R, s', d)$, entonces $\boldsymbol{\delta'(q, s) = (q_A, s', d)}$ *** se cambia q_R por q_A
- Si $\delta(q, s) = (q', s', d)$, con $q' \neq q_A$ y $q' \neq q_R$, entonces $\boldsymbol{\delta'(q, s) = (q', s', d)}$ *** los otros casos quedan igual

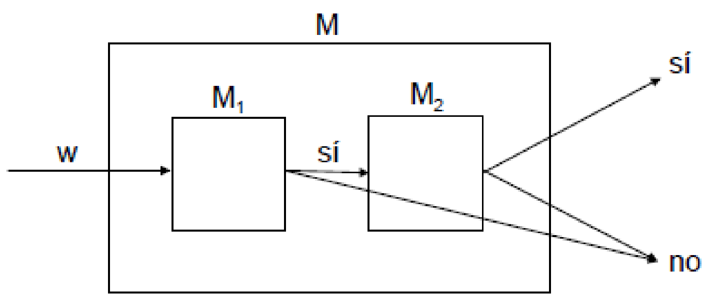
> Si un problema es decidible, entonces, también lo es el problema contrario.

Propiedad 2. Si L1 ∊ R y L2 ∊ R, entonces L1 ∩ L2 ∊ R y L1 ∪ L2 ∊ R.

Es decir, se establece que R es cerrado con respecto a la intersección y la unión.

Idea general.

Construir una MT M que ejecute secuencialmente las MT M_1 y M_2 y acepte sii M_1 y M_2 aceptan.



M_1 decide L_1 (hipótesis).

M_2 decide L_2 (hipótesis).

M decide $L_1 \cap L_2$ (construcción).

> Si dos problemas son decidibles también es decidible el problema común a ambos.

Lenguajes Recursivamente enumerables

Propiedad 3. Si L1 ∊ RE y L2 ∊ RE, entonces L1 ∪ L2 ∊ RE.

Es decir, se establece que RE es cerrado con respecto a la intersección y la unión.

1. Ejecuta M_1.
2. Si M_1 rechaza, entonces rechaza (no se detiene si M_1 no se detiene).
3. Ejecuta M_2.
4. Si M_2 acepta, entonces acepta, y si M_2 rechaza entonces rechaza (no se detiene si M_2 no se detiene).

La figura 3.4 muestra la máquina construida.

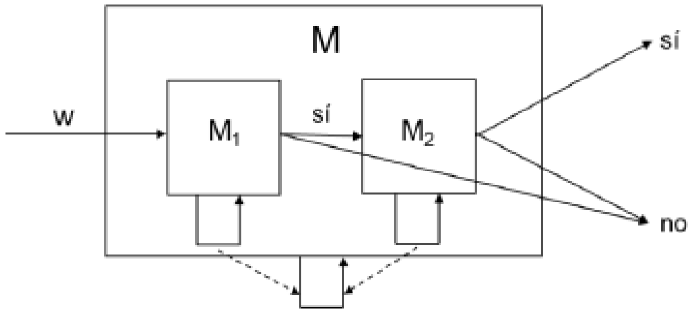


Figura 3.4. La MT M reconoce la intersección de los lenguajes que reconocen las MT M_1 y M_2.

> Revisar filmina 12 y página 46 y 47 del libro

> RE no es cerrado con respecto al complemento, es decir, existen lenguajes RE cuyos complementos no lo son.

CO - RE

Se define al conjunto CO-RE como el conjunto de todos los *complementos* de los lenguajes que están en RE, es decir, `L ∈ RE sii L^c ∈ CO-RE`. Se demuestra entonces que: `R = RE ∩ CO-RE`

Propiedad 4. R = RE ∩ CO-RE.

Es decir, existe una MT M que decide un lenguaje L (lo reconoce y se detiene siempre) sii existen dos MT M1 y M2 que reconocen L y L^c respectivamente (que no necesariamente se detienen)

> Un lenguaje es recursivo sii tanto el lenguaje como su complemento son recursivamente enumerables.

> Si un problema y el problema contrario son computables, entonces ambos son decidibles.

Propiedad 5. R se encuentra contenido en RE ∩ CO-RE.

Jerarquía de la computabilidad

Se define *L* como una partición de cuatro conjuntos, que ordenados por su grado de dificultad:

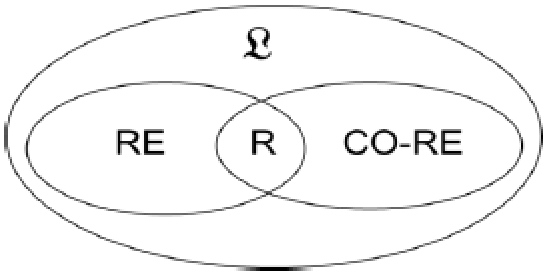


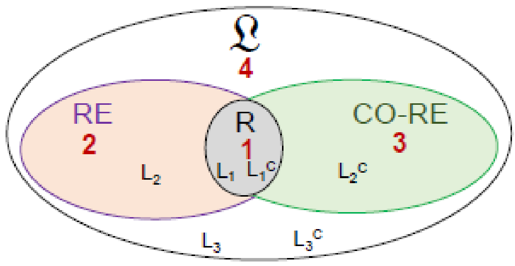
Figura 3.7. Versión definitiva de la jerarquía de la computabilidad.

- **R**: existe una MT que siempre se detiene. Si L está en R entonces L^c está en R.
- **RE-R**: son los lenguajes enumerables no recursivos que son reconocidos por una MT que en al menos un caso (negativo) no se detienen. Si L está en RE-R entonces L^c está en CO-RE-R
- **CO-RE-R**: lenguajes no aceptados por MT, con complementos aceptados por MT que no siempre paran. Si un lenguaje L está en CO-RE-R entonces su complemento L^c está en RE-R.
- ***L*-(RE** ∪ **CO-RE)**: No existe una MT que los reconozca, ni tampoco una MT que reconozca sus complementos. Si un lenguaje L está en *L*-(RE ∪ CO-RE) entones lo está su complemento L^c.

Las cuatro regiones de la jerarquía de la computabilidad

Región 1 (lenguajes aceptados por MT que siempre paran)
Conjunto R.
Si L está en R, entonces L^C está en R

Región 2 (lenguajes aceptados por MT que no siempre paran)
Conjunto RE – R.
Si L está en RE, entonces L^C está en CO-RE



Las cuatro regiones de la jerarquía, con grado de dificultad creciente.

Región 3 (lenguajes no aceptados por MT, con complementos aceptados por MT que no siempre paran)
Conjunto CO-RE – R.
Si L está en CO-RE, entonces L^C está en RE

Región 4 (lenguajes no aceptados por MT, con complementos no aceptados por MT)
Conjunto $\mathfrak{L}$ – (RE ∪ CO-RE).
Si L está en $\mathfrak{L}$ – (RE ∪ CO-RE), entonces L^C está en $\mathfrak{L}$ – (RE ∪ CO-RE)