

Complejidad Computacional

Tema 2: Decidibilidad y Reducibilidad

Modelos abstractos de computación

Hilbert:

- ✿ Todo problema matemático bien definido debe ser necesariamente susceptible de un planteamiento exacto, ya sea en forma de una respuesta real a la pregunta planteada o debido a la constatación de la imposibilidad de resolverlo...
- ✿ Descripciones de algoritmo:
 - ✿ Máquinas computadoras abstractas
 - ✿ Construcciones formales de procedimientos de cómputo
 - ✿ Construcciones formales productoras de clases de funciones

Modelos abstractos de computación

📖 Cuestiones planteadas: ¿Las matemáticas...

- ✿ son consistentes?

- ✿ son decidibles?

- ✿ son completas?

- ✿ Se demostró que era imposible

 - ✿ Church


 - ✿ Turing

 - ✿ Gödel

 - ✿ Kleene

 - ✿ Post

Teoría Complejidad Computacional

 Recursos requeridos para resolver un problema teóricamente computables.

 Clasificación de los problemas

- Problemas con solución (computables)

 - Complejidad: cantidad de recursos necesarios

 - Algoritmos efectivos o tratables

- Problemas sin solución (no computables)

Problemas de decisión

Problemas de decisión, D:

- Una entrada

 - Instancia: Conjunto I_d

- Una pregunta

 - Respuestas sí o no

- Instancias positivas: Conjunto I_d^+

- $D = (I_d, I_d^+)$, donde $I_d^+ \subseteq I_d$

Ejemplo:

- Problema de accesibilidad en un grafo

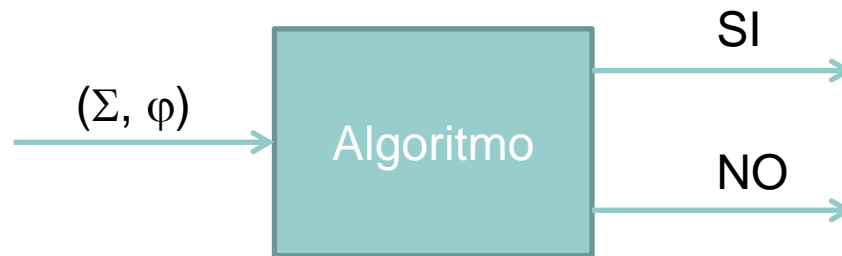
Problemas de decisión

Ejemplo:

- ✿ Máquina de Turing como reconocedora de un lenguaje
- ✿ ¿ $w \in L(M)$?
- ✿ Lenguaje reconocido por una Máquina de Turing
 - ✿ Lenguaje recursivamente enumerable
 - ✿ Problema computable / aceptable
- ✿ Si se garantiza que la Máquina de Turing siempre para
 - ✿ Lenguaje recursivo
 - ✿ Problema decidable

Problemas de decisión

📖 Problemas de decisión, D:



✿ Entrada

👉 Par (Σ, φ)

✿ Respuesta

👉 SI: Si se cumple $\Sigma \vdash^* \varphi$

✿
$$L = \{ \langle \Sigma, \varphi \rangle \mid \Sigma \subseteq L(P), \varphi \in L(P), \Sigma \vdash^* \varphi \}$$

Codificación de una MT

 Una Máquina de Turing puede codificarse como una secuencia binaria finita

- ✿ Simplificación:

 - Sólo un estado inicial y sólo un estado de aceptación

- ✿ A cada estado, símbolo del alfabeto y movimiento se le asigna una codificación

- ✿ Una transición se codifica a partir de los estados, símbolos y movimientos que la forman

- $\delta(q_i, s_j) = (q_k, s_l, m_m) \Rightarrow 0^i 10^j 10^k 10^l 10^m$

- ✿ Una máquina de Turing se codifica escribiendo consecutivamente las transiciones

- $111T_111T_211T_311 \cdots 11T_n111$

Problemas no decidibles

Problema decidable

- ✿ Máquina de Turing que:

 - Acepte cualquier $w \in L$

 - Rechace cualquier $w \notin L$

Problemas no decidibles

Identificar los límites de la computabilidad

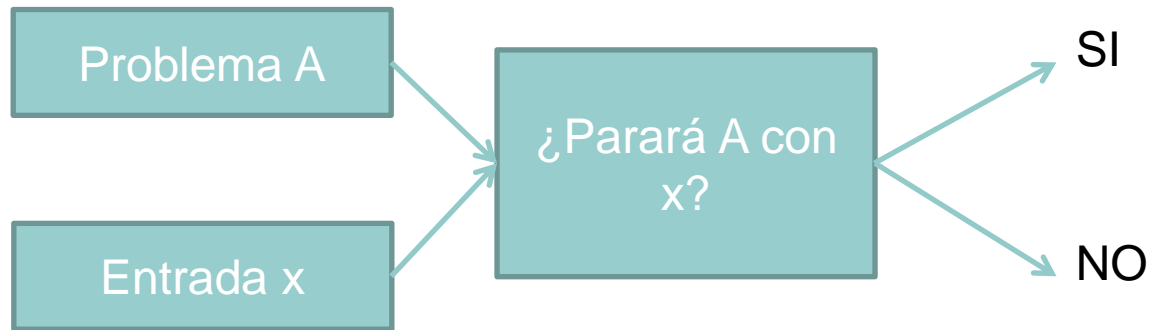
- ✿ Técnica de la diagonalización (Cantor)

 - Conjuntos numerables / incontables

Problema de la parada

📖 Problema de la parada:

- ✿ Dado un algoritmo A y un valor de entrada x, ¿parará A con x?



- ✿ Turing demostró ninguna máquina de Turing lo puede resolver

Problema de la parada

Reducción al absurdo:


```
int Halts(char * P, char * I) {  
    // P : código fuente  y  I : datos  
  
    // 1. Compilar el programa  
    // 2. Determinar si P finaliza su ejecución con I  
    return halt;
```

```
int main( ) {  
    leer_programa(I);  
  
    if (Halts(I, I))  
        while (1);  
    else  
        return 1;
```

Lenguaje complementario

 Complementario de un lenguaje: (co-L)

✿ Todas las palabras que no están en L

 co-Turing-reconocible

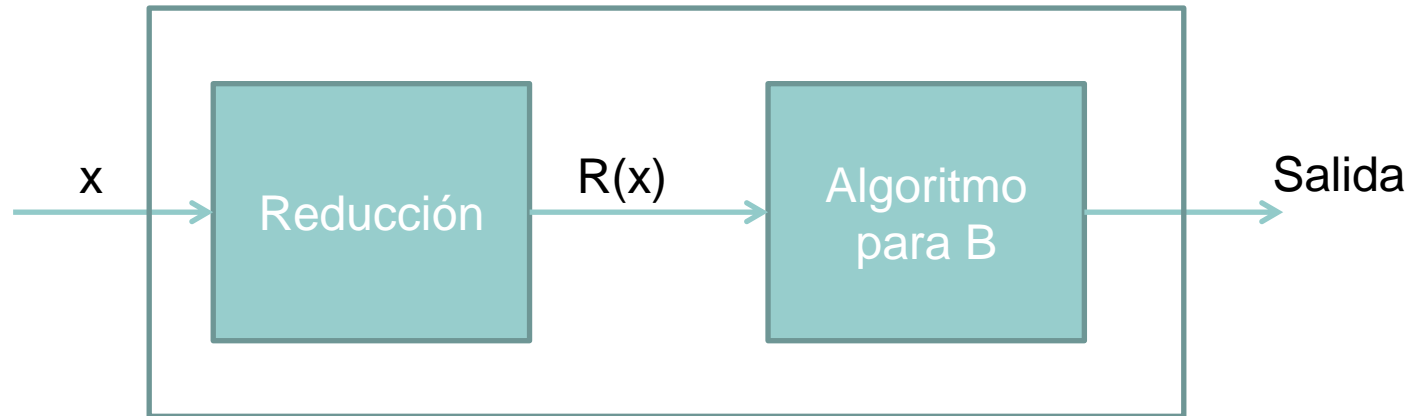
✿ Lenguaje complementario de un lenguaje Turing-reconocible

$L \text{ decidable} \Leftrightarrow L \text{ Turing-reconocible y } co-L \text{ Turing-reconocible}$

Reducibilidad

📖 Técnica de reducibilidad



- ⚙️ Reducir un problema a otro
- ⚙️ Una solución del segundo problema puede ser usada para resolver el primer problema
- ⚙️ Comprobar si un problema tiene o no solución efectiva



Reducibilidad



Método:

Reducir el problema A al problema B:

-  Existe una función computable que convierte instancias de A en instancias de B (Reducción)
-  Se puede resolver A con un algoritmo para resolver B

$$A \leq_m B$$

Complejidad


-  El tiempo de la función de transformación afecta al tiempo total
-  Si se puede calcular en tiempo polinómico

$$A \leq_p B$$

Reducibilidad

Formalmente:

✿ Cambiar un problema D por un problema D':

 Construir f del conjunto I_d al conjunto I'_d tal que

$$I \in I_d^+ \Leftrightarrow f(I) \in I_d'^+$$

Reducción de lenguajes

✿ A se reduce a B:

$$A \leq_m B, \text{ si existe } f: \Sigma^* \rightarrow \Sigma^* \\ \text{tal que } \forall w \in \Sigma^*: w \in A \Leftrightarrow f(w) \in B$$

Reducibilidad

📖 Clasificación: A no puede ser más difícil de resolver que B

⚙️ Si $A \leq_m B$ y B es decidable

👉 A es decidable

⚙️ Si $A \leq_m B$ y B es Turing-reconocible

👉 A es Turing-reconocible

⚙️ Si $A \leq_m B$ y A es indecible

👉 B es indecible

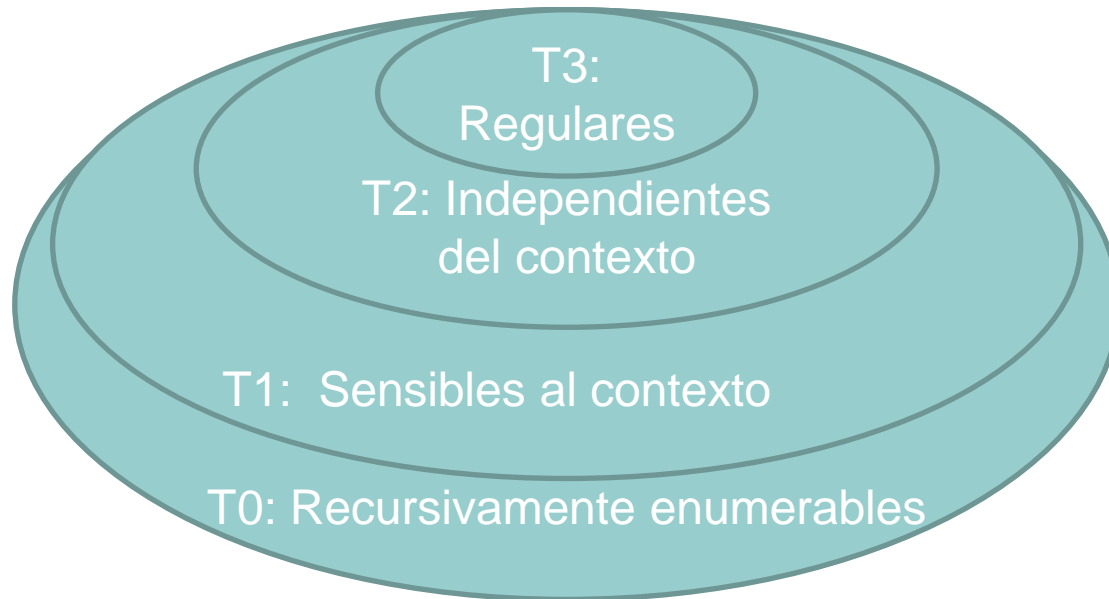
Clasificación de lenguajes

📖 Gramáticas: $G = (V, T, P, S)$

$$L(G) = \{u \in T^* \mid S \xRightarrow{*} u\}$$

📖 Jerarquía de Chomsky:

🌀 Clasificación de gramáticas



Clasificación de lenguajes

Gramáticas equivalentes

- ✿ Problema indecidible

Clasificación de Lenguajes

- ✿ Lenguaje de tipo $i \Leftrightarrow$ Generado por gramática tipo i

Lenguaje Tipo 0 (L0):

- ✿ Lenguaje recursivamente enumerable
- ✿ Lenguaje parcialmente decidable
- ✿ Lenguaje Turing-computable

Complejidad algorítmica

Complejidad:

Notación O:

-  Establecer clases de problemas de la misma dificultad

Máquina de Turing

-  n : longitud de la cadena de entrada

-  Complejidad espacial: número de celdas visitadas

-  Complejidad temporal: número de movimientos de la cabeza Lectura/Escritura

Clasificación de problemas

Clasificación:

- Computables

 - Eficientemente computables

 - Intratables

- No computables

Clases

- P

- NP

Máquina Universal de Turing

Turing:

- ✿ “Se puede demostrar que es posible construir una máquina especial de este tipo que pueda realizar el trabajo de todas las demás. Esta máquina especial puede ser denominada máquina universal”

Máquina Universal de Turing:

- ✿ MT programables que pueden simular cualquier otra MT:

 M_U : Capaz de simular una máquina de Turing M

$$M_U(M, w) = M(w)$$

Máquina Universal de Turing

Máquina Universal de Turing:

- ✿ Programa: Codificación de M y la entrada

- ✿ Máquina de Turing de 3 cintas:

1. $\langle M, w \rangle$
2. Área de trabajo
3. Estado actual de la máquina simulada

- ✿ Proceso

Lenguaje Universal:

$$L_U = \{ \langle M, w \rangle \mid M \text{ acepta } w \}$$

- ✿ L_U recursivamente enumerable, no recursivo

Tesis de Church - Turing

Tesis de Church:

- ✿ La clase de funciones calculables coincide con el de las funciones parcialmente recursivas

Tesis de Turing:

- ✿ La clase de las funciones que pueden ser calculadas mediante un método definido coincide con la clase de las funciones calculables mediante una Máquina de Turing

Tesis de Church - Turing

Equivalencia:

✿ $\{\text{Funciones Recursivas Parciales}\} = \{\text{Funciones Turing-Computables}\}$

Tesis de Church-Turing

- ✿ Todo algoritmo o procedimiento efectivo es Turing-computable
- ✿ Afirmación formalmente indemostrable