Ignacio Méndez Pérez
2023

Resumen Lógica para ciencias de la computación

Sea P un conjunto de variables proposicionales

Definición

Dado P, se define $\mathcal{L}(P)$ como el menor conjunto que satisface

1. $P \subseteq \mathcal{L}(P)$
2. Si $\varphi \in \mathcal{L}(P)$, entonces $(\neg\varphi) \in \mathcal{L}(P)$.
3. Si $\varphi, \psi \in \mathcal{L}(P)$ y $\circ \in \{\wedge,\ \vee,\ \rightarrow,\ \leftrightarrow\}$, entonces $(\varphi \circ \psi) \in \mathcal{L}(P)$.

Cada $\varphi \in \mathcal{L}(P)$ es una **fórmula proposicional**

Ejercicio

Demuestre que para $P = \{p, q, r\}$, se tiene que $\big((p \leftrightarrow (\neg q)) \vee r\big) \in \mathcal{L}(P)$

Desarrollo:

Para demostrar que la fórmula $((p \leftrightarrow (\neg q)) \vee r)$ pertenece al lenguaje $\mathcal{L}(P)$, debemos mostrar que se puede construir utilizando únicamente los símbolos del conjunto de proposiciones $P = \{p, q, r\}$ y los conectivos lógicos permitidos en $\mathcal{L}(P)$.

La fórmula $((p \leftrightarrow (\neg q)) \vee r)$ combina varios conectivos lógicos, por lo que vamos a descomponerla en pasos más sencillos:

1. La fórmula $\neg q$ implica que debemos negar la proposición q. Podemos escribirlo como $\neg q$.

2. La fórmula $(p \leftrightarrow (\neg q))$ implica una equivalencia lógica entre $p\ y\ \neg q$. Podemos expresarlo como $(p \rightarrow (\neg q)) \wedge ((\neg q) \rightarrow p)$.

   - Para $p \rightarrow (\neg q)$, podemos utilizar escribir el condicional $\rightarrow$ como: $\neg p \vee (\neg q)$.
   - Para $(\neg q) \rightarrow p$, podemos utilizar escribir el condicional $\rightarrow$ como $q \vee p$.

   Entonces, $(p \leftrightarrow (\neg q))$ se puede reescribir como $(\neg p \vee (\neg q)) \wedge (q \vee p)$.

3. Por último, tenemos la fórmula $((\neg p \vee (\neg q)) \wedge (q \vee p)) \vee r$, que combina las fórmulas anteriores utilizando el conectivo $\vee$ y el símbolo r.

Por lo tanto, hemos demostrado que la fórmula $((p \leftrightarrow (\neg q)) \vee r)$ se puede construir utilizando únicamente los símbolos del conjunto de proposiciones $P = \{p, q, r\}$ y los conectivos lógicos permitidos en $\mathcal{L}(P)$.Por lo tanto, $((p \leftrightarrow (\neg q)) \vee r)$ pertenece al lenguaje $\mathcal{L}(P)$.

Visualizando $\mathcal{L}(P)$

Esta definición nos permite visualizar $\mathcal{L}(P)$ por niveles, según cuántas reglas inductivas hemos aplicado como máximo.

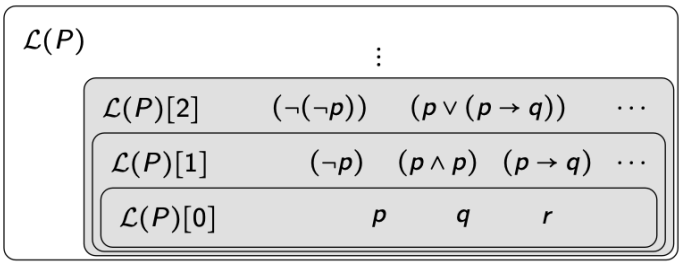
Definimos la **capa** $\mathcal{L}(P)[n]$ de $\mathcal{L}(P)$ para $n \geq 0$ como:

$$\begin{aligned} \mathcal{L}(P)[0] &= P \\ \mathcal{L}(P)[n+1] &= \mathcal{L}(P)[n] \\ &\quad \cup \{(\neg\varphi) \mid \varphi \in \mathcal{L}(P)[n]\} \\ &\quad \cup \{(\varphi \circ \psi) \mid \varphi, \psi \in \mathcal{L}(P)[n]\} \end{aligned}$$

La capa $\mathcal{L}(P)[n]$ contiene las fórmulas que se construyen con **a lo más** n aplicaciones de las reglas inductivas.

Con esto, podemos ver $\mathcal{L}(P)$ como la unión de las diferentes capas

$$\mathcal{L}(P) = \bigcup_{k=0}^{\infty} \mathcal{L}(P)[k]$$



Definición

Dado P y una fórmula $\varphi \in \mathcal{L}(P)$

- Si $\varphi = p$ para $p \in P$, diremos que φ es **atómica**.
- En caso contrario, diremos que φ es **compuesta**.

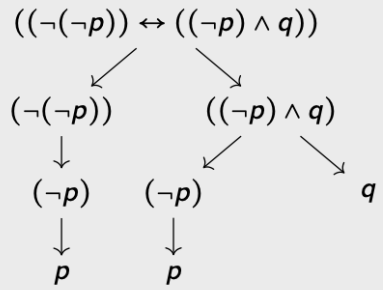
Si φ es compuesta

- Si $\varphi = (\neg\alpha)$ para $\alpha \in \mathcal{L}(P)$, diremos que $\neg$ es un **conectivo primario** de φ y α es una **sub-fórmula inmediata** de φ.
- Si $\varphi = (\alpha \circ \beta)$ para $\alpha, \beta \in \mathcal{L}(P)$, diremos que $\circ$ es un **conectivo primario** de φ y α, β son **sub-fórmulas inmediatas** de φ.

Ignacio Méndez Pérez
2023

Ejercicio

Dado $P = \{p, q\}$, el árbol sintáctico para $((\neg(\neg p)) \leftrightarrow ((\neg p) \wedge q))$ es



La **semántica** se preocupa del **significado** de los símbolos que presentamos en la **sintaxis**.

p	q	$(\neg p)$	$(p \wedge q)$	$(p \vee q)$	$(p \to q)$	$(p \leftrightarrow q)$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Si tomamos $\varphi = \Big(\neg\big((\neg p) \wedge (\neg q)\big)\Big)$

p	q	φ
0	0	0
0	1	1
1	0	1
1	1	1

- Dos fórmulas $\varphi, \psi \in \mathcal{L}(P)$ con **lógicamente equivalentes** si para toda valuación σ, se tiene que $\sigma(\varphi) = \sigma(\psi)$, denotándolo como $\varphi \equiv \psi$.

Daremos por demostradas todas las reglas vistas en Mat. Discretas

Algunas de ellas	Ejemplos
Idempotencia de $\wedge$ y $\vee$	$(p \wedge p) \equiv p$
Doble negación	$(\neg(\neg p)) \equiv p$
Conmutatividad de $\wedge$ y $\vee$	$(p \wedge q) \equiv (q \wedge p)$
Asociatividad de $\wedge$ y $\vee$	$(p \wedge (q \wedge r)) \equiv ((p \wedge q) \wedge r)$
Distributividad de $\wedge$ y $\vee$	$(p \wedge (q \vee r)) \equiv ((p \wedge q) \vee (p \wedge r))$
De Morgan	$(\neg(p \wedge q)) \equiv (\neg p) \vee (\neg q)$
Implicación	$(p \to q) \equiv ((\neg p) \vee q)$

Ignacio Méndez Pérez
2023

Definición
Un **literal** es una variable proposicional p o la negación de una variable proposicional.

Definición
Una fórmula proposicional φ está en **forma normal disyuntiva (DNF)** si es de la forma:

$$\varphi = \bigvee_{i=1}^{m} \left(\bigwedge_{j=1}^{n_j} l_{ij} \right)$$

Donde l_{ij} es un literal.

Definición
Una fórmula proposicional φ está en **forma normal conjuntiva (CNF)** si es de la forma:

$$\varphi = \bigwedge_{i=1}^{m} \left(\bigvee_{j=1}^{n_j} l_{ij} \right)$$

Donde l_{ij} es un literal.

Definición
Sean $\Sigma \subseteq \mathcal{L}(P)$ y $\varphi \in \mathcal{L}(P)$. Decimos que φ es **consecuencia lógica** de Σ, si para toda valuación σ tal que $\sigma(\Sigma) = 1$, se tiene que $\sigma(\varphi) = 1$. En tal caso, lo denotamos por:

$$\Sigma \vDash \varphi$$

- Al conjunto Σ le llamamos **conjunto de premisas** o también **base de conocimiento (KB)**.
- A la fórmula φ tal que $\Sigma \vDash \varphi$ le llamamos **consecuencia**.

Ejemplo: $\{p,\ p \to q\} \vDash q$

p	q	p	$p \to q$	q
0	0	0	1	0
0	1	0	1	1
1	0	1	0	0
1	1	1	1	1

Ignacio Méndez Pérez
2023

Teorema
Sean $\Sigma \subseteq \mathcal{L}(P)$ y $\varphi, \psi \in \mathcal{L}(P)$.

1. Σ es inconsistente si, y solo si, $\Sigma \vDash \varphi$, para cualquier φ.
2. φ es tautología si, y solo si, $\{T\} \vDash \varphi$, para cualquier tautología T. Es decir, es una afirmación que siempre es verdadera: $(p \vee \neg p) \rightarrow Q$.
3. $\Sigma \cup \{\varphi\} \vDash \psi$ si, y solo si, $\Sigma \vDash (\varphi \rightarrow \psi)$.
4. Si $\Sigma \vDash \varphi$ y $\Sigma \cup \{\varphi\} \vDash \psi$, entonces $\Sigma \vDash \psi$ (intermedia).

Teorema

Sea $\Sigma \subseteq \mathcal{L}(P)$ y $\varphi \in \mathcal{L}(P)$. Si $\Sigma \vDash \varphi$, entonces para toda fórmula $\psi \in \mathcal{L}(P)$ se tiene que $\Sigma \cup \{\psi\} \vDash \varphi$.

Al agregar premisas, no perdemos conclusiones que son deducidas en las premisas originales.

Definición
Dado P, sea Σ un conjunto de fórmula en $\mathcal{L}(P)$.

- Si existe $\sigma: \mathcal{L}(P) \rightarrow \{0,1\}$ tal que satisface cada fórmula $\varphi \in \Sigma$, decimos que Σ es **satisfacible**. Notación: $\sigma(\Sigma) = 1$.
- Modelos de un conjunto: valuaciones que lo satisfacen

$$\text{models}(\Sigma) = \{\sigma \mid \sigma(\Sigma) = 1\}$$

- Si $\text{models}(\Sigma) = \emptyset$, decimos que Σ es **insatisfacible**.

Teorema (inconsistencia)

$\Sigma \vDash \varphi$ si, y solo si $\Sigma \cup \{\neg\varphi\}$ es inconsistente.

Verificar **consecuencia** es "lo mismo" que verificar **satisfacibilidad** de un conjunto

$$\{\varphi_1, \dots, \varphi_n\} \text{ es satisfacible} \quad \Leftrightarrow \quad \bigwedge_{i=1}^{n} \varphi_i \text{ es satisfacible}$$

$\Sigma \vDash \varphi$ es equivalente a verificar satisfacibilidad de una **fórmula**

- **Problema SAT** = Ver si φ es satisfacible

Regla se resolución
Dadas cláusulas C_1, C_2 y un literal l

$$\frac{C_1 \vee l \quad C_2 \vee \neg l}{C_1 \vee C_2}$$

Ejemplo:

$$(p \wedge (p \to q)) \to q$$

$$p \to q \equiv \neg p \vee q$$

$$\left.\begin{array}{l} C_1: P \\ C_2: \neg p \vee q \end{array}\right\} \; q$$

Caso particular

$$\frac{\ell \quad \bar{\ell}}{\bot}$$

Definición
Una **demostración por resolución** de C desde Σ es una secuencia de cláusulas $C_1, \dots, C_n$ tal que

- Para cada $i \leq n$
  - $C_i \in \Sigma$ o
  - existen $j, k < i$ tales que C_i se obtiene de C_j, C_k usando la regla de resolución
- $C_n = C$

Lo denotamos por $\Sigma \vdash_R C$

El símbolo $\vdash$, representa la relación de derivación o demostración. Si decimos que $A \vdash B$, significa que la fórmula o conjunto de fórmulas A puede derivar o demostrar la fórmula B utilizando las reglas de inferencia y el método de demostración específico que se está utilizando.

Por otro lado, el símbolo $\vDash$, se utiliza para denotar la relación de consecuencia lógica. Si decimos que $A \vDash B$, significa que la fórmula o conjunto de fórmulas A implica lógicamente la fórmula B. Esto indica que en cualquier interpretación o modelo en el que A sea verdadera, también se cumplirá B.

Ignacio Méndez Pérez
2023

Definición
Un sistema deductivo es **correcto** si para todo Σ y φ se cumple

$$\Sigma \vdash \varphi \quad \Longrightarrow \quad \Sigma \vDash \varphi$$

Definición
Un sistema deductivo es **completo** si para todo Σ y φ se cumple

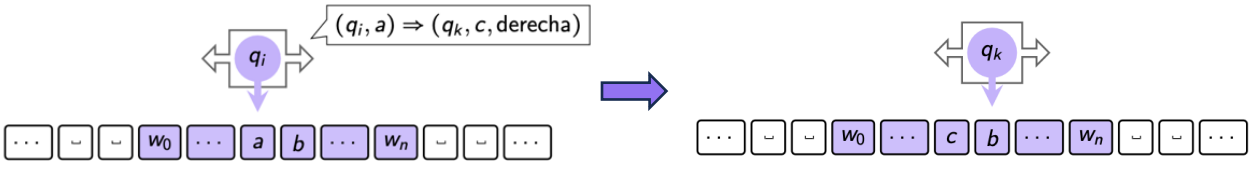
$$\Sigma \vDash \varphi \quad \Longrightarrow \quad \Sigma \vdash \varphi$$

Definición
Sea A un conjunto finito de símbolos al que llamamos **alfabeto**.

- Una **palabra** en A es:
  - $w = \epsilon$, la llamada **palabra vacía**.
  - $w = a$, para $a \in A$
  - $w = a \cdot w'$, para $a \in A$ y alguna palabra w'.
- A^* es el conjunto de todas las palabras con símbolos de A.

Máquina de Turing

1. Inicialización
   - Se comienza con la palabra de input $w = w_0 w_1 \dots w_n$.
   - El input se ubica en una posición **arbitraria** de la cinta.
   - La cabeza lectora se coloca en w_0 con estado inicial q_0
2. Durante su funcionamiento
   - La cabeza **lee** el símbolo a apuntado por ella.
   - Busca la instrucción (q_i, a) que dice que hacer con estado q_i y símbolo leído a.
     - Si existe la instrucción: la ejecuta, escribiendo un símbolo, cambiando su estado y moviéndose.
     - Si no, se **detiene**.



3. Detención
    - Si la máquina se detiene con estado q_j
        - Si q_j es un **estado final**, la máquina **acepta** el input w
        - Si no lo es, la máquina **rechaza** w.

Definición
Una **máquina de Turing determinista** (TM) es una estructura

$$\mathcal{M} = (Q, A, q_0, F, \delta)$$

- Q es un conjunto finito de **estados**.
- A es el alfabeto de **input**.
- q_0 es el estado **inicial**.
- $F \subseteq Q$ es un conjunto finito de **estados finales** (F≠ ∅)
- $\delta: Q \times (A \cup \{\_\}) \rightharpoonup Q \times (A \cup \{\_\}) \times \{\Leftarrow, \Rightarrow\}$ es la **función parcial de transición**.

Definición
Sea $\mathcal{M} = (Q, A, q_0, F, \delta)$ una MT. Definimos el **lenguaje aceptado** por $\mathcal{M}$ como

$$L(\mathcal{M}) = \{w \in A \mid \mathcal{M} \text{ acepta } w\}$$

Definición
Sea $\mathcal{M} = (Q, A, q_0, F, \delta)$ una MT. Una **configuración** de $\mathcal{M}$ es una palabra

$$u \cdot q \cdot v \in A^* \cdot Q \cdot A^*$$

- q es el estado actual
- $u \cdot v$ es el contenido de la cinta
- $|u| + 1$ es la posición actual de la cabeza.

Se define la relación **siguiente paso** si se puede llegar desde una configuración a otra.

$$q_0 000 \overset{\mathcal{M}}{\longmapsto} 0 q_1 00$$

Ignacio Méndez Pérez
2023

Definición
Un lenguaje L es **decidible** o **recursivo** si existe una MT $\mathcal{M}$ tal que

- $L = L(\mathcal{M})$
- $\mathcal{M}$ se detiene en todo input

Si L no es decidible, entonces decimos que es **indecidible**.

Codificación de una máquina de Turing

Resulta natural representar una máquina por una palabra binaria.
Dada una máquina $\mathcal{M} = (Q, A, q_0, F, \delta)$ con

- $Q = \{q_1, \dots, q_n\}$
- $A = \{0,1\}$
- $F = \{q_{i_1}, q_{i_2}, \dots, q_{i_m}\}$ para $1 \leq i_1 < i_2 < \cdots < i_m \leq n$.

Definimos la **codificación** de $\mathcal{M}$ como la palabra

$$cod(\mathcal{M}) = cod(Q) \cdot 00 \cdot cod(\delta) \cdot 00 \cdot cod(F)$$

Codificación de $Q = \{q_1, \dots, q_n\}$

- Representamos cada estado con su número en **unario**.
- Separamos con ceros.

$$cod(Q) \quad = \quad \underline{1}\,0\,\underline{1\,1}\,0 \cdots 0\ \underbrace{1 \cdots 1}_{n \text{ veces}}$$

Codificación de $F = \{q_{i_1}, q_{i_2}, \dots, q_{i_m}\}$ para $1 \leq i_1 < i_2 < \cdots < i_m \leq n$

- Estados en **unario**
- Separamos con ceros.

$$cod(F) \quad = \quad \underbrace{1 \cdots 1}_{i_1 \text{ veces}}\ 0\ \underbrace{1 \cdots 1}_{i_2 \text{ veces}}\ 0 \cdots 0\ \underbrace{1 \cdots 1}_{i_m \text{ veces}}$$

Codificación de δ

- Símbolos legales en la cinta

$$cod(0) = 1 \qquad cod(1) = 11 \qquad cod(\_) = 111$$

- Direcciones

$$cod(\Leftarrow) = 1 \qquad cod(\Longrightarrow) = 11$$

- k-ésima transición, e.g. $t_k := \delta(q_i, 1) = (q_j, \_, \Longrightarrow)$

$$cod(t_k) = \underbrace{1 \cdots 1}_{i \text{ veces}} 0\, \underline{1\,1}\, 0\, \underbrace{1 \cdots 1}_{j \text{ veces}} 0\, \underline{1\,1\,1}\, 0\, \underline{1\,1}$$

- δ con m transiciones

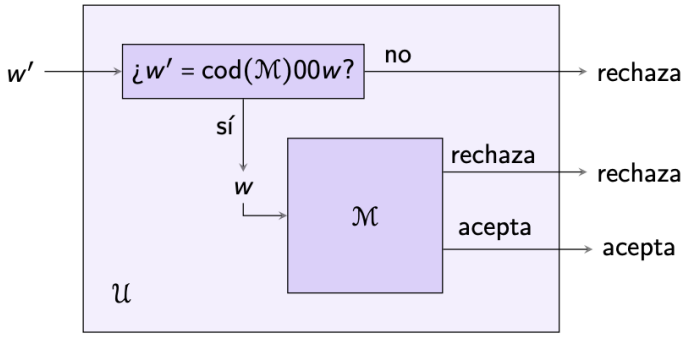
$$cod(\delta) = cod(t_1) \cdot 0 \cdots 0 \cdot cod(t_m)$$

Definición
Una máquina de Turing $\mathcal{U}$ es una **máquina universal** si cumple que

- Si su entrada es de la forma $cod(\mathcal{M})00w$, para $w \in A^*$ y $\mathcal{M}$ máquina con alfabeto A, entonces

$$\mathcal{U} \text{ acepta } cod(\mathcal{M})00w \quad \text{si y solo si} \quad \mathcal{M} \text{ acepta } w$$

- Si su entrada no es de la forma $cod(\mathcal{M})00w$, entonces $\mathcal{U}$ rechaza su entrada.



Problema Halting: Dada una MT $\mathcal{M}$ y una palabra w, se detiene $\mathcal{M}$?
Lenguaje de Halting:

$$H = \{w' \in \{0,1\}^* \mid \text{existen } \mathcal{M} \text{ y } w \in A^* \text{ tales que } w' = cod(\mathcal{M})00w \text{ y } \mathcal{M} \text{ se detiene en } w\}$$

Ignacio Méndez Pérez
2023

Problema diagonal: Dada una máquina de Turing $\mathcal{M}$, se detiene $\mathcal{M}$ en $cod(\mathcal{M})$?
Lenguaje de diagonal:

$$DG = \{w \in \{0,1\}^* \mid existe\ \mathcal{M}\ tal\ que\ w = cod(\mathcal{M}) \\ y\ \mathcal{M}\ se\ detiene\ en\ cod(\mathcal{M})\}$$

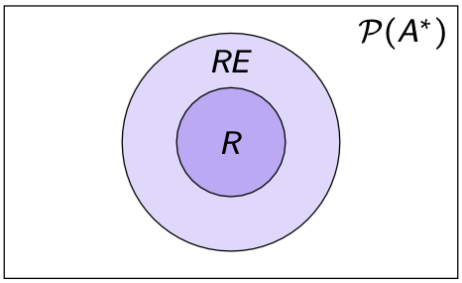
Teorema
El lenguaje DG es indecidible, es decir, no existe una máquina de Turing $\mathcal{M}$ tal que

- $DG = L(\mathcal{M})$
- $\mathcal{M}$ se detiene en todos los inputs.

Este teorema nos ayuda a verificar que el lenguaje H (Halting) también es indecidible.

Relaciones entre RE y R

$$R = \{L \mid L \text{ es decidible}\}$$
$$RE = \{L \mid L \text{ es recursivamente enumerable}\}$$



Teorema
Si un lenguaje es decidible, entonces es recursivamente enumerable.

Teorema
Si un lenguaje L y su complemento $\bar{L}$ son RE, entonces L es decidible.

Notación
Dada una MT determinista $\mathcal{M}$ con alfabeto A que **se detiene en toda entrada**, llamamos:

- **Paso de $\mathcal{M}$ en w** a un par (C_i, C_{i+1}) en una ejecución de $\mathcal{M}$ sobre el input w. Donde C_i es la configuración i.
- **Tiempo de $\mathcal{M}$ en w** al número de pasos ejecutados por $\mathcal{M}$ con entrada w. Lo denotamos por:

$$tiempo_{\mathcal{M}}(w) = |\{(C_i, C_{i+1}) \mid (C_i, C_{i+1}) \text{ es un paso de } \mathcal{M}\ en\ w\}|$$

Ignacio Méndez Pérez
2023

Definición

Dados lenguajes L_1, L_2 con alfabetos A, decimos que L_1 es **reducible en tiempo polinomial** a L_2 si existe una **función computable en tiempo polinomial** $f: A^* \to A^*$ tal que para todo $w \in A^*$

$$w \in L_1 \quad \text{sí y solo si} \quad f(w) \in L_2$$

En tal caso, llamamos a f una **reducción polinomial** de L_1 a L_2 y denotamos

$$L_1 \leq_p L_2$$

Definición (hardness)

Dada una clase de complejidad $\mathcal{C}$ que contiene a P, decimos que un lenguaje L es **$\mathcal{C}$-hard** si

$$\forall L' \in \mathcal{C}. \quad L' \leq_p L$$

Definición (completitud)

Dada una clase de complejidad $\mathcal{C}$ que contiene a P, decimos que un lenguaje L es **$\mathcal{C}$-completo** si

- $L \in \mathcal{C}$
- L es $\mathcal{C}$-hard

Definición

Una **máquina de Turing NO determinista** (NTM) es una estructura

$$\mathcal{M} = (Q, A, q_0, F, \Delta)$$

- Q es un conjunto finito de **estados**.
- A es el alfabeto de **input**.
- q_0 es el estado **inicial**.
- $F \subseteq Q$ es un conjunto finito de **estados finales** (F≠ ∅)
- $\Delta: Q \times (A \cup \{\_\}) \times Q \times (A \cup \{\_\}) \times \{\Leftarrow, \Rightarrow\}$ es la **relación de transición** y en donde ($\Delta \neq \emptyset$).

Ignacio Méndez Pérez
2023

Teorema
Para toda MT $\mathcal{M}$ existe una NTM $\mathcal{N}$ tal que $L(\mathcal{M}) = L(\mathcal{N})$

Teorema
Para toda NMT $\mathcal{N}$ existe una TM $\mathcal{M}$ tal que $L(\mathcal{N}) = L(\mathcal{M})$

El no determinismo NO entrega más poder a las máquinas

Notación
Dada una NMT determinista $\mathcal{M}$ con alfabeto A, llamamos:

- **Paso de $\mathcal{M}$ en w** a un par (C_i, C_{i+1}) en una ejecución de $\mathcal{M}$ sobre el input w. Donde C_i es la configuración i.
- **Tiempo de $\mathcal{M}$ en w** al número de pasos de la **ejecución más corta** de $\mathcal{M}$ que acepta w. Lo denotamos por:

$$tiempo^*_{\mathcal{M}}(w) = |\{(C_i, C_{i+1}) \mid (C_i, C_{i+1}) \text{ es un paso de } \mathcal{M} \text{ en } w\}|$$

Teorema
Sea L un lenguaje. $L \in NP$ sí y solo si, existe una máquina determinista $\mathcal{M}$ que funciona en tiempo polinomial y un polinomio $p(\cdot)$ tales que

$$L = \{ x \mid existe\ y\ tal\ que\ |y| = p(x) \text{ y } \mathcal{M} \text{ acepta } (x, y)\}$$

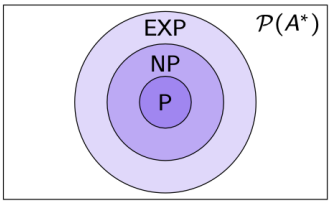
En otras palabras, un lenguaje L se considera perteneciente a la clase NP si existe un algoritmo determinista que pueda verificar si una instancia dada pertenece al lenguaje L en tiempo polinomial.

Ejemplos:

1. $SAT = \{\varphi \mid \varphi \text{ es satisfacible}\}$. Dado una fórmula $\varphi \in SAT$,
   - Testigo σ valuación de largo polinomial respecto a $|\varphi|$
   - Verificador $\mathcal{M}$ que evalúa $\sigma(\varphi)$ en tiempo polinomial.
2. $3COL = \{G \mid G \text{ grafo no dirigido 3 coloreable}\}$. Dado un grafo $G \in 3COL$,
   - Testigo 3-coloración para los vértices.
   - Verificador $\mathcal{M}$ que revisa cada arista (cantidad de aristas polinomial con respecto a $|G|$).

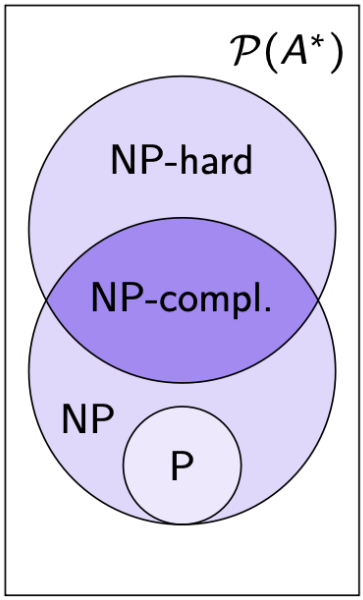
Ignacio Méndez Pérez
2023

Teorema
Las clases P, NP y EXP cumplen que $P \subseteq NP \subseteq EXP$.



– PARENTESIS –

- **NP**: es la clase de problemas que pueden ser verificados en tiempo polinomial por una máquina de Turing no determinista. Esto significa que, dada una solución candidata, se puede verificar su validez en tiempo polinómico. Sin embargo, no implica necesariamente que estos problemas puedan ser resueltos de manera eficiente.



- **NP-Completo**: Los problemas NP-Completo son una subclase de NP que son, en cierto sentido, los más difíciles dentro de NP. Un problema se considera NP-Completo si cumple dos condiciones:
  a) Pertenece a NP
  b) Cualquier problema en NP puede reducirse a él en tiempo polinomial. En esencia, los problemas NP-Completo son aquellos para los cuales, si se encuentra una solución eficiente, se obtendría una solución eficiente para todos los problemas en NP. El problema 3SAT es un ejemplo clásico de un problema NP-Completo.

- **NP Hard**: Es una clase de problemas que son al menos tan difíciles como los problemas NP-Completo, pero no necesariamente están en NP. Un problema NP Hard no necesita ser verificable en tiempo polinomial, pero cualquier problema en NP puede reducirse a él en tiempo polinomial. En resumen, los problemas NP Hard son tan difíciles como los problemas NP-Completo, pero no necesariamente son verificables en tiempo polinomial.

Ignacio Méndez Pérez
2023

Teorema (Cook-Levin)
El problema SAT es NP-completo.

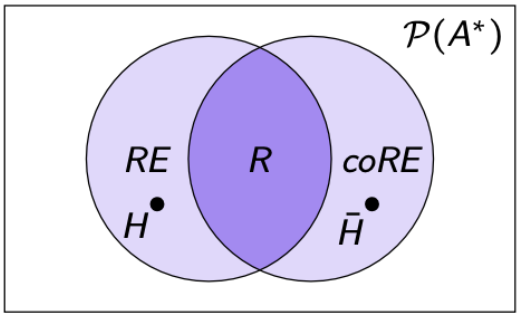
Para probar que L es NP-Completo:

1. Demostrar que $L \in NP$
2. Demostrar que existe una reducción polinomial desde un **problema NP-completo conocido** a L.

Lenguajes **RE**: aceptados por alguna máquina

Lenguajes **coRE**: su complemento es aceptado por alguna máquina

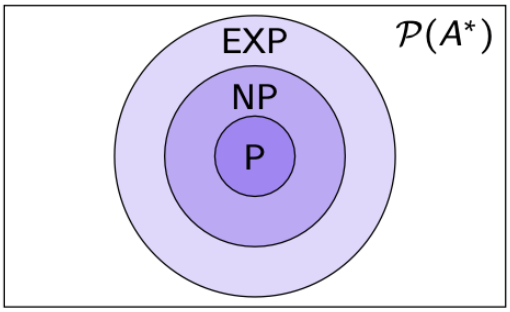
Lenguajes **R** o **decidibles**: aceptados por alguna máquina que siempre se detiene



Lenguajes **EXP**: decididos por $\mathcal{M}$ MT en t. exponencial

Lenguajes **NP**: aceptados por $\mathcal{N}$ NMT en t. polinomial

Lenguajes **P**: decididos por $\mathcal{M}$ MT en t. polinomial



Ignacio Méndez Pérez
2023

Lógica de primer orden (LPO)

Llamaremos **lógica de primer orden (LPO)** a nuestra nueva herramienta.

Notación

Un **vocabulario** $\mathcal{L}$ es la unión de 3 conjuntos dados por

Constantes: $\{c_1, \dots, c_l, \dots\}$
Funciones: $\{f_1, \dots, f_m, \dots\}$
Relaciones: $\{R_1, \dots, R_n, \dots\}$

constantes	:	$\{0, 1\}$
funciones	:	$\{s, +, \cdot\}$
relaciones	:	$\{<\}$

Dada una función f o relación R del vocabulario $\mathcal{L}$, diremos que su **aridad** es el número de argumentos que posee.

- f puede tener aridad mayor a 0.
- R puede tener aridad mayor o igual a 0.

Diremos que los elementos de $\mathcal{L}$ son los **símbolos del vocabulario**

Definición

El conjunto de **$\mathcal{L}$-términos** es el menor conjunto que satisface

- Cada constante $c \in \mathcal{L}$ es un $\mathcal{L}$-término.
- Cada variable x es un $\mathcal{L}$-término.
- Si $t_1, \dots, t_n$ son $\mathcal{L}$-términos y $f \in \mathcal{L}$ es una función n-aria, entonces $f(t_1, \dots, t_n)$ es un $\mathcal{L}$-término.

Ejemplo

Para el vocabulario de los naturales $\mathcal{L}$ que vimos, son $\mathcal{L}$-términos

$$0 \qquad s(s(1)) \qquad s(0) \cdot s(x)$$

Definición

El conjunto de **$\mathcal{L}$-fórmulas** es el menor conjunto que satisface

- Si t_1, t_2 son $\mathcal{L}$-términos, entonces $t_1 = t_2$ es $\mathcal{L}$-fórmula.
- Si $t_1, \dots, t_n$ son $\mathcal{L}$-términos y $R \in \mathcal{L}$ una relación n-aria, entonces $R(t_1, \dots, t_n)$ es $\mathcal{L}$-fórmula.
- Si φ, ψ son $\mathcal{L}$-fórmulas, entonces $(\neg\varphi)$, $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \to \psi)$ y $(\varphi \leftrightarrow \psi)$ son $\mathcal{L}$-fórmulas.

Diremos que las $\mathcal{L}$-fórmulas de la forma $t_1 = t_2$ y $R(t_1, \dots, t_n)$ son **fórmulas atómicas**

Ejemplo

Sea $\mathcal{L} = \{0, 1, s, +, \cdot, <\}$. Las siguientes son $\mathcal{L}$-fórmulas

- $\varphi_1 := \ 1 = s(0)$
- $\varphi_2 := \ \forall x.\ x < s(x)$
- $\varphi_3 := \ \forall x. \exists y.\ x = y + y$
- $\varphi_4 := \ \forall x. \forall y.\ (s(x) = s(y) \to x = y)$