Repaso de algunos conceptos básicos

IIC3810

Problemas de decisión

Alfabeto Σ: Conjunto finito de símbolos.

- Ejemplo: $\Sigma = \{0, 1\}$

Problemas de decisión

Alfabeto Σ: Conjunto finito de símbolos.

- Ejemplo: $\Sigma = \{0, 1\}$

Palabra w: Secuencia finita de símbolos de Σ

- Ejemplo: $w = 01101$

Problemas de decisión

Alfabeto Σ: Conjunto finito de símbolos.

- Ejemplo: $\Sigma = \{0, 1\}$

Palabra w: Secuencia finita de símbolos de Σ

- Ejemplo: $w = 01101$

Σ^*: Conjunto de todas las palabras construidas con símbolos de Σ

Problemas de decisión

Alfabeto Σ: Conjunto finito de símbolos.

- Ejemplo: $\Sigma = \{0, 1\}$

Palabra w: Secuencia finita de símbolos de Σ

- Ejemplo: $w = 01101$

Σ^*: Conjunto de todas las palabras construidas con símbolos de Σ

Lenguaje L: Conjunto de palabras.

- Ejemplo: $L = \{0^n 1^n \mid n \in \mathbb{N}\}$

Problemas de decisión

Problema de decisión asociado a un lenguaje L: Dado $w \in \Sigma^*$, decidir si $w \in L$

Problemas de decisión

Problema de decisión asociado a un lenguaje L: Dado $w \in \Sigma^*$, decidir si $w \in L$

Ejemplo

Recuerde que

$$\text{SAT} \quad = \quad \{\varphi \mid \varphi \text{ es una fórmula en lógica proposicional satisfacible}\}$$

Podemos ver SAT como un problema de decisión.

- $\Sigma = \{x, \_, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\}$

  Algunas palabras de Σ^* tales como $(\neg x\_0)$ y $(x\_31 \wedge x\_27)$ representan fórmulas, mientras que otras tales como $\neg\neg$ y $x_1\neg x_2 \wedge \wedge$ no representan fórmulas

- $\text{SAT} = \{w \in \Sigma^* \mid w \text{ representa una fórmula y } w \text{ es satisfacible}\}$

Complejidad de un problema de decisión

La complejidad de un lenguaje L es la complejidad del problema de decisión asociado a L.

¿Cuándo decimos que L puede ser solucionado eficientemente?

- Cuando existe un algoritmo eficiente que decide L

¿Cuándo decimos que L es un problema difícil?

- Cuando **no existe** un algoritmo eficiente que decide L

Máquinas de Turing: Formalización

Definición

Máquina de Turing (MT) determinista: $(Q, \Sigma, \Gamma, q_0, \delta, F)$

- Q *es un conjunto finito de estados*
- Σ *es un alfabeto tal que* $\vdash, \texttt{B} \notin \Sigma$
- Γ *es un alfabeto tal que* $\Sigma \cup \{\vdash, \texttt{B}\} \subseteq \Gamma$
- $q_0 \in Q$ *es el estado inicial*
- $F \subseteq Q$ *es un conjunto de estados finales*
- δ *es una función parcial:*

$$\delta \quad : \quad Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \square, \to\}$$

δ *es llamada función de transición*

Máquinas de Turing: Funcionamiento

La cinta de la máquina de Turing es infinita hacia la derecha.

- El símbolo $\vdash$ es usado para demarcar la posición 0 de la cinta

Supuestos

- Si $\delta(q, \vdash)$ está definido: $\delta(q, \vdash) = (q', \vdash, X)$, con $X \in \{\rightarrow, \square\}$
- Si $a \in (\Gamma \setminus \{\vdash\})$ y $\delta(q, a)$ está definido: $\delta(q, a) = (q', b, X)$, con $b \in (\Gamma \setminus \{\vdash\})$

Máquinas de Turing: Funcionamiento

Σ es el alfabeto de entrada y Γ es el alfabeto de la cinta.

- Una palabra $w \in \Sigma^*$ de entrada de largo n es colocada en las posiciones 1, ..., n de la cinta
- Las posiciones siguientes $(n+1,\ n+2,\ \ldots)$ contienen el símbolo B

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta
  - Cambia de estado desde q a q'

Máquinas de Turing: Funcionamiento

Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta.

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p

- Si el símbolo en la posición p es a y $\delta(q, a) = (q', b, X)$, entonces:
  - La máquina escribe el símbolo b en la posición p de la cinta
  - Cambia de estado desde q a q'
  - Mueve la cabeza lectora a la posición $p - 1$ si X es $\leftarrow$, y a la posición $p + 1$ si X es $\rightarrow$. Si X es $\square$, entonces la cabeza lectora permanece en la posición p

Máquinas de Turing: Aceptación

Los estados de F son utilizados como estados de aceptación.

- Una palabra w es aceptada por una máquina M si y sólo si la ejecución de M con entrada w se detiene en un estado de F

Máquinas de Turing: Aceptación

Los estados de F son utilizados como estados de aceptación.

- Una palabra w es aceptada por una máquina M si y sólo si la ejecución de M con entrada w se detiene en un estado de F

Definición

Lenguaje aceptado por una máquina de Turing M:

$$L(M) = \{w \in \Sigma^* \mid M \text{ acepta } w\}$$

Máquinas de Turing: Ejercicios

1. Construya una Máquina de Turing que acepte el lenguaje de las palabras $w \in \{0, 1\}^*$ tal que w contiene un número par de símbolos 0.

2. Construya una máquina de Turing que acepte el lenguaje de las palabras $w \in \{0, 1\}^*$ tal que w es un palíndromo.

Complejidad de un algoritmo

Una Máquina de Turing puede no detenerse en alguna entrada.

- Primera noción de algoritmo: MT que se detiene en todas las entradas

¿Cómo se mide el tiempo de ejecución de un algoritmo?

Para una MT con alfabeto Σ:

- Paso de M: Ejecutar una instrucción de la función de transición
- $tiempo_M(w)$: Número de pasos ejecutados por M con entrada $w \in \Sigma^*$

Complejidad de un algoritmo

Definición

El tiempo de funcionamiento de una MT M en el peor caso es definido por la función t_M:

$$t_M(n) \;=\; \text{máx}\{\, tiempo_M(w) \;|\; w \in \Sigma^* \; y \; |w| = n \,\}.$$

Ejercicio

Construya una máquina de Turing que funcione en tiempo $O(n^2)$ y acepte el lenguaje $L = \{w \in \{0,1\}^* \mid w \text{ es un palíndromo}\}$

MT determinista: complejidad de un lenguaje

Un lenguaje L es aceptado por una MT M en tiempo $O(t(n))$ si $L = L(M)$ y $t_M(n)$ es $O(t(n))$

- La definición es idéntica para el caso de $\Omega(t(n))$ y $\Theta(t(n))$

Un ingrediente necesario: No determinismo

Definición

Máquina de Turing no determinista: $(Q, \Sigma, \Gamma, q_0, \delta, F)$

- Q *es un conjunto finito de estados*
- Σ *es un alfabeto finito tal que* $\vdash, \mathtt{B} \not\in \Sigma$
- Γ *es un alfabeto finito tal que* $\Sigma \cup \{\vdash, \mathtt{B}\} \subseteq \Gamma$
- $q_0 \in Q$ *es el estado inicial*
- $F \subseteq Q$ *es un conjunto de estados finales*
- δ *es una relación de transición:*

$$\delta \quad \subseteq \quad Q \times \Gamma \times Q \times \Gamma \times \{\leftarrow, \square, \rightarrow\}$$

Máquinas de Turing no determinista: Funcionamiento

La inicialización es igual que para el caso determinista.

- Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p que contiene un símbolo a.

- Sea $T = \{(q', b, X) \mid (q, a, q', b, X) \in \delta\}$. Si $T \neq \emptyset$, entonces la máquina elije $(q', b, X) \in T$ y:

Máquinas de Turing no determinista: Funcionamiento

La inicialización es igual que para el caso determinista.

- Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p que contiene un símbolo a.

- Sea $T = \{(q', b, X) \mid (q, a, q', b, X) \in \delta\}$. Si $T \neq \emptyset$, entonces la máquina elije $(q', b, X) \in T$ y:
  - escribe el símbolo b en la posición p de la cinta

Máquinas de Turing no determinista: Funcionamiento

La inicialización es igual que para el caso determinista.

- Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p que contiene un símbolo a.

- Sea $T = \{(q', b, X) \mid (q, a, q', b, X) \in \delta\}$. Si $T \neq \emptyset$, entonces la máquina elije $(q', b, X) \in T$ y:
  - escribe el símbolo b en la posición p de la cinta
  - cambia de estado desde q a q'

Máquinas de Turing no determinista: Funcionamiento

La inicialización es igual que para el caso determinista.

- Al comenzar a funcionar, la máquina se encuentra en el estado q_0 y su cabeza lectora está en la posición 1 de la cinta

En cada instante la máquina se encuentra en un estado q y su cabeza lectora está en una posición p que contiene un símbolo a.

- Sea $T = \{(q', b, X) \mid (q, a, q', b, X) \in \delta\}$. Si $T \neq \emptyset$, entonces la máquina elije $(q', b, X) \in T$ y:
  - escribe el símbolo b en la posición p de la cinta
  - cambia de estado desde q a q'
  - mueve la cabeza lectora a la posición $p - 1$ si X es $\leftarrow$, y a la posición $p + 1$ si X es $\rightarrow$. Si X es $\square$, entonces la cabeza lectora permanece en la posición p

Máquinas de Turing no deterministas: Aceptación

Una palabra w es aceptada por una MT no determinista M si y sólo si existe una ejecución de M con entrada w que se detiene en un estado de F.

Definición

Lenguaje aceptado por una MT no determinista M:

$$L(M) \quad = \quad \{w \in \Sigma^* \mid M \text{ acepta } w\}$$

Determinismo vs no determinismo

¿Es posible aceptar más lenguajes con las MTs no deterministas?

Determinismo vs no determinismo

¿Es posible aceptar más lenguajes con las MTs no deterministas?

Teorema

Si un lenguaje L es aceptado por una MT no determinista M_1, entonces L es aceptado por una MT determinista M_2.

Determinismo vs no determinismo

¿Es posible aceptar más lenguajes con las MTs no deterministas?

Teorema

Si un lenguaje L es aceptado por una MT no determinista M_1, entonces L es aceptado por una MT determinista M_2.

Ejercicio

Demuestre el teorema.

- ¿Cuál es la diferencia de complejidad entre M_1 y M_2?

Máquinas de Turing no deterministas: Complejidad

Para una MT no determinista:

- Paso de M: Ejecutar una instrucción de la relación de transición
- $tiempo_M(w)$: Número de pasos de M con entrada w en la ejecución más corta que acepta a w

  Sólo está definido para palabras aceptadas por M

Máquinas de Turing no deterministas: Complejidad

Definición

El tiempo de funcionamiento de una MT no determinista M en el peor caso es definido por la función t_M:

$$t_M(n) \;=\; \text{máx}\left[\{n\} \cup \{\mathit{tiempo}_M(w) \;\mid\; w \in \Sigma^*, \right.$$

$$\left. |w| = n \text{ y } M \text{ acepta a } w\}\right]$$

MT no determinista: complejidad de un lenguaje

Un lenguaje L es aceptado por una MT no determinista M en tiempo $O(t(n))$ si $L = L(M)$ y $t_M(n)$ es $O(t(n))$

Clases de complejidad deterministas

Dado: Alfabeto Σ

DTIME(t): conjunto de todos los lenguajes $L \subseteq \Sigma^*$ que pueden ser aceptados en tiempo $O(t)$ por una MT determinista.

Dos clases fundamentales:

$$\begin{aligned} \mathrm{P} &= \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(n^k) \\ \mathrm{EXPTIME} &= \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(2^{n^k}) \end{aligned}$$

Clases de complejidad deterministas

Dado: Alfabeto Σ

DTIME(t): conjunto de todos los lenguajes $L \subseteq \Sigma^*$ que pueden ser aceptados en tiempo $O(t)$ por una MT determinista.

Dos clases fundamentales:

$$\begin{aligned} \mathrm{P} &= \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(n^k) \\ \mathrm{EXPTIME} &= \bigcup_{k \in \mathbb{N}} \mathrm{DTIME}(2^{n^k}) \end{aligned}$$

P: Conjunto de todos los problemas que pueden ser solucionados "eficientemente".

Clases de complejidad no deterministas

NTIME(t): conjunto de todos los lenguajes $L \subseteq \Sigma^*$ que pueden ser aceptados en tiempo $O(t)$ por una MT no determinista.

Una clase fundamental:

$$\text{NP} \quad = \quad \bigcup_{k \in \mathbb{N}} \text{NTIME}(n^k)$$

¿Cuál es la relación entre las clases anteriores?

Tenemos que:

$$\text{P} \quad \subseteq \quad \text{NP} \quad \subseteq \quad \text{EXPTIME}$$

Además sabemos que:

$$\text{P} \quad \subsetneq \quad \text{EXPTIME}$$

Problemas en una clase de complejidad

Una clase de complejidad contiene un conjunto de problemas de decisión.

Problemas en una clase de complejidad

Una clase de complejidad contiene un conjunto de problemas de decisión.

- ¿Hay alguno de estos problemas que *represente* a la clase?

Problemas en una clase de complejidad

Una clase de complejidad contiene un conjunto de problemas de decisión.

- ¿Hay alguno de estos problemas que *represente* a la clase?

Un caso muy conocido: SAT representa a la clase NP.

Problemas en una clase de complejidad

Una clase de complejidad contiene un conjunto de problemas de decisión.

- ¿Hay alguno de estos problemas que *represente* a la clase?

Un caso muy conocido: SAT representa a la clase NP.

- ¿En qué sentido la representa?

Problemas en una clase de complejidad

Una clase de complejidad contiene un conjunto de problemas de decisión.

- ¿Hay alguno de estos problemas que *represente* a la clase?

Un caso muy conocido: SAT representa a la clase NP.

- ¿En qué sentido la representa?
- ¿Qué sucede si encontramos un algoritmo eficiente para SAT?

Problemas en una clase de complejidad

Una clase de complejidad contiene un conjunto de problemas de decisión.

- ¿Hay alguno de estos problemas que *represente* a la clase?

Un caso muy conocido: SAT representa a la clase NP.

- ¿En qué sentido la representa?
- ¿Qué sucede si encontramos un algoritmo eficiente para SAT?

Vamos a definir las nociones necesarias para estudiar cuando un problema representa a una clase de complejidad.

La noción de reducción: Máquinas que calculan

Definición

Una MT calculadora (MTC): $(Q, \Sigma, \Gamma, q_0, \delta)$

- Q *es un conjunto finito de estados*
- Σ *es un alfabeto finito tal que* $\vdash, \mathtt{B} \notin \Sigma$
- Γ *es un alfabeto finito tal que* $\Sigma \cup \{\vdash, \mathtt{B}\} \subseteq \Gamma$
- $q_0 \in Q$ *es el estado inicial*
- δ *es una función parcial:*

$$\delta \quad : \quad Q \times \Gamma \to Q \times \Gamma \times \{\leftarrow, \square, \rightarrow\} \times (\Sigma \cup \{\mathtt{B}\})$$

δ *es llamada función de transición*

MT calculadora: Funcionamiento

La máquina tiene dos cintas infinitas hacia la derecha.

- La primera cinta es de lectura y la segundo de escritura
- El símbolo $\vdash$ es usado para demarcar la posición 0 de cada cinta

MT calculadora: Funcionamiento

Σ es el alfabeto de entrada y salida, y Γ es el alfabeto de las cintas.

- Una palabra $w \in \Sigma^*$ de entrada de largo n es colocada en las posiciones 1, …, n de la primera cinta
- Las siguientes posiciones $(n+1, n+2, \ldots)$ de la primera cinta contienen el símbolo $\texttt{B}$
- La segunda cinta contienen el símbolo $\texttt{B}$ en las posiciones 1, 2, 3, $\ldots$

La máquina tiene una cabeza lectora por cinta.

- Al comenzar, la máquina se encuentra en el estado q_0, y cada cabeza lectora está en la posición 1 de su cinta

MT calculadora: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i ($i = 1, 2$) se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, a_2) = (q', b_1, X_1, b_2)$, entonces:

MT calculadora: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i ($i = 1, 2$) se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, a_2) = (q', b_1, X_1, b_2)$, entonces:
    - Cambia de estado desde q a q'

MT calculadora: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i ($i = 1, 2$) se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, a_2) = (q', b_1, X_1, b_2)$, entonces:
  - Cambia de estado desde q a q'
  - Escribe el símbolo b_1 en la posición p_1 de la primera cinta

MT calculadora: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i ($i = 1, 2$) se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, a_2) = (q', b_1, X_1, b_2)$, entonces:
    - Cambia de estado desde q a q'
    - Escribe el símbolo b_1 en la posición p_1 de la primera cinta
    - Mueve la cabeza lectora de la primera cinta a la posición $p_1 - 1$ si X_1 es $\leftarrow$, y a la posición $p_1 + 1$ si X_1 es $\rightarrow$. Si X_1 es $\square$, entonces la máquina no mueve la cabeza lectora de la primera cinta

MT calculadora: Funcionamiento

En cada instante la máquina se encuentra en un estado q y su cabeza lectora i ($i = 1, 2$) se encuentra en la posición p_i.

- Si el símbolo en la posición p_i es a_i y $\delta(q, a_1, a_2) = (q', b_1, X_1, b_2)$, entonces:
  - Cambia de estado desde q a q'
  - Escribe el símbolo b_1 en la posición p_1 de la primera cinta
  - Mueve la cabeza lectora de la primera cinta a la posición $p_1 - 1$ si X_1 es $\leftarrow$, y a la posición $p_1 + 1$ si X_1 es $\rightarrow$. Si X_1 es $\square$, entonces la máquina no mueve la cabeza lectora de la primera cinta
  - Si $b_2 \in \Sigma$, entonces escribe el símbolo b_2 en la posición p_2 de la segunda cinta, y mueve la cabeza lectora de esta cinta a la posición $p_2 + 1$. Si $b_2 = \texttt{B}$, entonces no cambia la configuración de la segunda cinta

MTC: Función calculada

El tiempo ocupado por una MTC se define de la misma forma que para una MT (determinista)

MTC: Función calculada

El tiempo ocupado por una MTC se define de la misma forma que para una MT (determinista)

Definición

Una función $f : \Sigma^* \to \Sigma^*$ puede ser calculada en tiempo $O(t(n))$ si existe una MTC M tal que:

- *M para en todas las entradas*
- *t_M es $O(t(n))$*
- *Con entrada $w \in \Sigma^*$: M se detiene en una configuración que tiene en la segunda cinta a $f(w)$, precedido por el símbolo $\vdash$ y seguido por una cadena de símbolos $\mathtt{B}$*

La noción de reducción polinomial

Dados lenguajes L_1 y L_2 con alfabeto Σ

Definición

L_1 es reducible en tiempo polinomial a L_2, denotado como $L_1 \leq_m^p L_2$, si existe una función f computable en tiempo $O(n^k)$ para una constante k tal que para todo $w \in \Sigma^*$:

$$w \in L_1 \text{ si y sólo si } f(w) \in L_2$$

La noción de completitud

Definición (Hardness)

Dada una clase de complejidad $\mathcal{C}$*, un lenguaje* L *es* *hard* *para* $\mathcal{C}$ *si para todo* $L' \in \mathcal{C}$ *existe una reducción polinomial de* L' *a* L*.*

La noción de completitud

Definición (Hardness)

Dada una clase de complejidad $\mathcal{C}$, un lenguaje L es hard para $\mathcal{C}$ si para todo $L' \in \mathcal{C}$ existe una reducción polinomial de L' a L.

Definición (Completitud)

Dada una clase de complejidad $\mathcal{C}$, un lenguaje L es completo para $\mathcal{C}$ si $L \in \mathcal{C}$ y L es hard para $\mathcal{C}$.

La noción de completitud

Definición (Hardness)

Dada una clase de complejidad $\mathcal{C}$, un lenguaje L es hard para $\mathcal{C}$ si para todo $L' \in \mathcal{C}$ existe una reducción polinomial de L' a L.

Definición (Completitud)

Dada una clase de complejidad $\mathcal{C}$, un lenguaje L es completo para $\mathcal{C}$ si $L \in \mathcal{C}$ y L es hard para $\mathcal{C}$.

Notación

Si un lenguaje L es completo para una clase de complejidad $\mathcal{C}$, decimos que L es $\mathcal{C}$-completo.

Completitud en NP

Teorema (Cook-Levin)

SAT es NP-completo.

Reducción polinomial: Una propiedad fundamental

Proposition

Si $L_1 \leq_m^p L_2$ *y* $L_2 \leq_m^p L_3$, *entonces* $L_1 \leq_m^p L_3$.

Otros problemas completos para NP

La propiedad anterior nos dice que si tenemos un problema L completo para una clase $\mathcal{C}$, entonces es más simple encontrar otros problemas $\mathcal{C}$-completos.

- L' es $\mathcal{C}$-completo si $L' \in \mathcal{C}$ y $L \leq_m^p L'$

Vamos a usar esta idea para mostrar otros problemas completos para NP.

Un problema NP-completo: CNF-SAT

Un problema muy útil al hacer reducciones:

$$\text{CNF-SAT} = \{\varphi \mid \varphi \text{ es una conjunción de cláusulas y } \varphi \text{ es satisfacible}\}$$

Teorema

CNF-SAT es NP-completo.

Otro problema NP-completo: 3-CNF-SAT

Notación

Una k-clausula es una clausula con a lo más k literales.

Un problema aun más útil al hacer reducciones:

$$\text{3-CNF-SAT} = \{\varphi \mid \varphi \text{ es una conjunción de } \text{3-cláusulas y } \varphi \text{ es satisfacible}\}$$

Teorema

3-CNF-SAT es NP-completo.

Clases de complejidad: Complemento

Dado un lenguaje L sobre un alfabeto Σ:

$$\overline{L} \;=\; \{w \in \Sigma^* \mid w \notin L\}$$

Clases de complejidad: Complemento

Dado un lenguaje L sobre un alfabeto Σ:

$$\overline{L} \;=\; \{w \in \Sigma^* \mid w \notin L\}$$

Definición

Dada una clase de complejidad $\mathcal{C}$, el conjunto de los complementos de $\mathcal{C}$ se define como: co-$\mathcal{C} \;=\; \{\overline{L} \mid L \in \mathcal{C}\}$

La clase co-NP

Esta es una clase de complejidad muy estudiada.

- ¿Puede identificar algún problema en esta clase?
- ¿Puede identificar algún problema co-NP-completo?
- ¿Es esta clase igual a NP?