Escuela Técnica Superior de Ingeniería Informática

etsinf

DSIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

# Computabilidad y Complejidad

## Tema 8: NP-completitud

# Tema 8: NP-completitud

## Índice


1. Definiciones básicas
2. El Teorema de Cook: SAT es NP-completo
3. Problemas NP-completos básicos y sus reducciones.


## Bibliografía básica recomendada


- Introduction to automata theory, languages and computation. J.E. Hopcroft, J.D. Ullman, R. Motwani. Ed. Addison-Wesley. 2001.
- Theory of Computational Complexity. S. Du, K. Ko. John Wiley & Sons. 2000
- Computers and intractability : A guide to the theory of NP-completeness. M. Garey, D. Johnson. Ed. W.H. Freeman. 1979.

# Definiciones básicas

La clase de complejidad $\mathcal{NP}$ se define mediante máquinas de Turing no deterministas con una complejidad temporal polinómica.

Alternativamente, podemos caracterizarla mediante aquellos problemas que se pueden resolver mediante algoritmos no deterministas con una fase de verificación polinómica:

Algoritmo no determinista $A_{\Pi}$

*Sea $\Pi$ un problema de decisión sujeto a un conjunto de restricciones R.*

*Sea S una estructura de soluciones para el problema $\Pi$*

Fase I: Conjetura

generar_estructura(S)

**Complejidad lineal en S**
**Fase no determinista**

Fase II: Verificación

verificar(S,R)

**Complejidad polinómica en S y R**
**Fase determinista**

# Definiciones básicas

(En lo sucesivo no haremos distinción entre problemas de decisión y lenguajes ya que los primeros se pueden representar como lenguajes a partir de esquemas de codificación "razonables")

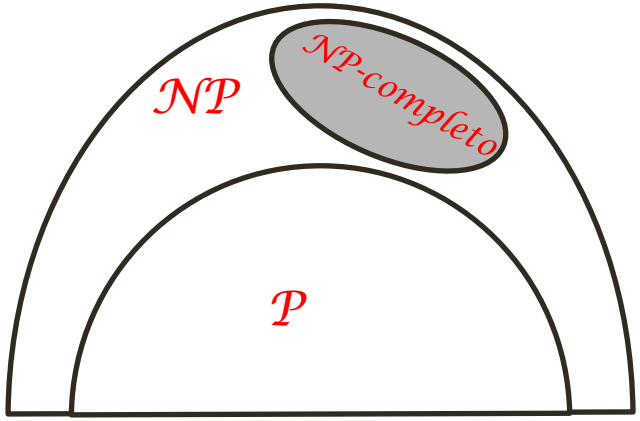
Sean dos lenguajes $L_1$ y $L_2$ definidos sobre el alfabeto $\Sigma$ y sea la función f computable en tiempo polinómico, total y no inyectiva definida sobre cadenas de $\Sigma$. Diremos que $L_1$ es m-reducible polinómicamente a $L_2$ si se cumple el siguiente enunciado

$$L_1 \leq_m^p L_2 \text{ si } (\forall x \in \Sigma^*)\ (x \in L_1 \Leftrightarrow f(x) \in L_2)$$

Un lenguaje $L_1$ pertenece a NP-completo si se cumplen las dos siguientes condiciones:

(a) El lenguaje $L_1$ pertenece a NP

(b) Para todo lenguaje $L_2$ que pertenezca a NP se cumple que $L_2 \leq_m^p L_1$



# El Teorema de Cook

El problema de la satisfacibilidad booleana (SAT)

Sea $U=\{u_1,u_2,\ldots,u_m\}$ un conjunto finito de variables booleanas

Una asignación de valores de verdad es una función $t:U \rightarrow \{T,F\}$
de forma que si $t(u_i)=T$ entonces $u_i$ se interpreta como cierto
si $t(u_i)=F$ entonces $u_i$ se interpreta como falso

Cada variable booleana $u_i$ define dos literales:
$u_i$ que es cierto bajo t sii la variable $u_i$ se interpreta como cierto
$\overline{u}_i$ que es cierto bajo t sii la variable $u_i$ se interpreta como falso

Una cláusula sobre U es un conjunto de literales asociados a las variable de U. Su valor de verdad se interpreta como la disyunción de los valores de verdad interpretados por sus literales (i.e. si alguno de los literales se interpreta como cierto entonces la cláusula también). Diremos que la cláusula es satisfacible si existe una asignación de valores de verdad que la haga cierta.

Un conjunto finito de cláusulas sobre U es satisfacible sii existe una asignación de valores de verdad sobre U que, simultáneamente, haga ciertas todas las cláusulas.

# El Teorema de Cook

El problema de la satisfacibilidad booleana (SAT) se define a partir de un conjunto de variables U y un conjunto de cláusulas C y consiste en establecer si existe o no una asignación de valores de verdad a las variables que satisfaga las cláusulas

Ejemplo

Sea $U = \{x, y, z\}$

Sea $C = \{\{x, \bar{y}, z\}\{x, z\}\{\bar{y}, \bar{z}\}\{\bar{x}, z\}\}$

Podemos reescribir el conjunto C como la siguiente fórmula booleana

$$(x \vee \bar{y} \vee z) \wedge (x \vee z) \wedge (\bar{y} \vee \bar{z}) \wedge (\bar{x} \vee z)$$

La siguiente asignación de valores de verdad satisface el conjunto C

t(x)=T  t(y)=F  T(z)=T

# El Teorema de Cook

El Teorema de Cook (S.A. Cook, 1971) SAT pertenece a NP-Completo

Para demostrar el Teorema de Cook se debe demostrar que SAT pertenece a NP y que todo problema (lenguaje) de NP se puede reducir polinómicamente a SAT.

I) SAT pertenece a NP

Un algoritmo no determinista polinómico para resolver SAT consistiría en realizar una asignación no determinista de valores de verdad para las variables (fase de conjetura) y en la comprobación de la satisfacibilidad de las cláusulas (fase de verificación). La complejidad temporal en la fase de verificación sería

$$O(n \times m)$$

siendo $n$ el número de variables booleanas de U y $m$ el número de cláusulas de C

"The complexity of theorem-proving procedures," *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing* (STOC '71), ACM, New York, NY, USA, 1971, pp. 151-158.

# El Teorema de Cook

El Teorema de Cook (S.A. Cook, 1971) SAT pertenece a NP-Completo

II) $(\forall L \in NP)\ L \leq_m^p SAT$

Tomemos un lenguaje L perteneciente a NP. Existirá una máquina de Turing no determinista M con complejidad temporal polinómica p(n) que acepta a L y contiene una única cinta. La demostración de la reducción de los problemas de NP a SAT se basa en la construcción de un conjunto de cláusulas de forma que se satisfacen sii una cadena de entrada w es aceptada por la máquina M.

$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ $Q=\{q_0,q_1,...,q_r\}$ $\Gamma = \{s_0, s_1, ,..., s_v\}$

Podemos asumir que $F=\{q_1\}$, $B=s_0$ y existe un único estado de rechazo $q_2$ y la cadena de entrada w tiene una longitud igual a n. Además, un movimiento a la derecha lo denotaremos mediante +1 y a la izquierda con -1.

Definición de las variables de U perteneciente a SAT

Q(i,k) : Tras la i-ésima transición el estado de M es $q_k$
H(i,j) : Tras la i-esima transición la cabeza de cinta de M se encuentra en la celda j-ésima
S(i,j,k) : Tras la i-esima transición el contenido de la celta j-ésima es $s_k$

El número de variables será igual a card(Q) + card(H) + card(S) siendo

card(Q) = (p(n)+1)×(r+1) (cláusulas de tipo Q(i,k))
card(H) = (p(n)+1)×2(p(n)+1) (cláusulas de tipo H(i,j) desde la celda –p(n) hasta la celda p(n))
card(S) = (p(n)+1)×(v+1)×2(p(n)+1) (cláusulas de tipo S(i,j,k))

# El Teorema de Cook

El Teorema de Cook (S.A. Cook, 1971) SAT pertenece a NP-Completo

II) $(\forall L \in NP)\ L \leq_m^p SAT$

Definición de las cláusulas de C perteneciente a SAT

Las cláusulas las agruparemos según su semántica y su condición de satisfacibilidad

Grupo I

$$\{Q(i,0), Q(i,1), \ldots, Q(i,r)\} \quad 0 \leq \mathrm{i} \leq \mathrm{p(n)}$$

$$\{\overline{Q(i,j)}, \overline{Q(i,j')}\} \quad 0 \leq i \leq p(n) \quad 0 \leq j < j' \leq r$$

Grupo II

$$\{H(i,-p(n)), H(i,-p(n)+1), \ldots, H(i,p(n)+1\} \quad 0 \leq i \leq p(n)$$

$$\{\overline{H(i,j)}, \overline{H(i,j')}\} \quad 0 \leq i \leq p(n) \quad -p(n) \leq j < j' \leq p(n)+1$$

Grupo III

$$\{S(i,j,0), S(i,j,1), \ldots, S(i,j,v)\} \quad 0 \leq i \leq p(n) \quad -p(n) \leq j \leq p(n)+1$$

$$\{\overline{S(i,j,k)}, \overline{S(i,j,k')}\} \quad 0 \leq i \leq p(n) \quad -p(n) \leq j \leq p(n)+1 \quad 0 \leq k < k' \leq v$$

# El Teorema de Cook

El Teorema de Cook (S.A. Cook, 1971) SAT pertenece a NP-Completo

II) $(\forall L \in NP)\ L \leq_m^p SAT$

Definición de las cláusulas de C perteneciente a SAT

Grupo IV

$\{Q(0,0)\}$

$\{H(0,0)\}$

$\{S(0,0,0)\}$

$\{S(0,1,w_1)\}$

...

$\{S(0,n,w_n)\}$

$\{S(0,n+1,0)\}$

...

$\{S(0,p(n)+1,0)\}$

Grupo V

$\{Q(p(n),1)\}$

Grupo VI-1

$\{\overline{S(i,j,l)}, H(i,j), S(i+1,j,l)\}\quad 0 \leq i < p(n)\quad -\mathrm{p(n)} \leq j \leq p(n)+1\quad 0 \leq l \leq v$

Grupo VI-2

$\{\overline{H(i,j)}, \overline{Q(i,k)}, \overline{S(i,j,l)}, H(i+1,j+\Delta)\}$

$\{\overline{H(i,j)}, \overline{Q(i,k)}, \overline{S(i,j,l)}, Q(i+1,k')\}$

$\{\overline{H(i,j)}, \overline{Q(i,k)}, \overline{S(i,j,l)}, S(i+1,j,l')\}$

$0 \leq i \leq p(n)\quad -p(n) \leq j \leq p(n)+1\quad 0 \leq k \leq r\quad 0 \leq 1 \leq v$

En este caso

si $q_k \in Q - \{q_1, q_2\}$

entonces $\Delta$, $k'$ y $l'$ se definen a partir de $(q_{k'}, s_{l'}, \Delta) \in \delta(q_k, s_l)$

si $q_k \in \{q_1, q_2\}$

entonces $\Delta = 0\quad k' = k\quad l' = l$

# El Teorema de Cook

El Teorema de Cook (S.A. Cook, 1971) SAT pertenece a NP-Completo

- La reducción propuesta por Cook se puede hacer en tiempo polinómico (el número de variables y cláusulas es polinómico a partir de la computación en la máquina de Turing no determinista)

- La reducción propuesta conserva las soluciones: El conjunto de cláusulas es satisfacible sii la cadena de entrada es aceptada por la máquina de Turing

# Problemas NP-completos básicos

- 3-SAT

  Versión del SAT donde las cláusulas tienen exactamente 3 literales

- C3D (Concordancia en 3D)

  Sea el conjunto M formado por tríos de W×X×Y donde W,X e Y son conjuntos disjuntos de la misma cardinalidad (supongamos que sea q).
  Cuestión: ¿ Contiene M un subconjunto de q elementos de forma que ninguno de ellos coincida en ninguna de sus componentes con los demás ?

- CV (Cobertura por vértices)

  Dado un grafo G=(V,E) una cobertura por vértices del grafo G es un subconjunto V' de V de forma que cualquier arista de E tiene unos de sus vértices en V'
  Cuestión: Dado un grafo G=(V,E) y un valor entero positivo k ≤ card(V) ¿Existe una cobertura por vértices de G con k ó menos vértices ?

- CLIQUE

  Dado un grafo G=(V,E) un clique de G es un subconjunto V' de V de forma que todo par de elementos de V' forman una arista de E
  Cuestión: Dado un grafo G=(V,E) y un valor entero positivo k ≤ card(V) ¿Existe un clique de G con k ó más vértices ?

- CH (Circuito Hamiltoniano)

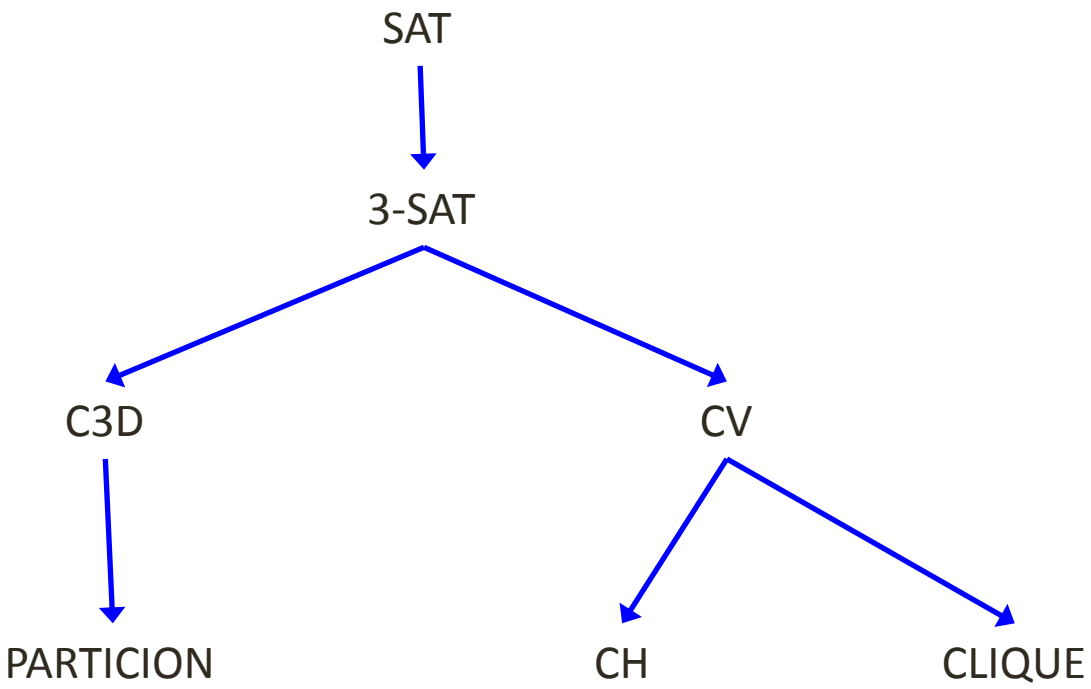
  Cuestión: Dado un grafo G=(V,E) ¿Existe en G un circuito hamiltoniano ?

- PARTICION

  Sea A un conjunto finito y f una función de forma que $(\forall a \in A) f(a) \in Z^+$
  Cuestión: ¿ Existe un subconjunto A' de A que cumpla la siguiente condición ?

$$\sum_{a \in A'} f(a) = \sum_{a \in A - A'} f(a)$$

# Reducciones entre problemas NP-completos básicos



# Reducciones entre problemas NP-completos básicos

Teorema $SAT \leq_m^p 3-SAT$

Para reducir SAT a 3-SAT, tomaremos cualquier instancia de SAT y la transformaremos en una de 3-SAT en un tiempo polinómico y conservando las soluciones. Actuaremos mediante un estudio por casos de las cláusulas de SAT

Caso I

La cláusula sólo tiene un literal {u}. Añadiremos dos variables nuevas $y_1$ e $y_2$ y el siguiente conjunto de cláusulas

$$\{\{u, y_1, y_2\}, \{u, \bar{y}_1, y_2\}, \{u, y_1, \bar{y}_2\}, \{u, \bar{y}_1, \bar{y}_2\}\}$$

Caso II

La cláusula tiene dos literales {u,v}. Añadiremos una variable nueva w y el siguiente conjunto de cláusulas

$$\{\{u, v, w\}, \{u, v, \overline{w}\}\}$$

Caso III

La cláusula tiene tres literales {u,v,w}. Añadiremos esta cláusula sin ninguna modificación.

# Reducciones entre problemas NP-completos básicos

Teorema $SAT \leq_m^p 3-SAT$

Caso IV

La cláusula tiene más de tres literales $\{u_1,u_2,\ldots,u_m\}$ m ≥ 4. Añadiremos un nuevo conjunto de variables $\{y_1,y_2,\ldots,y_{m-3}\}$ y el siguiente conjunto de cláusulas

$$\{\{u_1,u_2,y_1\}\}\cup\{\{\bar{y}_i,u_{i+2},y_{i+1}\}:1\leq i\leq m-4\}\cup\{\{\bar{y}_{m-3},u_{m-1},u_m\}\}$$

Por ejemplo, si la cláusula fuera {u,v,w,x,z} entonces el conjunto de cláusulas nuevas sería

$$\{\{u,v,y_1\},\{\bar{y}_1,w,y_2\},\{\bar{y}_2,x,z\}\}$$

Se puede demostrar que :

1) La transformación propuesta se puede realizar en tiempo polinómico con la cardinalidad de los conjuntos de variables y cláusulas

2) La transformación conserva las soluciones (SAT tiene solución afirmativa sii su transformada 3-SAT la tiene)

# Algunas conclusiones

- A partir del problema SAT se han podido clasificar multitud de problemas NP-completos en una gran diversidad de áreas de conocimiento y sus aplicaciones (por ejemplo: *"Decidir si es posible llegar a un punto cualquiera de un nivel de Super Mario Bros es un problema NP-completo".* Classic Nintendo Games are (NP-)Hard. G. Aloupis, Erik D. Demaine, A. Guo. March, 2012 http://arxiv.org/pdf/1203.1895v1.pdf). En la actualidad el catálogo de problemas NP-completos sigue creciendo cada año de forma considerable.

- Clasificar un problema como NP-completo compromete su resolución práctica (existen instancias que sólo pueden resolverse en tiempo exponencial con el mejor algoritmo determinista secuencial conocido). Esta percepción cambiaría si la conjetura P=NP fuera demostrada.

- Las demostraciones de NP-completitud no se pueden obtener de forma algorítmica. Requieren un grado de conocimiento del problema profundo y, en la mayoría de los casos, se llevan a cabo de forma constructiva. Muchas demostraciones de NP-completitud tienen un nivel de dificultad considerable.