Lógica en la Informática

Josefina Sierra Santibáñez

7 de marzo de 2020

Problemas Decisionales y Clases de Complejidad

- Problema decisional: su solución es una respuesta de tipo SI o NO.
- Determinar si una fórmula es satisfactible, tautología, consecuencia lógica de otra o lógicamente equivalente a otra son problemas decisionales.
- En la lógica proposicional estos problemas son decidibles: existe un algoritmo que termina y nos da una respuesta correcta.
- Coste: *notación asintótica* expresa la rapidez de crecimiento del coste de un algoritmo en función del tamaño de la entrada.
- Orden creciente de coste: logarítmico $O(\log n)$ < lineal $O(n)$ < cuadrático $O(n^2)$ < cúbico $O(n^3)$ < ... < exponencial $O(2^n)$.
- Coste polinómico: proporcional a un polinomio o inferior.
- Clases de complejidad: clases de problemas, no de algoritmos.
- **P**: clase de problemas decidibles mediante un algoritmo determinista Polinómico (p.e. 2-SAT).
- **EXP**: clase de problemas decidibles mediante un algoritmo determinista EXPonencial (p.e. ajedrez generalizado).

La Clase **NP**

- **NP**: clase de problemas decidibles mediante un algoritmo No-determinista Polinómico.
- Un problema pertenece a **NP** si, cuando la respuesta al problema es **sí**, existe un certificado que permite verificar que la respuesta es **sí** en tiempo polinómico (no se requiere para respuestas negativas).
- SAT pertenece a **NP**. Cuando la respuesta es **sí**, el certificado es el modelo de la fórmula. Ya que se puede comprobar en tiempo lineal si una interpretación satisface una fórmula.
- 3-Colorabilidad pertenece a **NP**. Cuando la respuesta es **sí**, el certificado es la asignación de color a cada nodo del grafo. Ya que se puede comprobar en tiempo polinómico si la asignación de colores es diferente para cada par de nodos adyacentes.
- Muchos problemas prácticos de la vida real pertenecen a **NP**. Ejemplos de problemas **NP** son SAT, Hamiltonian circuit, vertex cover, clique, 3-dimensional matching y 3-coloring.

La Clase **NP**

La clase **NP** trata de capturar la noción de verificabilidad en tiempo polinómico.

- Esta noción es distinta de *decidibilidad* en tiempo polinómico.
- Cuando decimos que se puede verificar en tiempo polinómico si una ruta dada es una solución al **problema del viajante**, no tenemos en cuenta el tiempo necesario para encontrar una ruta con las características adecuadas entre el número exponencial de posibles rutas.
- Simplemente afirmamos que dada una ruta para una instancia I del problema del viajante, podemos comprobar en tiempo polinómico si pasa por todas las ciudades de I y si la distancia total recorrida es menor que B (i.e. el límite establecido en I).

Transformación/Reducción Polinómica

Transformación/Reducción polinómica de un problema decisional P_1 en otro problema decisional P_2 es una función f que transforma cada instancia I de P_1 en una instancia $f(I)$ de P_2 tal que

1. Existe un algoritmo determinista polinómico que calcula f.
2. Para toda instancia I de P_1 se tiene que la respuesta de P_1 para I es **sí** $\Leftrightarrow$ la respuesta de P_2 para $f(I)$ es **sí**.

Decimos que P_1 se reduce a P_2 si existe una transformación polinómica de P_1 en P_2, y lo denotamos $P_1 \leq^p P_2$.

También decimos que P_1 es polinómicamente reducible a P_2

Transformación/Reducción Polinómica

Lema 1 Si $P_1 \leq^p P_2$, entonces $P_2 \in \mathbf{P}$ implica $P_1 \in \mathbf{P}$ (equivalentemente, $P_1 \notin \mathbf{P}$ implica $P_2 \notin \mathbf{P}$).

- El lema 1 nos permite interpretar $P_1 \leq^p P_2$ como P_2 es como mínimo tan difícil como P_1.

Lema 2 (transitividad) Si $P_1 \leq^p P_2$ y $P_2 \leq^p P_3$, entonces $P_1 \leq^p P_3$.

Dos problemas decisionales P_1 y P_2 son polinómicamente equivalentes si $P_1 \leq^p P_2$ y $P_2 \leq^p P_1$.

NP-Completitud

Definicion 1 Un problema P_c es **NP-hard** (NP-difícil) si para todo problema P de **NP**, se tiene que $P \leq^p P_c$.

- La definición 1 implica que si existiera un algoritmo polinómico que resolviera un solo problema **NP-hard**, podríamos resolver todos los problemas de **NP** en tiempo polinómico (i.e. $\mathbf{NP} = \mathbf{P}$).
- Se sabe que $\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{EXP}$ y que al menos una de las dos inclusiones es estricta, ya que $\mathbf{P} \subset \mathbf{EXP}$.

Definicion 2 Un problema P_c es **NP-completo** si pertenece a **NP** y es **NP**-hard.

Reducciones Polinómicas y NP-Completitud

Teorema de Cook El problema de determinar la satisfactibilidad de una fórmula de la lógica proposicional (SAT) es NP-completo.

- SAT es NP-completo significa que cualquier problema de la clase **NP** se puede reducir en tiempo polinómico a un problema SAT, es decir, dado un problema Π de **NP** y unos datos D para Π, se puede construir en tiempo polinómico una fórmula F_D que es satisfactible si y sólo si la respuesta de Π para D es **sí**.
- Además, a partir de un certificado del problema SAT para F_D (i.e. un modelo de F_D) suele ser sencillo reconstruir un certificado (i.e. una solución) para el problema Π con datos D.
- El problema **K-SAT con $K \geq 3$ es NP-completo**.
- El problema **2-SAT es lineal**.
- El problema **Horn-SAT es lineal**.

Ejemplo de Reducción Polinómica: 3-Coloreado $\leq^p$ SAT

Sea $G = (V, E)$ un grafo con nodos V y ejes E, $|V| = n$ y $|E| = m$. El problema consiste en determinar si es posible asignar un color entre tres posibles a cada nodo de V de manera que todo par de nodos adyacentes tengan distinto color.

Introducimos $3 \cdot n$ símbolos proposicionales $x_{i,c}$.

Cada $x_{i,c}$ representa la afirmación "se asigna el color c al nodo i", $i = \{1, \ldots, n\}$ y $c = \{1, 2, 3\}$.

Sea S el conjunto de cláusulas formado por

- Para cada nodo $i \in V$, una cláusula de la forma $x_{i,1} \vee x_{i,2} \vee x_{i,3}$. *A cada nodo se le asigna* **al menos un** *color.*

  Cada cláusula de este tipo es una restricción de cardinalidad **ALO** (At Least One).

Ejemplo de Reducción Polinómica: 3-Coloreado $\leq^p$ SAT

- Para cada nodo $i \in V$ y para cada par de colores $1 \leq c_1 < c_2 \leq 3$, una cláusula de la forma $\neg x_{i,c_1} \vee \neg x_{i,c_2}$
  A cada nodo se le asigna **como máximo un** *color.*

  El conjunto de $\binom{3}{2}$ cláusulas de la forma $\neg x_{i,c_1} \vee \neg x_{i,c_2}$ que se añaden para cada nodo i es un ejemplo de restricción de cardinalidad **AMO** (At Most One).

- Para cada eje $(i,j) \in E$, tres cláusulas de la forma $\{\neg x_{i,1} \vee \neg x_{j,1},\ \neg x_{i,2} \vee \neg x_{j,2},\ \neg x_{i,3} \vee \neg x_{j,3}\}$.
  No se puede asignar el mismo color a dos nodos adyacentes.

S es satisfactible si y sólo si existe solución para 3-coloreado de G.

A partir de un modelo de S es fácil construir un coloreado para G.

Ejercicios: construcción de reducciones polinómicas a SAT

- Ejemplo Sudoku de la sección 7 del capítulo 3 de los apuntes.
- Problemas 28, 29 y 31 del capítulo 3 de los apuntes.
- ej. 3 Otoño 2015
- ej. 4 Primavera 2015
- ej. 4 Otoño 2014
- ej. 3 Otoño 2013
- ej. 4 y 5 Primavera 2013
- ej. 3 Primavera 2012
- ej. 4 Otoño 2012
- ej. 3 Otoño 2011

Demostración de que un Problema es NP-Completo

El siguiente lema proporciona un método para demostrar que un problema decisional es **NP**-completo.

Lema 3 Si P_1 y P_2 pertenecen a **NP**, P_1 es **NP-completo** y $P_1 \leq^p P_2$, entonces P_2 es **NP-completo**.

Para demostrar que un problema Π es **NP**-completo es preciso:

1. Demostrar que Π pertenece a **NP**.
2. Seleccionar un problema **NP-completo** conocido P^c.
3. Construir una reducción polinómica f de P^c a Π, i.e. demostrar que $P^c \leq^p \Pi$.

Ejemplo de aplicación del lema 3

Demuestra que 3-SAT es NP-completo utilizando el hecho de que SAT es NP-completo.

Comprobamos las tres condiciones del lema 3

1. 3-SAT pertenece a NP, ya que dada una interpretación I para el lenguaje (i.e. vocabulario) de un conjunto S de cláusulas 3-SAT se puede comprobar en tiempo polinómico si $I \models S$.
2. SAT es NP-completo.
3. La transformación de Tseitin transforma cualquier fórmula de la lógica proposicional F en un conjunto de cláusulas Tseitin(F) equisatisfactible (i.e. la transformación de Tseitin es una reducción de SAT a 3-SAT). Por tanto, SAT $\leq^p$ 3-SAT.

Por el lema 3, podemos afirmar que 3-SAT es NP-completo.

Ejercicios de determinar la complejidad de un problema:

- ej. 3 Primavera 2019
- ej. 4 Primavera 2018 (model counting)
- ej. 3 y 4 Otoño 2018 (MaxSAT)
- ej. 3 Otoño 2016
- ej. 3 y 4 Primavera 2016 (MaxSAT)
- ej. 2 Otoño 2015 (MinOnes)
- ej. 1 y 2 Primavera 2015
- ej. 1c Otoño 2013
- ej. 2 Primavera 2013 (Tseitin y NP-completitud, DNF tautología e insatisfactible)
- ej. 1 Primavera 2012
- ej. 3 y 4 Otoño 2012