

$$2SAT \in P$$

Ejercicio 24 de Sipser

Juan Carlos Alcausa Luque

Universidad de Málaga

Contenido

- 1 Introducción
- 2 Grafo de Implicaciones
- 3 Algoritmo y Demostración
- 4 Análisis de Complejidad

Problema 2SAT

Definición

Una **2cnf-fórmula** es un *AND* de cláusulas, donde cada cláusula es un *OR* de **a lo sumo dos literales**.

Problema

$2SAT = \{ \langle \phi \rangle \mid \phi \text{ es una 2cnf-fórmula satisfacible} \}$

¿Es $2SAT \in P$? Es decir, ¿existe un algoritmo polinómico para decidir si una 2cnf-fórmula es satisfacible?

Ejemplos de Fórmulas 2-CNF

Fórmula 1

$$\phi_1 = (x_1 \vee x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_3)$$

Fórmula 2

$$\phi_2 = (x \vee y) \wedge (y \vee z) \wedge (\neg x \vee \neg z)$$

Fórmula 3

$$\phi_3 = (x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_2)$$

Transformación a Grafo

Equivalencias Lógicas

Para cualquier cláusula $(a \vee b)$:

- $(a \vee b) \equiv (\neg a \rightarrow b)$
- $(a \vee b) \equiv (\neg b \rightarrow a)$

Construcción del Grafo

- Por cada variable x_i , creamos dos nodos: x_i y $\neg x_i$
- Por cada cláusula $(a \vee b)$, añadimos dos aristas dirigidas:
 - $\neg a \rightarrow b$
 - $\neg b \rightarrow a$
- Para cláusulas unitarias (a) , añadimos $\neg a \rightarrow a$

Teorema

Una fórmula 2-CNF ϕ es satisfacible si y solo si no existe ninguna variable x_i tal que x_i y $\neg x_i$ estén en la misma componente fuertemente conexa (SCC) del grafo de implicaciones.

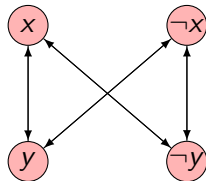
Idea

- Si x_i y $\neg x_i$ están en la misma SCC, entonces $x_i \Rightarrow \neg x_i$ y $\neg x_i \Rightarrow x_i$
- Esto crea una contradicción lógica: no existe ninguna asignación válida
- Un **ciclo de inconsistencia** es un ciclo que contiene tanto x_i como $\neg x_i$

Ejemplo: Fórmula No Satisfacible

Consideremos:

$$\phi_3 = \begin{aligned} &(x \vee y) \wedge \\ &(\neg x \vee y) \wedge \\ &(x \vee \neg y) \wedge \\ &(\neg x \vee \neg y) \end{aligned}$$



Transformando a
implicaciones:

- $(\neg x \rightarrow y), (\neg y \rightarrow x)$
- $(x \rightarrow y), (\neg y \rightarrow \neg x)$
- $(\neg x \rightarrow \neg y), (y \rightarrow x)$
- $(x \rightarrow \neg y), (y \rightarrow \neg x)$

Resultado

¡Todos los nodos están en la misma SCC!
Contradicción: ϕ_3 no es satisfacible.

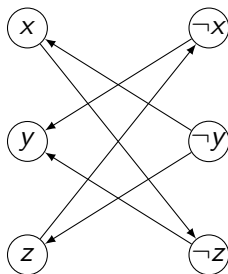
Ejemplo: Fórmula Satisfacible

Consideremos:

$$\phi_2 = (x \vee y) \wedge (y \vee z) \wedge (\neg x \vee \neg z)$$

Transformando a implicaciones:

- $(\neg x \rightarrow y), (\neg y \rightarrow x)$
- $(\neg y \rightarrow z), (\neg z \rightarrow y)$
- $(x \rightarrow \neg z), (z \rightarrow \neg x)$



Resultado

Este grafo no tiene SCCs grandes, solo SCCs triviales (nodos individuales). No hay ciclos que contengan tanto una variable como su negación. Por tanto, la fórmula ϕ_2 es satisfacible. Una asignación válida es: $x = 1, y = 1, z = 0$.

Teorema

ϕ es satisfacible si y solo si no hay un ciclo de inconsistencia en su grafo de implicaciones.

Demostración.

- (\Rightarrow) Si existe un ciclo de inconsistencia, entonces $x_i \Rightarrow \neg x_i$ y $\neg x_i \Rightarrow x_i$ para algún i . Esto crea la equivalencia lógica $x_i \leftrightarrow \neg x_i$, que es una contradicción.
- (\Leftarrow) Si no existe ciclo de inconsistencia, podemos construir una asignación satisfactoria: elegimos una variable no asignada x_i , le asignamos un valor (y a todos sus implicados), eliminamos esos nodos y repetimos hasta asignar todas las variables.



Algoritmo para 2SAT

Algorithm 1 2SAT (vars, cláusulas)

```
1: grafo  $\leftarrow$  construir_grafo_de_implicación(vars, cláusulas)
2: mapa_scc  $\leftarrow$  encontrar_SCCs(grafo)
3: for cada  $x \in$  vars do
4:   if mapa_scc[x] = mapa_scc[-x] then
5:     return falso
6:   end if
7: end for
8: return verdadero
```

Explicación

- Construimos el grafo de implicación
- Encontramos todas las componentes fuertemente conexas (SCCs)
- Verificamos que ninguna variable y su negación estén en la misma SCC

Complejidad Temporal

$$T(V, E) = O(|V| + |E| + |V| + |E| + \frac{|V|}{2} * 2) = O(|V| + |E|)$$

- 1 **Construcción del grafo de implicación:** $|V| + |E|$
- 2 **Encontrar las componentes fuertemente conexas:** $O(|V| + |E|)$
 - La función encontrar_SCCs(grafo) implementa el algoritmo de Kosaraju que tiene una complejidad temporal de $O(|V| + |E|)$
 - Esta es la parte dominante de la complejidad total
- 3 **Verificación de contradicciones:** $|V|$
 - El bucle for itera sobre cada variable, habiendo $\frac{|V|}{2}$ variables
 - La comprobación de si una variable y su negación están en la misma SCC es una operación de tiempo constante.

Conclusión

El algoritmo tiene complejidad temporal $O(|V| + |E|)$, que expresado en términos de variables y cláusulas es $O(n + m)$.

Por tanto, $2SAT \in P$

Conclusiones

- 2SAT es un problema de satisfacibilidad resoluble en tiempo polinómico, lo que implica que $2SAT \in P$
- La clave del enfoque es la transformación del problema a un grafo de implicaciones
- La verificación de satisfacibilidad se reduce a comprobar la ausencia de ciclos de inconsistencia
- El algoritmo tiene una complejidad temporal de $O(|V| + |E|)$