IIC1253 Matemáticas Discretas

Sasha Kozachinskiy

DCC UC

11.08.2025

Hoy...

Hoy...

- Lógica proposicional: problema de satisfacibilidad, modelamiento lógico, sat solvers.

Fórmulas satisfacibles

Definición (Satisfacibilidad)

Una fórmula proposicional ϕ *se llama* **satisfacible** *si existe una asignación de sus variables a 0s y 1s tal que* ϕ *toma valor 1.*

Fórmulas satisfacibles

Definición (Satisfacibilidad)

Una fórmula proposicional ϕ *se llama* **satisfacible** *si existe una asignación de sus variables a 0s y 1s tal que* ϕ *toma valor 1.*

¿Y en términos de equivalencia?

Fórmulas satisfacibles

Definición (Satisfacibilidad)

Una fórmula proposicional ϕ se llama **satisfacible** *si existe una asignación de sus variables a 0s y 1s tal que ϕ toma valor 1.*

¿Y en términos de equivalencia?

Proposición

Una fórmula proposicional ϕ es satisfacible si y sólo si ϕ

Fórmulas satisfacibles

Definición (Satisfacibilidad)

Una fórmula proposicional ϕ *se llama* **satisfacible** *si existe una asignación de sus variables a 0s y 1s tal que* ϕ *toma valor 1.*

¿Y en términos de equivalencia?

Proposición

Una fórmula proposicional ϕ *es satisfacible si y sólo si* ϕ *no es equivalente a 0.*

Problema de satisfacibilidad

Problema de satisfacibilidad

¿Es satisfacible? $(x \to \neg y) \to x$ $= \varphi$

Si, digamos $x = \underline{1}$, $y = \underline{1}$
satisface φ

Problema de satisfacibilidad

¿Es satisfacible? $(x \to \neg y) \to x$

¡No!

¿Es satisfacible? $(x \vee \neg y) \wedge (y \vee \neg z) \wedge (\neg x) \wedge z$

Para cualquier asign. que satisface

tenemos $z = 1$, $x = 0$, $(0 \vee \neg y) = 1$, $(y \vee 0) = 1$

$y = 0 \qquad y = 1$

Problema de satisfacibilidad

¿Es satisfacible? $(x \to \neg y) \to x$

¿Es satisfacible? $(x \vee \neg y) \wedge (y \vee \neg z) \wedge (\neg x) \wedge z$

¿Es satisfacible? $(A \wedge B \wedge \neg A) \vee (D \wedge W \wedge \neg W) \vee (B \wedge \neg B)$

No

Problema de satisfacibilidad

¿Es satisfacible? $(x \to \neg y) \to x$

¿Es satisfacible? $(x \vee \neg y) \wedge (y \vee \neg z) \wedge (\neg x) \wedge z$

¿Es satisfacible? $(A \wedge B \wedge \neg A) \vee (D \wedge W \wedge \neg W) \vee (B \wedge \neg B)$

$x_1 \oplus x_2$

¿Es satisfacible? $\underbrace{(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)}_{x_1 \neq x_2} \wedge \underbrace{(x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)}_{x_2 \neq x_3}$

$x_1 = 0 \qquad x_2 = 1 \qquad x_3 = 0$

$x_1 \oplus x_2 = 1$

x_1	x_2	$x_1 \oplus x_2$
0	0	0
1	0	1
0	1	1
1	1	0

Problema de satisfacibilidad

¿Es satisfacible? $(x \to \neg y) \to x$

¿Es satisfacible? $(x \vee \neg y) \wedge (y \vee \neg z) \wedge (\neg x) \wedge z$

¿Es satisfacible? $(A \wedge B \wedge \neg A) \vee (D \wedge W \wedge \neg W) \vee (B \wedge \neg B)$

¿Es satisfacible? $(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$

Definición

Problema de satisfacibilidad *es el siguiente problema algorítmico: dado una fórmula proposicional* ϕ*, decidir, si es satisfacible or no*

Problema de satisfacibilidad

¿Es satisfacible? $(x \to \neg y) \to x$

¿Es satisfacible? $(x \vee \neg y) \wedge (y \vee \neg z) \wedge (\neg x) \wedge z$

¿Es satisfacible? $(A \wedge B \wedge \neg A) \vee (D \wedge W \wedge \neg W) \vee (B \wedge \neg B)$

¿Es satisfacible? $(x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2) \wedge (x_2 \vee x_3) \wedge (\neg x_2 \vee \neg x_3)$

Definición

Problema de satisfacibilidad *es el siguiente problema algorítmico: dado una fórmula proposicional* ϕ*, decidir, si es satisfacible or no (y si es, encontrar una asignación de las variables a 0s y 1s tal que* ϕ *toma valor 1, es decir,* **asignación que satisface** ϕ*).*

Satisfacibilidad de DNFs y CNFs

Proposición

La fórmula $\phi = \bigvee_{i=1}^{n} \psi_i$ *es satisfacible si y sólo si por lo menos una de las fórmulas* $\psi_1, \ldots, \psi_n$ *es satisfacible.*

Satisfacibilidad de DNFs y CNFs

Proposición

La fórmula $\phi = \bigvee_{i=1}^{n} \psi_i$ *es satisfacible si y sólo si por lo menos una de las fórmulas* $\psi_1, \ldots, \psi_n$ *es satisfacible.*

Corolario

Problema de satisfacibilidad para DNFs se puede resolver en tiempo lineal.

Satisfacibilidad de DNFs y CNFs

Proposición

La fórmula $\phi = \bigvee_{i=1}^{n} \psi_i$ *es satisfacible si y sólo si por lo menos una de las fórmulas* $\psi_1, \ldots, \psi_n$ *es satisfacible.*

Corolario

Problema de satisfacibilidad para DNFs se puede resolver en tiempo lineal.

Proposición

La formula $\phi = \bigwedge_{i=1}^{n} \psi_i$ *es satisfacible si y sólo si todas las fórmulas* $\psi_1, \ldots, \psi_n$ *son satisfacibles.*

Satisfacibilidad de DNFs y CNFs

Proposición

La fórmula $\phi = \bigvee_{i=1}^{n} \psi_i$ *es satisfacible si y sólo si por lo menos una de las fórmulas* $\psi_1, \ldots, \psi_n$ *es satisfacible.*

Corolario

Problema de satisfacibilidad para DNFs se puede resolver en tiempo lineal.

Proposición

~~*La fórmula* $\phi = \bigwedge_{i=1}^{n} \psi_i$ *es satisfacible si y sólo si todas las fórmulas* $\psi_1, \ldots, \psi_n$ *son satisfacibles.*~~ *WRONG!*

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

CNF-SAT

Muchos problemas combinatorios se reducen al CNF-SAT – problema de satisfacibilidad para CNFs.

CNF-SAT

Muchos problemas combinatorios se reducen al CNF-SAT – problema de satisfacibilidad para CNFs.

Conjetura (P$\neq$ NP)

No existe un algoritmo para CNF-SAT que trabaja en tiempo polinomial.

CNF-SAT

Muchos problemas combinatorios se reducen al CNF-SAT – problema de satisfacibilidad para CNFs.

Conjetura (P$\neq$ NP)

No existe un algoritmo para CNF-SAT que trabaja en tiempo polinomial.

`https://www.claymath.org/millennium-problems`

CNF-SAT

Muchos problemas combinatorios se reducen al CNF-SAT – problema de satisfacibilidad para CNFs.

Conjetura (P$\neq$ NP)

No existe un algoritmo para CNF-SAT que trabaja en tiempo polinomial.

https://www.claymath.org/millennium-problems https://en.wikipedia.org/wiki/List_of_NP-complete_problems

Casos simples

Definición

Una CNF se llama 2-CNF si cada cláusula tiene no más que 2 variables.

Casos simples

Definición

Una CNF se llama 2-CNF si cada cláusula tiene no más que 2 variables.

Por ejemplo, $(x \vee \neg y) \wedge (y \vee \neg z) \wedge (\neg x) \wedge z$

Casos simples

Definición

Una CNF se llama 2-CNF si cada cláusula tiene no más que 2 variables.

Por ejemplo, $(x \vee \neg y) \wedge (y \vee \neg z) \wedge (\neg x) \wedge z$

Definición

Una CNF se llama Horn-CNF si cada cláusula tiene no más que 1 variable sin negación.

Por ejemplo, $(\neg a \vee b \vee \neg c) \wedge c \wedge (\neg b \vee \neg a \vee \neg d)$

Casos simples

Definición

Una CNF se llama 2-CNF si cada cláusula tiene no más que 2 variables.

Por ejemplo, $(x \vee \neg y) \wedge (y \vee \neg z) \wedge (\neg x) \wedge z$

Definición

Una CNF se llama Horn-CNF si cada cláusula tiene no más que 1 variable sin negación.

Por ejemplo, $(\neg a \vee b \vee \neg c) \wedge c \wedge (\neg b \vee \neg a \vee \neg d)$

Teorema

2-CNF-SAT y Horn-CNF-SAT se puede resolver en tiempo polinomial.

Sat Solvers

Aunque CNF-SAT es probablemente muy difícil en el peor caso, casos prácticos en muchas instancias se puede resolver muy rápido a través de *Sat Solvers*.

Sat Solvers

Aunque CNF-SAT es probablemente muy difícil en el peor caso, casos prácticos en muchas instancias se puede resolver muy rápido a través de *Sat Solvers*. Ejercicio: resolver CNF-SAT para estas

CNFs en z3-solver:

$$(x \vee y) \wedge (\neg x \vee \neg y) \wedge (y \vee z) \wedge (\neg y \vee \neg z) \wedge (z \vee x) \wedge (\neg z \vee \neg x) \tag{1}$$

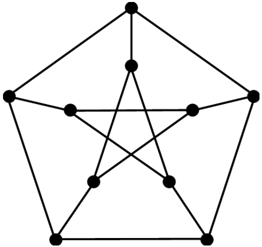
$$\bigwedge_{i=0}^{3} (x_i \vee x_{(i+1)\,\%4}) \wedge (\neg x_i \vee \neg x_{(i+1)\,\%4}) \tag{2}$$

`https://colab.research.google.com/drive/`

`1uP7HAsmhGdMGFN9ANVJTyIWdQsAsXtCv?usp=sharing`

3-COL

¿Se puede colorear vértices de este grafo usando 3 colores tal que vértices adyacentes tengan colores distintos? Resolver en z3-solver



Variables:

$$x_{i,c} = \begin{cases} 1 & \text{vértice } i \text{ tiene color } c, \\ 0 & \text{si no} \end{cases}$$

$$i = 0, ., 9, \;\; c = 0, 1, 2.$$

Cláusulas?

Mini-sudoku

Resolver este mini-sudoku en z3-sover

		4	3
2	3		

Variables: Cláusulas?

¡Gracias!