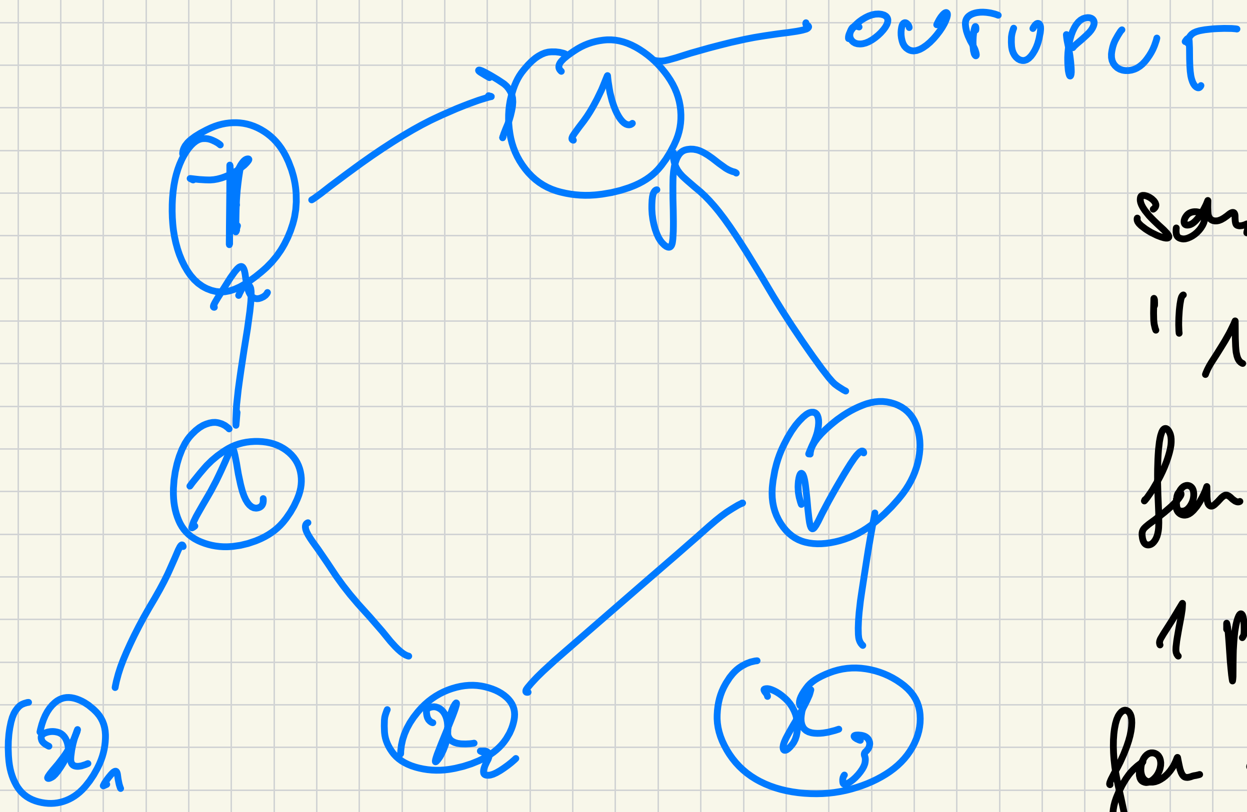
SATISFIABILITY

Vedremo diversi linguaggi riconducibili alla soddisfacibilità di circuiti booleani:

- CIRCUIT-SAT
- FORMULA-SAT
- CNF-SAT
- K-SAT
- 3-SAT

Un circuito Booleano è un grafo diretto, aciclico con input $x_1, \dots, x_n$ e singolo output.



Gli altri nodi sono le porte "$\vee$" "$\wedge$" e "$\neg$".

fan-in: 2 oppure 1 per $\neg$

fan-out: arbitrario

Se fan-out 1 si chiama formula.

Di fatto $C : \{0,1\}^n \to \{0,1\}$. Esistono modi per rappresentare C come codifica $\langle C \rangle$.

Tempo di esecuzione: funzione di n ed m dove $m = \#$ porte.

$|\langle C \rangle| \geq m \geq n$

infatti mi aspetto $O(m \log m)$

CIRCUIT-EVAL $= \{ \langle C, x \rangle : C(x) = 1 \}$

CIRCUIT-EVAL $\in P$, perché una TM può emulare l'esecuzione del circuito.

DEF CIRCUIT-SAT $= \{ \langle C \rangle : \exists\, x \in \{0,1\}^n$ t.c. $C(x) = 1 \}$

FORMULA-SAT, lo stesso ma C è una formula.

Ovviamente CIRCUIT-SAT, FORMULA-SAT $\in$ EXP.

Made with Goodnotes

Perché posso risolverli in tempo $O(2^n \cdot poly(n))$

Non sappiamo invece se questi problemi sono in P. (Spoiler: CIRCUIT-SAT $\in$ P sse P = NP!)

Possiamo anche considerare formule più ristrette.

Ad es.:

- CNF: grosso "$\land$" di clausole dove ogni clausola è "$\lor$" di letterali (variabili)

$$(x_1 \lor x_2 \lor x_3) \land (x_2 \lor \neg x_4) \land \dots\dots$$

$\Rightarrow$ CNF-SAT

DEF $CNF\text{-}SAT = \{ \langle \phi \rangle : \phi \in CNF, \exists x \in \{0,1\}^n$ t.c. $\phi(x) = 1 \}$

$CNF\text{-}SAT \in EXP$, ma $CNF\text{-}SAT \in P$ sse $P = NP$.

Complessità: in termini di n = # variabili ed m = # clausole.

3-SAT ; K-SAT : ciascuna clausola ha $\leq K$ letterali.

3-SAT $\in EXP$; 3SAT $\in P$ sse $P = NP$.

Alcuni fatti interessanti: Miglior algoritmo per 3SAT ho tempo $O(1{,}34^n)$.

$$4\text{-SAT} \in DTIME(1{,}5^n \cdot poly(n))$$

$$5\text{-SAT} \in DTIME(1{,}6^n \cdot poly(n))$$

THM 2SAT $\in P$.

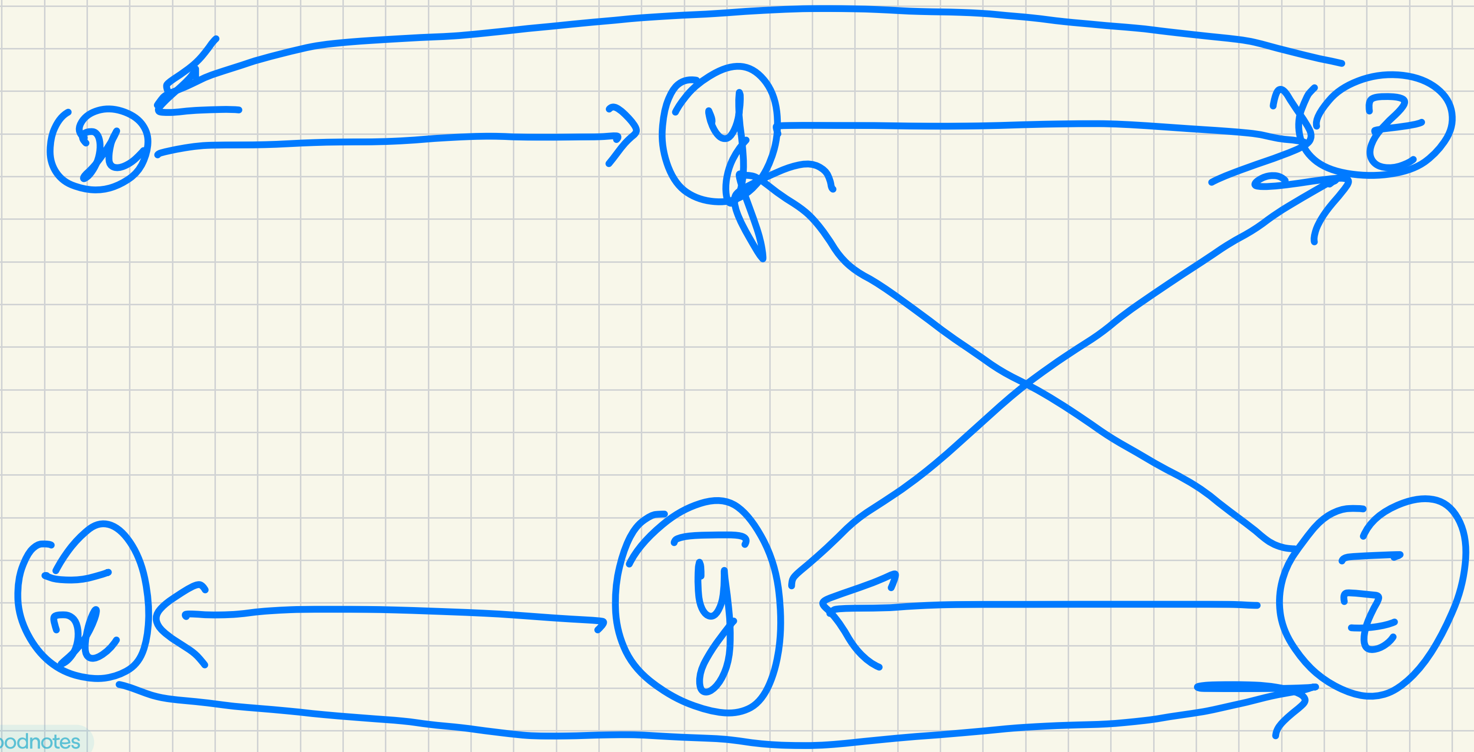
Dim. Daremo un'idea. Associamo alla formula un grafo.

$$(x_1 \vee x_2) \equiv (\bar{x}_1 \rightarrow x_2)$$
$$\equiv (\bar{x}_2 \rightarrow x_1)$$

x_1	x_2	$x_1 \vee x_2$	$\bar{x}_1 \rightarrow x_2$
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Sia $\phi(x_1, \dots, x_n)$ la formula. Per ogni clausola $a \vee b$ in ϕ costruisco il grafo G di vertici $\{x_1, \bar{x}_1, \dots, x_n, \bar{x}_n\}$ e poi aggiungo gli archi $\bar{a}b$ e $\bar{b}a$.

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



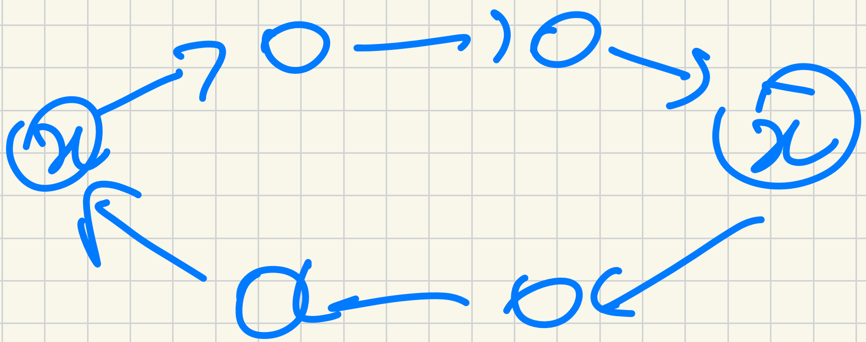
LEMMA ϕ è soddisfacibile sse nessuna componente di G fortemente connessa contiene x e $\bar{x}$.

Dim. Componente fortemente connessa: ogni nodo raggiungibile e partire da altro nodo.

$\Rightarrow$ Sia ϕ soddisfacibile. Allora se ab è arco di G allora se $a = T$, b deve essere T. Questo perché l'arco ab è presente a causa della clausola $\bar{a} \vee b$.

Conseguenza: Per ogni letterale x se c'è un cammino $x \rightsquigarrow \bar{x}$ allora $x = F$. Inoltre se c'è $\bar{x} \rightsquigarrow x$, allora $\bar{x} = F$ ($x = V$)

Per questa ragione se ϕ è soddisfacibile per nessun letterale x ci può essere $x \rightsquigarrow \bar{x}$ e $\bar{x} \rightsquigarrow x$



Ovvero nessuna componente fortemente connessa contiene x e $\bar{x}$.

Perché? Supponiamo $x = T$, se così fosse $\bar{x} = T$ il che è impossibile.
Analogamente per $\bar{x} \rightsquigarrow x$. Se $\bar{x} = T$ onde $x = T$, il che è impossibile.

Made with Goodnotes

($\Leftarrow$) Se nessuna comp. fortemente connessa contiene x, $\bar{x}$, allora ϕ soddisfacibile.
Ordiniamo topologicamente le componenti fortemente connesse $C_1, \dots, C_m$ del grafo G.
L'assegnamento: Per ogni x, si pone $x = T$ sse x appare DOPO di $\bar{x}$. Else, $x = F$.

AFF. Per nessun arco ab di G, il vertice a è assegnato T e b è assegnato F.
Questo implica che ϕ è soddisfacibile. Infatti, se ci fosse una clausola $x \vee y$ t.c. $x = y = F$ avremmo arco $\bar{x}y$ t.c. $\bar{x} = T$ e $y = F$.

Dimostro l'affermazione. Suppongo non sia vero: ab arco tale che $a = T$ e $b = F$.
Suppongo a sia nella componente C_i.
Siccome c'è l'arco ab, c'è la clausola $\bar{a} \vee b$ che genera anche l'arco $\bar{b}\bar{a}$.
Siccome $a = T$ ($\bar{a} = F$), il vertice $\bar{a}$ appare in componente C_j t.c. $j < i$.

$$\left(\begin{array}{lll} x & \text{FALSO}, & x \text{ appare prima } \bar{x} \\ \bar{a} & \text{FALSO}, & \bar{a} \text{ appare prima } a. \end{array} \right)$$

D'altra parte, siccome $b = F$ allora $\bar{b}$ appare in C_K t.c. $K > i > j$

Pertanto l'arco $\bar{b}$ è contraddice ordine topologico delle componenti $\square$

NON DETERMINISMO

Cosa possiamo dire su questi problemi che non sappiamo essere in P (3COL, 3SAT, etc.)
Data una soluzione, possiamo verificarne la correttezza:

- 3COL. La soluzione $c_1, c_2, \dots, c_n \in \{R, B, Y\}$
Basta controllare che $c_i \neq c_j \quad \forall (i,j) \in E$.

- 3SAT. la soluzione è assegnamento
$$x_1, \dots, x_n$$

NP è la classe di linguaggi per cui possiamo associare un verificatore.

DEF Una TM V per L è verificatore se:

- V prende $\langle x, y \rangle$
- abbiamo che $\forall x \in L$ V $x \in L \iff \exists y$ t.c. $V(\langle x, y \rangle) = ACC$.

(i) $x \in L \Rightarrow \exists y$ t.c. $V(\langle x, y \rangle) = ACC$

(ii) $x \notin L \Rightarrow \forall y$ $V(\langle x, y \rangle) = REJ$

Diciamo che V ha tempo polinomiale se

V è eseguibile in tempo $O(|x|^k)$ ovvero polinomiale in $|x|$.
Notare che un tale verificatore implica che $|y| = poly(|x|)$.

DEF NP è l'insieme di linguaggi L che ammettono un verificatore a tempo polinomiale.