Il problema SAT, P vs NP, ed alcuni risvolti pratici

Agostino Dovier

Dept of Mathematics, Computer Science, and Physics
University of Udine
Udine (Italy)

UDINE, 14 Giugno 2018

SAT

- Dato un insieme di variabili Booleane $V = \{X_1, \ldots, X_n\}$
- e una formula proposizionale in CNF costruita su V:

$$\Phi = (\ell_1^1 \vee \cdots \vee \ell_{n_1}^1) \wedge \cdots \wedge (\ell_1^k \vee \cdots \vee \ell_{n_k}^k)$$

  dove ogni ℓ_j^i è una variable X_p o la sua negazione $\neg X_p$
- Il problema è quello di stabilire l'esistenza di un assegnamento di valori vero/falso (1/0) alle variabili in grado di rendere vera la formula Φ
- Ricordiamo che:
  $\neg 0 = 1, \neg 1 = 0$
  $0 \vee 0 = 0, 0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$
  $0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0, 1 \wedge 1 = 1$

SAT
Esempio

$$(X_1 \vee \neg X_2) \qquad (\neg X_1 \vee X_2 \vee \neg X_3) \qquad (\neg X_1 \vee \neg X_2 \vee X_3) \qquad (\neg X_1 \vee \neg X_2 \vee \neg X_3)$$

1. Data una possibile soluzione, verificarla è facile.
2. Le possibili soluzioni sono "tante" $(2^{|\mathbf{V}|})$.

SAT
ESEMPIO

$(X_1 \vee \neg X_2)$ $(\neg X_1 \vee X_2 \vee \neg X_3)$ $(\neg X_1 \vee \neg X_2 \vee X_3)$ $(\neg X_1 \vee \neg X_2 \vee \neg X_3)$

$X_1 = 1$

1. Data una possibile soluzione, verificarla è facile.
2. Le possibili soluzioni sono "tante" ($2^{|\mathbf{V}|}$).

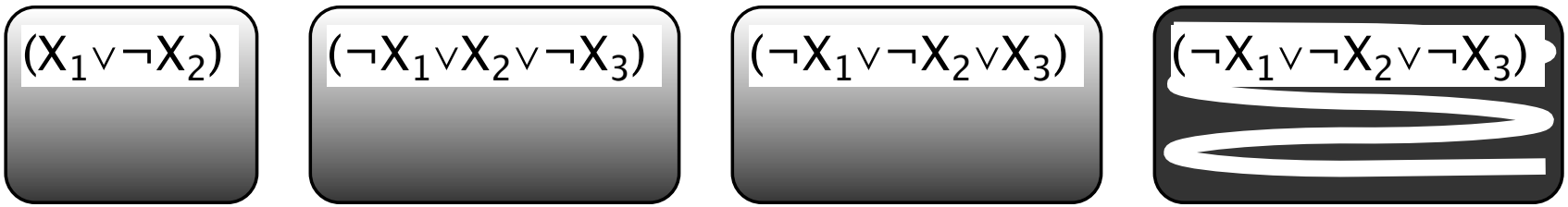
SAT
Esempio

$(X_1 \vee \neg X_2)$ $(\neg X_1 \vee X_2 \vee \neg X_3)$ $(\neg X_1 \vee \neg X_2 \vee X_3)$ $(\neg X_1 \vee \neg X_2 \vee \neg X_3)$

$X_1=1$ $X_2=1$

1. Data una possibile soluzione, verificarla è facile.
2. Le possibili soluzioni sono "tante" ($2^{|\mathbf{V}|}$).

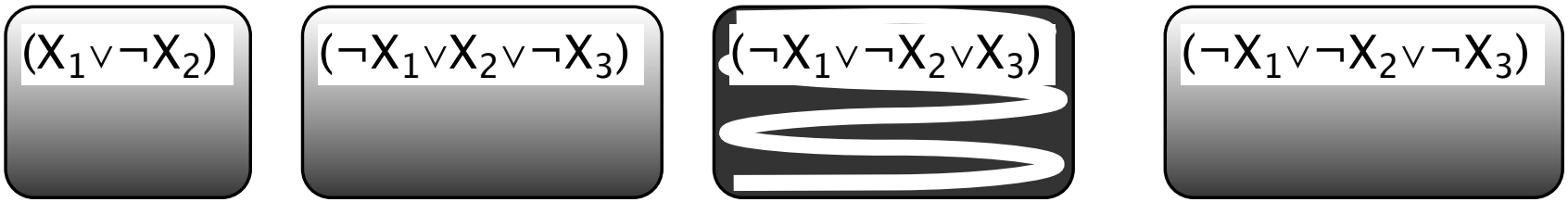
SAT
Esempio



$X_1=1$ $X_2=1$ $X_3=1$

1. Data una possibile soluzione, verificarla è facile.
2. Le possibili soluzioni sono "tante" ($2^{|\mathbf{V}|}$).

SAT
ESEMPIO



$X_1=1$ $X_2=1$ $X_3=0$

1. Data una possibile soluzione, verificarla è facile.
2. Le possibili soluzioni sono "tante" ($2^{|\mathbf{V}|}$).

SAT
Esempio

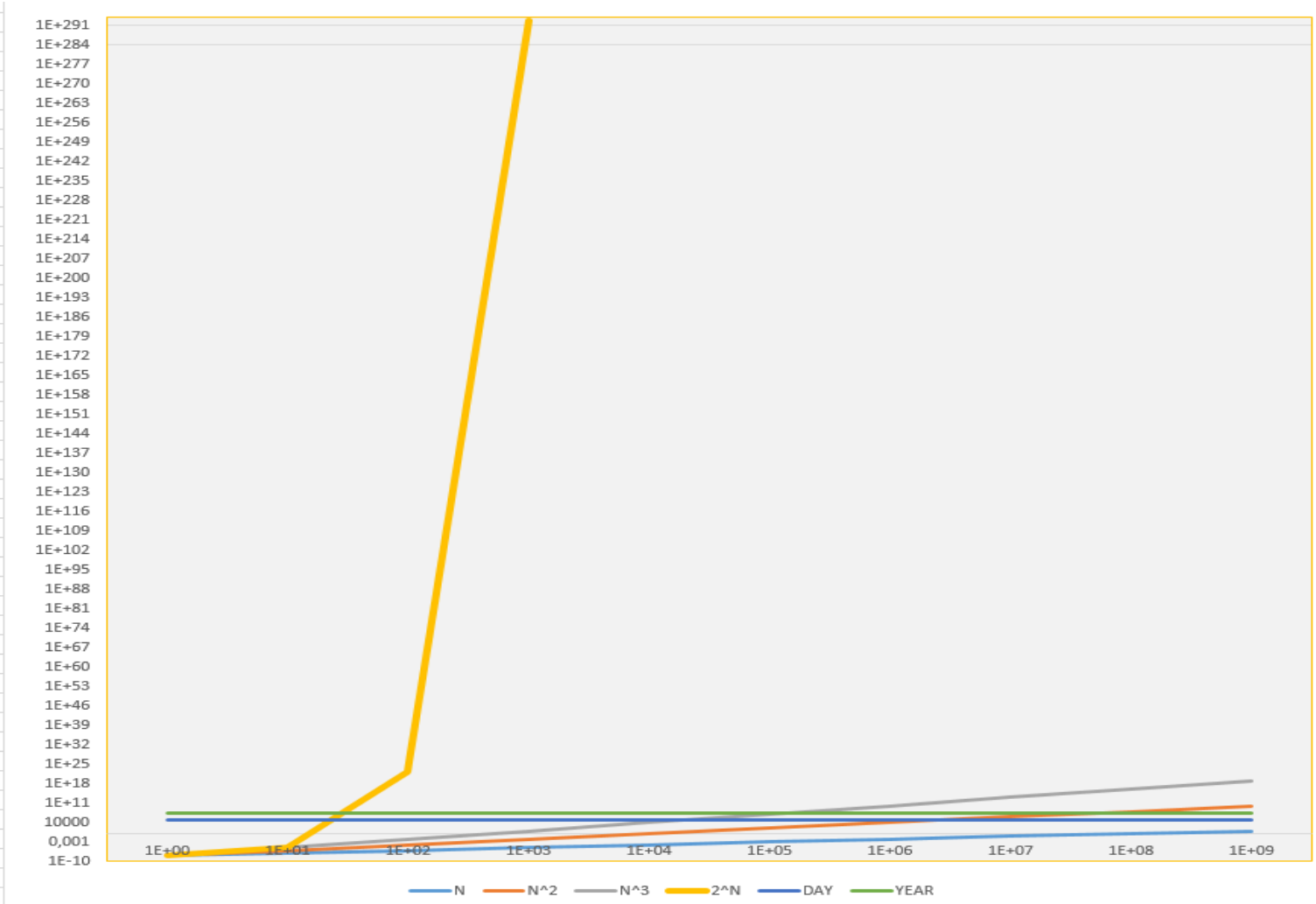
$(X_1 \vee \neg X_2)$ $(\neg X_1 \vee X_2 \vee \neg X_3)$ $(\neg X_1 \vee \neg X_2 \vee X_3)$ $(\neg X_1 \vee \neg X_2 \vee \neg X_3)$

$X_1=1$ $X_3=0$ $X_2=0$

1. Data una possibile soluzione, verificarla è facile.
2. Le possibili soluzioni sono "tante" ($2^{|\mathbf{V}|}$).

Algoritmi e Complessità

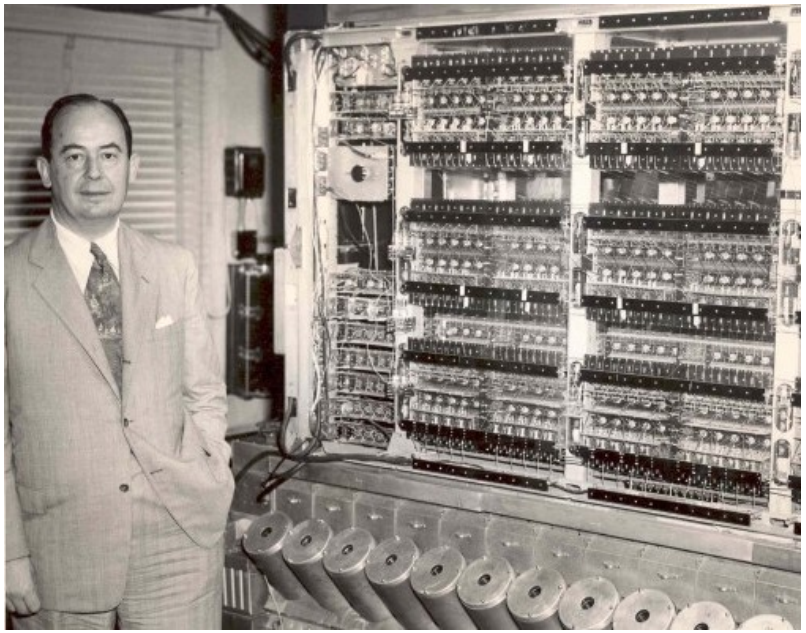
Supponiamo 1 operazione ogni 10^{-9}s



La lettera di Gödel a von Neumann

HTTPS://RJLIPTON.WORDPRESS.COM/THE-GDEL-LETTER/

(1906–1978)



(1903–1957)

La lettera di Gödel a von Neumann

HTTPS://RJLIPTON.WORDPRESS.COM/THE-GDEL-LETTER/

Princeton, 20 March 1956

Lieber Herr v. Neumann:

With the greatest sorrow I have learned of your illness. [...] I would like to allow myself to write you about a mathematical problem, of which your opinion would very much interest me: One can obviously easily construct a Turing machine, which for every formula F in first order predicate logic and every natural number n, allows one to decide if there is a proof of F of length n (length = number of symbols). Let $\psi(F, n)$ be the number of steps the machine requires for this and let $\varphi(n) = \max_F \psi(F, n)$. The question is how fast $\varphi(n)$ grows for an optimal machine. [...] If there really were a machine with $\varphi(n) \sim kn$ (or even $\sim kn^2$), this would have consequences of the greatest importance. Namely, it would obviously mean that in spite of the undecidability of the Entscheidungsproblem, the mental work of a mathematician concerning Yes-or-No questions could be completely replaced by a machine. [...] Now it seems to me, however, to be completely within the realm of possibility that $\varphi(n)$ grows that slowly. [...]

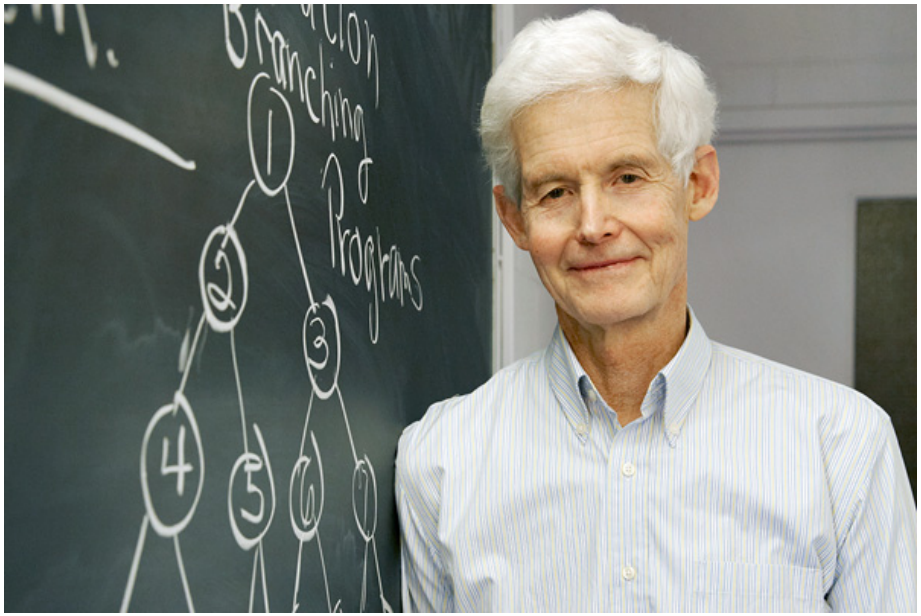
SAT e Gödel

Dunque Gödel era ottimista sull'esistenza di un algoritmo polinomiale (addirittura lineare o quadratico) per la risoluzione del problema SAT.

Purtroppo non sappiamo cosa ne pensasse von Neumann (che morí di Cancro poco dopo).

60 anni dopo non sappiamo ancora se Gödel avesse ragione. Nè se avesse torto.

Arrivano Cook e Levin



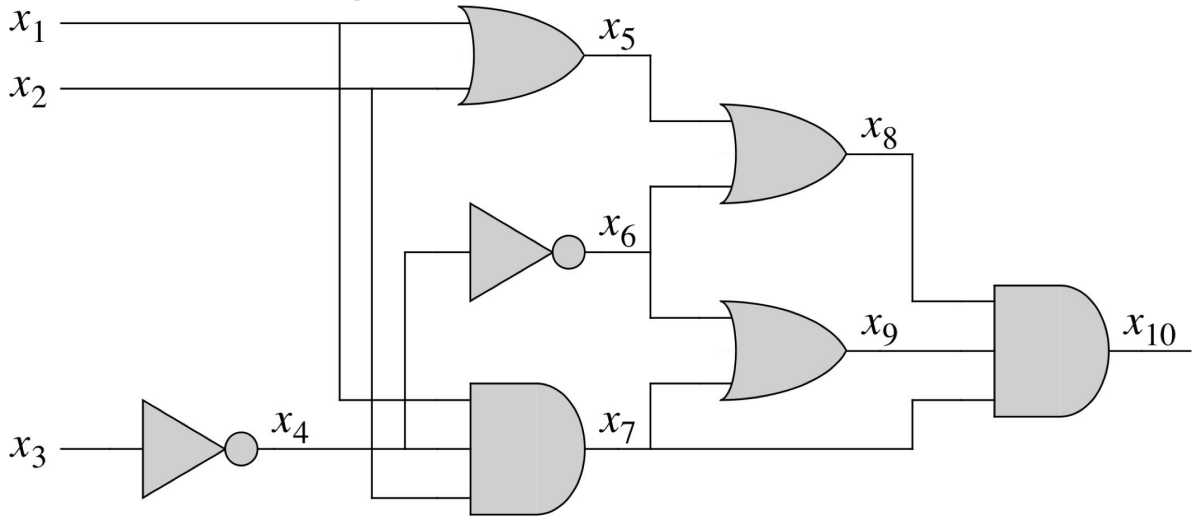
1939, Turing Award 1982

1948, Knuth Prize 2012

NP COMPLETEZZA

CIRCUIT SAT È NP-COMPLETO (IDEM PER SAT)

Input: Un circuito logico con porte `and`, `or`, e `not` con una sola uscita e n porte di ingresso.
Problema: Stabilire se esiste un assegnamento per le porte di ingresso che rende l'output `true`.



NP COMPLETEZZA

SIGNIFICATO

- Ogni problema in cui sappiamo verificare facilmente una soluzione
- Ma non abbiamo idea di come trovarla se non provandole tutte
- (Turni di lezione, calendario campionati sportivi, turni di lavoro, percorsi ottimali per flotte di TIR, predizione di strutture biologiche, . . .)
- SI RIDUCE a SAT
- Senza entrae nei dettagli: se trovo un algoritmo efficiente per SAT lo eredito per qualunque di quei problemi.

I PROBLEMI DEL MILLENNIO

HTTP://WWW.CLAYMATH.ORG/MILLENNIUM-PROBLEMS

1. Yang-Mills and Mass Gap
2. Riemann Hypothesis
3. P vs NP Problem. If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given n cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.
4. Navier-Stokes Equation
5. Hodge Conjecture
6. Poincaré Conjecture (risolta da Grigoriy Perelman nel 2003)
7. Birch and Swinnerton-Dyer Conjecture

On *P versus NP*

- Abbiamo visto Gödel e Von Neumann
- *Now my general conjecture is as follows: for almost all sufficiently complex types of enciphering . . . the mean key computation length increases exponentially with the length of the key, or in other words, the information content of the key . . . The nature of this conjecture is such that I cannot prove it, even for a special type of ciphers. Nor do I expect it to be proven.* [John Nash, 1955 (Nobel Economia 1994)]
- *P versus NP* — a gift to mathematics from computer science [Steve Smale (Fields 1996, Wolf 2006)]
- *The P versus NP problem deals with the central mystery of computation. The story of the long assault on this problem is our Iliad and our Odyssey; it is the defining myth of our field.* [Eric Allender, 2009]

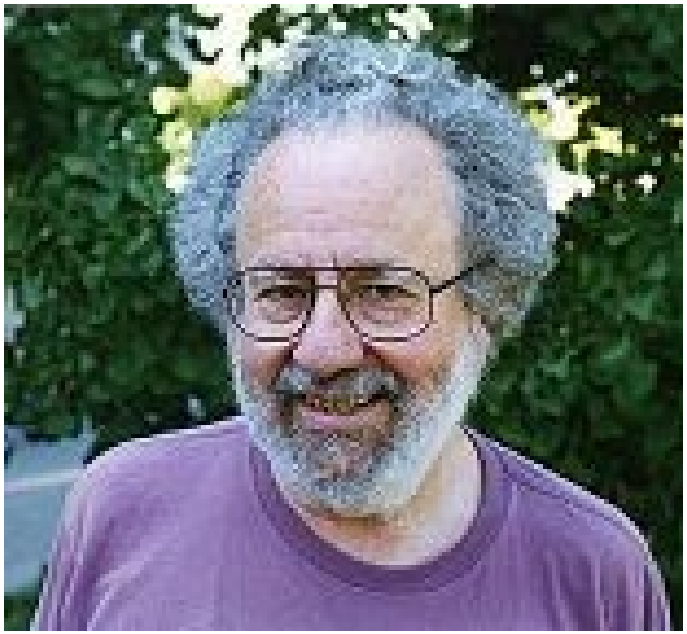
On P versus NP

Grazie a Cook-Levin:

- Per mostrare che $P = NP$ sarebbe sufficiente scrivere un algoritmo polinomiale che risolva SAT.
- Per mostrare che $P \subset NP$ ($P \neq NP$) bisognerebbe dimostrare che un tale algoritmo non esiste.

e si vincerebbe un milione di dollari (e gloria imperitura)

SAT SOLVING



Martin Davis (1928)

Hillary Putnam (1926–2016)

SAT SOLVING

UNIT PROPAGATION: UNA PRIMA FORMA DI RAGIONAMENTO

- Scegliamo una clausola

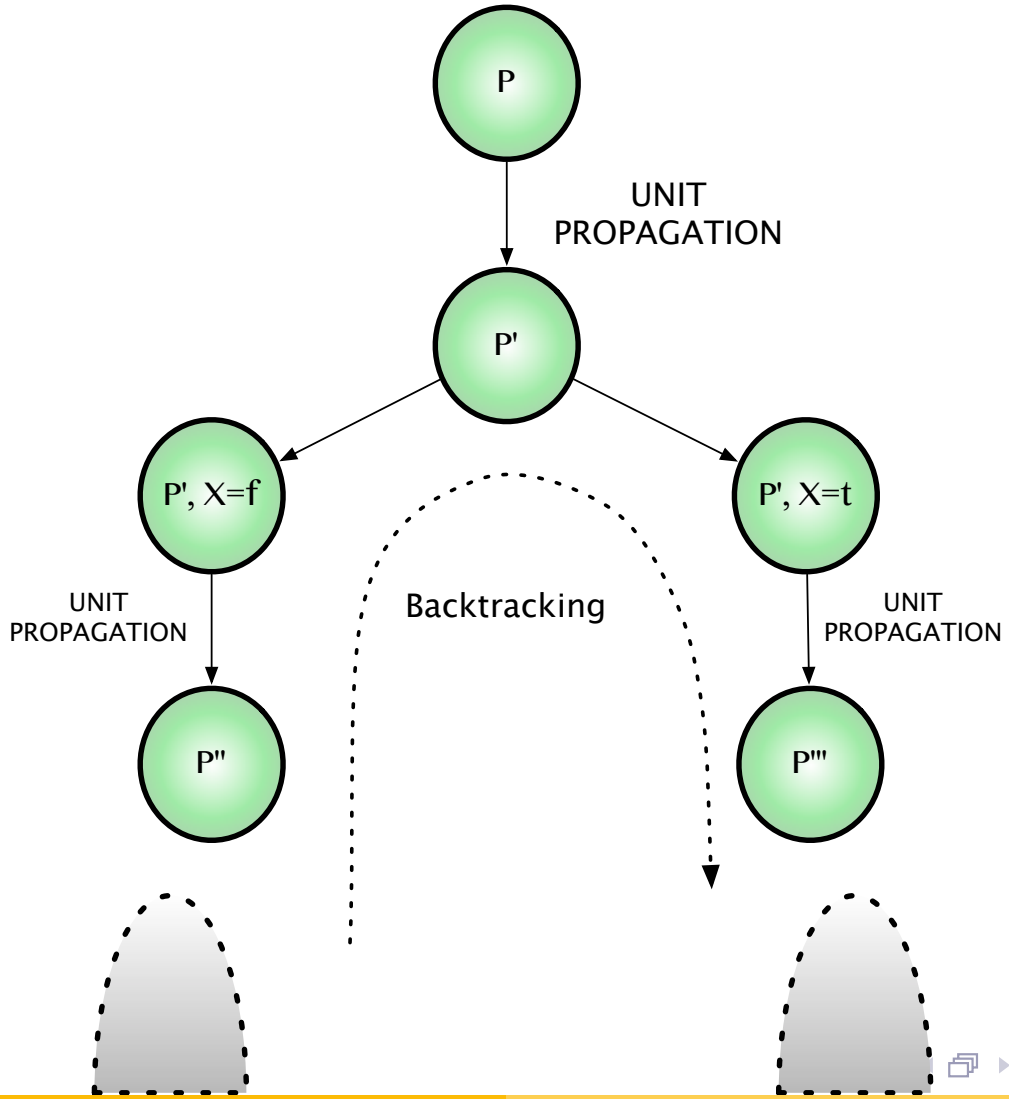
$$X_1 \vee \neg X_2 \vee X_3$$

  Assumete che l'assegnamento parziale sia $X_1 = 0, X_2 = 1$, inferiamo deterministicamente che $X_3 = 1$.
- Tutte le clausole non ancora soddisfatte sono verificate e si procede ad inferire conseguenze di questo tipo (finchè si può).
- Se invece fosse

$$X_1 \vee \neg X_2 \vee X_3 \vee X_4$$

  Assumete che l'assegnamento parziale sia $X_1 = 0, X_2 = 1$, una tra X_3 e X_4 deve essere vera ma devo esplorare più strade.

SAT SOLVING
DPLL (1962)



SAT SOLVING
DPLL (1962)

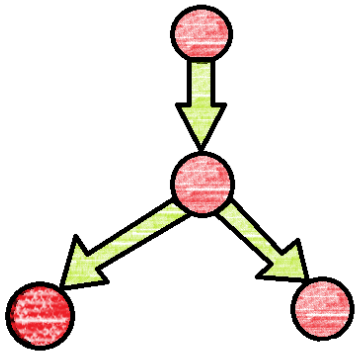
Given Φ (CNF), establishing whether exists θ s.t. $\Phi\theta$ is true.

DP(Φ,θ)
  $\theta' \leftarrow$ unit_propagation(Φ,θ)
  if (ok($\Phi\theta'$)) return θ'
  else if (ko($\Phi\theta'$)) return **false**
  else $X \leftarrow$ select_variable(Φ,θ')
    $\theta'' \leftarrow$DP(Φ, $\theta'[X/\textbf{true}]$)
    if ($\theta'' \neq$ **false**) return θ''
    else return DP(Φ, $\theta'[X/\textbf{false}]$)

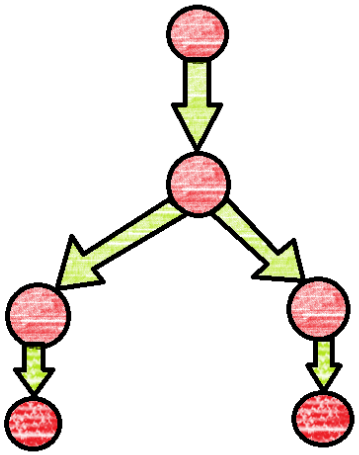
SEARCHING=PROPAGATION+ND ASSIGNMENT

SEARCHING=PROPAGATION+ND ASSIGNMENT

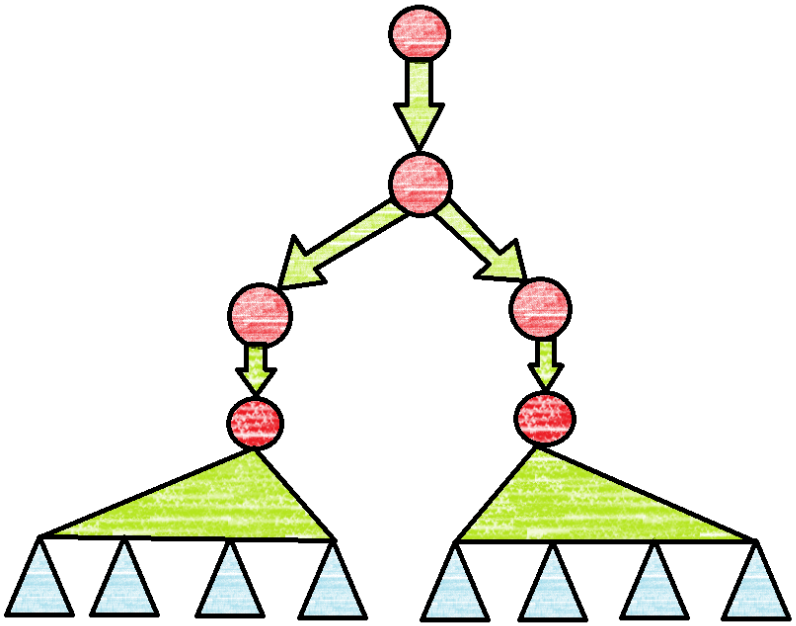
SEARCHING=PROPAGATION+ND ASSIGNMENT



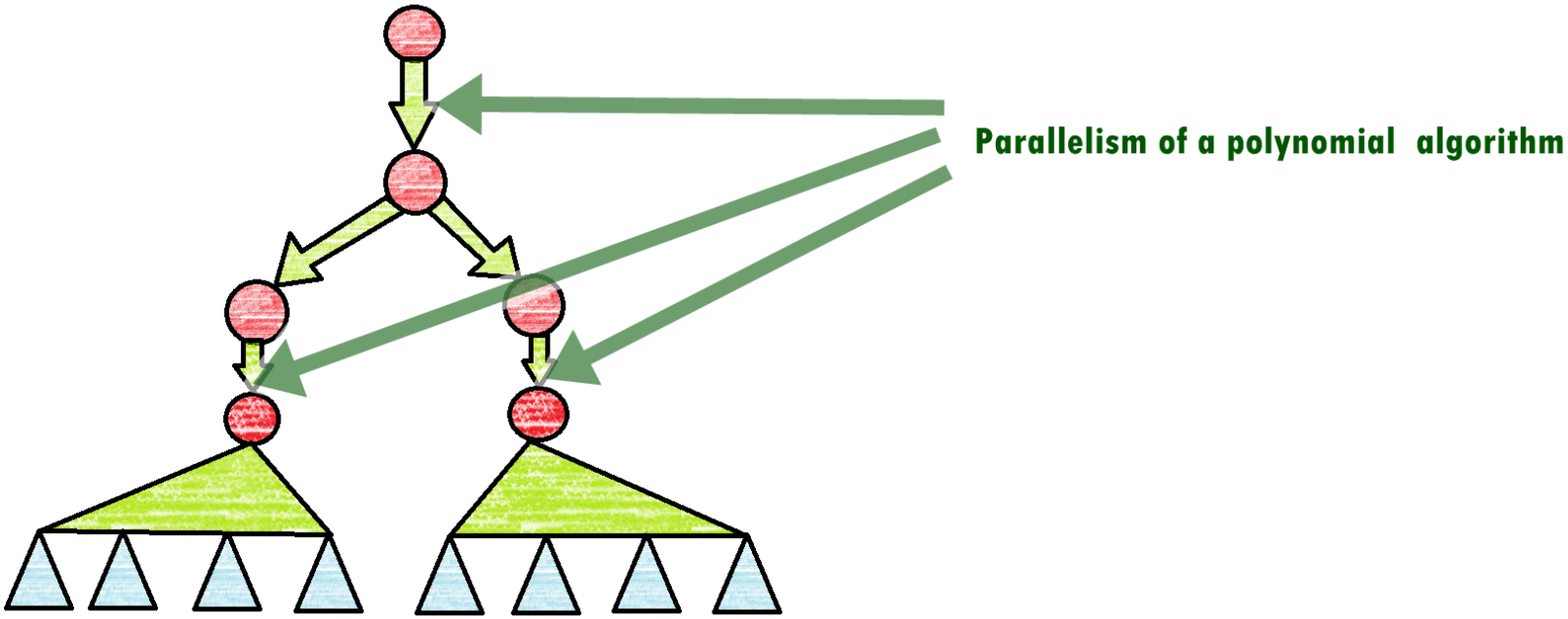
SEARCHING=PROPAGATION+ND ASSIGNMENT



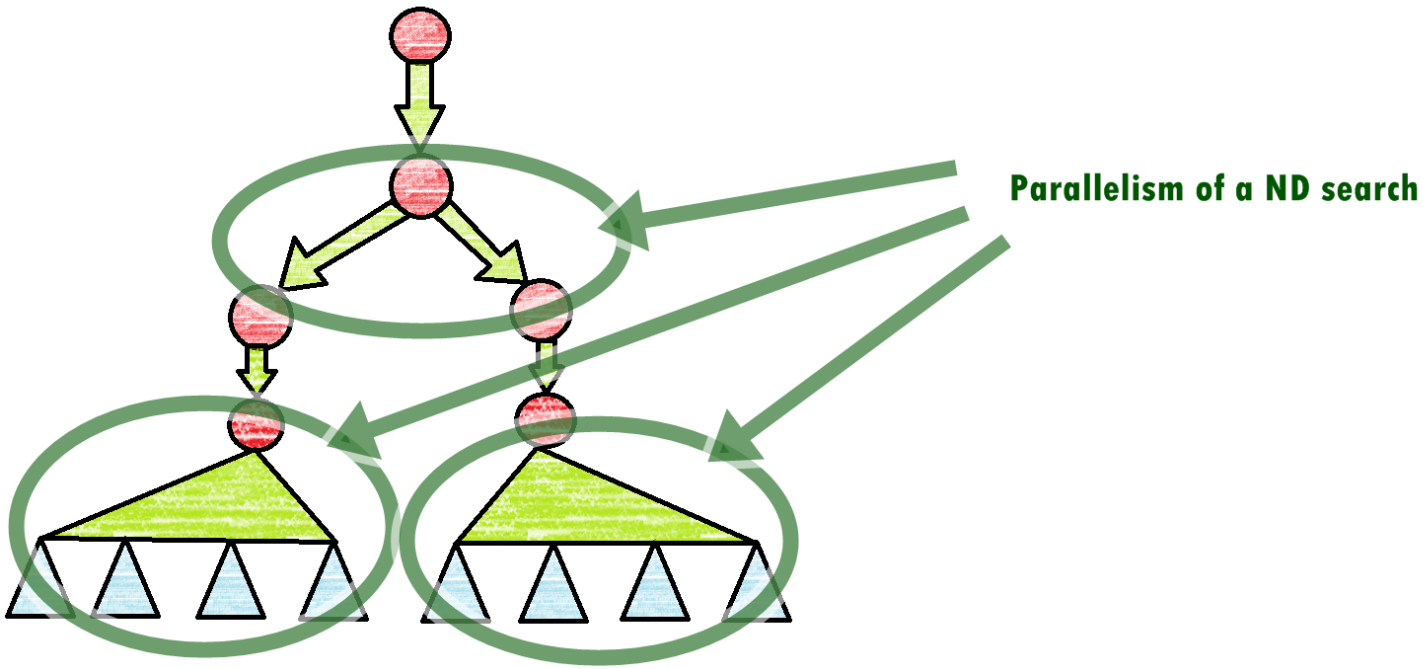
SEARCHING=PROPAGATION+ND ASSIGNMENT



SEARCHING=PROPAGATION+ND ASSIGNMENT



SEARCHING=PROPAGATION+ND ASSIGNMENT



SAT COMPETITION

- Come abbiamo visto, SAT è il primo problema mostrato essere NP-completo
- Ogni problema in NP può essere ridotto a SAT
- Una soluzione efficiente per SAT sarebbe ereditata da tutti i problemi in NP
- Pertanto da DPLL in poi parte la progettazione di SAT SOLVERS
- Dal 2002 viene organizzata la cosiddetta SAT COMPETITION `http://www.satcompetition.org/`
- Un enorme speed-up avvenne con l'introduzione del cosiddetto conflict-driven clause learning.
- Vengono usate "euristiche" basate su analisi di "features"

Il formato DIMACS

- Per la SAT competition è stato fissato un formato standard.
- Le istanze sono in un file ASCII (con suffisso ".cnf")
- Le variabili sono rappresentate da numeri interi (eccetto lo zero), con il $-$ si denota la negazione.
- I commenti iniziano con "c", lo "0" termina le clausole
- C'è una linea iniziale iniziante con "p" che memorizza il numero di variabili e clausole.

$$(X_1 \vee \neg X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_3)$$

```
c ****************************************
c SAT Encoding of problem above
c ****************************************
p cnf 3 2
1  -2  3 0
-1 -3 0
```

IDEA: RISOLVERE PROBLEMI NP USANDO SAT

- Scrivo un programma nel mio linguaggio preferito che mi crea una istanza di SAT nel formato DIMACS "equivalente" al mio problema.
- In pratica faccio una riduzione.
- Uso un sat solver (dal sito della competizione ne posso scaricare diversi)
- Un grande classico è MiniSAT `http://minisat.se/`
- Se il problema da affrontare è NP-completo, non ve ne pentirete!

ESEMPIO: SUDOKU

				1				
			2		3			
		4				5		
	6						7	
8				5				9
	1						3	
		5				4		
			7		1			
				9				

ESEMPIO: SUDOKU

- In ogni coppia (r, c) in $\{(1,1),(1,2),\ldots,(9,8),(9,9)\}$ devo mettere uno ed un solo numero da 1 a 9.
- Introduco 9 variabili per ogni cella (r, c): $X_1^{(r,c)}, \ldots, X_9^{(r,c)}$
- Devo dire che almeno una di esse è vera:

$$X_1^{(r,c)} \vee X_2^{(r,c)} \vee X_3^{(r,c)} \vee X_4^{(r,c)} \vee X_5^{(r,c)} \vee X_6^{(r,c)} \vee X_7^{(r,c)} \vee X_8^{(r,c)} \vee X_9^{(r,c)}$$

- Devo dire che mai due di esse sono vere ovvero, per $i = 1, \ldots, 8$ per $j = i + 1, \ldots, 9$:

$$\neg X_i^{(r,c)} \vee \neg X_j^{(r,c)}$$

- Stiamo parlando di 9^3 variabili, dunque di uno spazio di ricerca di dimensione $2^{729} \approx 2.8 \cdot 10^{219}$

ESEMPIO: SUDOKU

- In ogni riga r per ogni valore k dall'1 al 9 c'è almeno una volta
- Devo dire che almeno una di esse è vera:

$$X_k^{(r,1)} \vee X_k^{(r,2)} \vee X_k^{(r,3)} \vee X_k^{(r,4)} \vee X_k^{(r,5)} \vee X_k^{(r,6)} \vee X_k^{(r,7)} \vee X_k^{(r,8)} \vee X_k^{(r,9)}$$

- Ma non due volte: Devo dire che mai due di esse sono vere ovvero, per $i = 1, \ldots, 8$ per $j = i + 1, \ldots, 9$:

$$\neg X_k^{(r,i)} \vee \neg X_k^{(r,j)}$$

- Analogamente per le colonne (ve la lascio)

ESEMPIO: SUDOKU

0	0	0	1	1	1	2	2	2
0	0	0	1	1	1	2	2	2
0	0	0	1	1	1	2	2	2
3	3	3	4	4	4	5	5	5
3	3	3	4	4	4	5	5	5
3	3	3	4	4	4	5	5	5
6	6	6	7	7	7	8	8	8
6	6	6	7	7	7	8	8	8
6	6	6	7	7	7	8	8	8

Identifico i sottoquadrati

- Se le righe r vanno da 0 a 8 e le colonne c vanno da 0 a 8
- La cella (r, c) sta nel quadrato $3 * (r \,\texttt{div}\, 3) + c \,\texttt{div}\, 3$.
- Con questa idea si mettono i vincoli (analoghi ai precedenti) sulle variabili in ogni sottoquadrato.
  Se r e c vanno da 1 a 9 (anziché da 0 a 8) basta mettere qualche "+1" e "-1" :)
- Quanto visto non dipende dalla specifica istanza!

ESEMPIO: SUDOKU

				1				
			2		3			
		4				5		
	6						7	
8				5				9
	1						3	
		5				4		
			7		1			
				9				

Devo esplicitare i fatti noti

- $X_1^{(1,5)}$
- $X_2^{(2,4)}$ e $X_3^{(2,6)}$
- ...
- $X_9^{(9,5)}$

ESEMPIO: SUDOKU

A questo punto dobbiamo solo trovare un mapping sensato tra

$$X_k^{(r,c)}$$

e un numero da 1 a 729 (per il DIMACS format)

Diamo un occhiata a un codice in C e alla sua esecuzione.

Conclusioni

- Importanza teorica e pratica del problema SAT:
- Il problema P vs NP vi aspetta!
- Se dovete risolvere un problema NP completo NON scrivete una soluzione ad-hoc ma trasformatelo in istanze di SAT.
- I SAT solver (ma anche gli ASP solvers, i CP solvers, talvolta i tools della ricerca operativa basati sul simplesso, talvolta i tools approssimati di ricerca locale) risolvono il vostro problema meglio di quello che potreste fare in diversi mesi, forse anni. Sono basati su tecniche generali sviluppate per più di 50 anni e testate su benchmarks via via più difficili.