Constraint Programming & Planning

Agostino Dovier

Università di Udine
Dipartimento di Matematica e Informatica

Udine, Marzo 2011

SOMMARIO

METODI PER RISOLVERE CSP/COP

- SAT
- Answer Set Programming

SAT

Definizione

- Dato un insieme di variabili $X_1, \ldots, X_n$ a valori Booleani e
- Una formula logica proposizional ottenuta da esse, già in forma normale

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

  ove ogni ℓ_j^i è una variabile X_p o la sua negazione $\neg X_p$
- Stabilire (e trovare) un assegnamento delle variabili in {false,true} (o {0,1}) che renda vera la formula.
- Ricordiamo che:
  - $0 \vee 0 = 0, 0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$
  - $0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0, 1 \wedge 1 = 1$
  - $\neg 0 = 1, \neg 1 = 0$

SAT

Definizione

- Dato un insieme di variabili $X_1, \ldots, X_n$ a valori Booleani e
- Una formula logica proposizional ottenuta da esse, già in forma normale

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

  ove ogni ℓ_j^i è una variabile X_p o la sua negazione $\neg X_p$
- Stabilire (e trovare) un assegnamento delle variabili in {false,true} (o {0,1}) che renda vera la formula.
- Ricordiamo che:
  - $0 \vee 0 = 0, 0 \vee 1 = 1 \vee 0 = 1 \vee 1 = 1$
  - $0 \wedge 0 = 0 \wedge 1 = 1 \wedge 0 = 0, 1 \wedge 1 = 1$
  - $\neg 0 = 1, \neg 1 = 0$

SAT

- SAT è stato il primo problema dimostrato essere NP-completo (Cook)
- Ogni problema in NP si può trasformare in un problema di SAT.
- Un CSP su domini finiti è tipicamente in NP (salvo uso di vincoli particolari che potrebbero farci addirittura “uscire” da NP)
- Pertanto potremmo studiare la *riduzione* del nostro problema a un’istanza di SAT.
- Esistono vari SAT solvers liberi e particolarmente efficienti (p.es. MiniSAT, PicoSAT, RelSAT, etc.—si veda il sito della SAT competition) che possono poi essere usati indirettamente per risolvere il nostro problema.
- Le riduzioni non sono di solito immediate.

SAT

Esempio: Le 4 regine

- Dobbiamo dire che le variabili X_1, X_2, X_3, X_4 hanno valore da 1 a 4.
- Per ognuna di esse (poniamo X_i) introduciamo 4 variabili Booleane: $Z_1^i, Z_2^i, Z_3^i, Z_4^i$.
- Devo dire che una di esse è vera: $Z_1^i \vee Z_2^i \vee Z_3^i \vee Z_4^i$.
- Devo dire che due non sono vere: $\neg Z_1^i \vee \neg Z_2^i, \neg Z_1^i \vee \neg Z_3^i, \neg Z_1^i \vee \neg Z_4^i, \ldots, \neg Z_3^i \vee \neg Z_4^i$.
- Poi devo mettere i vincoli di attacco orizzontale: per $i \neq j$ devo dire: $\neg Z_1^i \vee \neg Z_1^j, \neg Z_2^i \vee \neg Z_2^j, \neg Z_3^i \vee \neg Z_3^j, \neg Z_4^i \vee \neg Z_4^j$.
- Infine devo mettere i vincoli di attacco diagonale: per $i \neq j$. Poniamo $i = 2, j = 4$, devo dire: $\neg Z_1^2 \vee \neg Z_3^4, \neg Z_2^2 \vee \neg Z_4^4, \neg Z_3^2 \vee \neg Z_1^4, \neg Z_4^2 \vee \neg Z_2^4$.

SAT

Esempio: Le 4 regine

- Dobbiamo dire che le variabili X_1, X_2, X_3, X_4 hanno valore da 1 a 4.
- Per ognuna di esse (poniamo X_i) introduciamo 4 variabili Booleane: $Z_1^i, Z_2^i, Z_3^i, Z_4^i$.
- Devo dire che una di esse è vera: $Z_1^i \vee Z_2^i \vee Z_3^i \vee Z_4^i$.
- Devo dire che due non sono vere:
  $\neg Z_1^i \vee \neg Z_2^i, \neg Z_1^i \vee \neg Z_3^i, \neg Z_1^i \vee \neg Z_4^i, \ldots, \neg Z_3^i \vee \neg Z_4^i$.
- Poi devo mettere i vincoli di attacco orizzontale: per $i \neq j$ devo dire:
  $\neg Z_1^i \vee \neg Z_1^j, \neg Z_2^i \vee \neg Z_2^j, \neg Z_3^i \vee \neg Z_3^j, \neg Z_4^i \vee \neg Z_4^j$.
- Infine devo mettere i vincoli di attacco diagonale: per $i \neq j$. Poniamo $i = 2, j = 4$, devo dire:
  $\neg Z_1^2 \vee \neg Z_3^4, \neg Z_2^2 \vee \neg Z_4^4, \neg Z_3^2 \vee \neg Z_1^4, \neg Z_4^2 \vee \neg Z_2^4$.

SAT

Esempio: Le 4 regine

- Dobbiamo dire che le variabili X_1, X_2, X_3, X_4 hanno valore da 1 a 4.
- Per ognuna di esse (poniamo X_i) introduciamo 4 variabili Booleane: $Z_1^i, Z_2^i, Z_3^i, Z_4^i$.
- Devo dire che una di esse è vera: $Z_1^i \vee Z_2^i \vee Z_3^i \vee Z_4^i$.
- Devo dire che due non sono vere: $\neg Z_1^i \vee \neg Z_2^i, \neg Z_1^i \vee \neg Z_3^i, \neg Z_1^i \vee \neg Z_4^i, \ldots, \neg Z_3^i \vee \neg Z_4^i$.
- Poi devo mettere i vincoli di attacco orizzontale: per $i \neq j$ devo dire: $\neg Z_1^i \vee \neg Z_1^j, \neg Z_2^i \vee \neg Z_2^j, \neg Z_3^i \vee \neg Z_3^j, \neg Z_4^i \vee \neg Z_4^j$.
- Infine devo mettere i vincoli di attacco diagonale: per $i \neq j$. Poniamo $i = 2, j = 4$, devo dire: $\neg Z_1^2 \vee \neg Z_3^4, \neg Z_2^2 \vee \neg Z_4^4, \neg Z_3^2 \vee \neg Z_1^4, \neg Z_4^2 \vee \neg Z_2^4$.

SAT

Esempio: Le 4 regine

- Dobbiamo dire che le variabili X_1, X_2, X_3, X_4 hanno valore da 1 a 4.
- Per ognuna di esse (poniamo X_i) introduciamo 4 variabili Booleane: $Z_1^i, Z_2^i, Z_3^i, Z_4^i$.
- Devo dire che una di esse è vera: $Z_1^i \vee Z_2^i \vee Z_3^i \vee Z_4^i$.
- Devo dire che due non sono vere: $\neg Z_1^i \vee \neg Z_2^i, \neg Z_1^i \vee \neg Z_3^i, \neg Z_1^i \vee \neg Z_4^i, \ldots, \neg Z_3^i \vee \neg Z_4^i$.
- Poi devo mettere i vincoli di attacco orizzontale: per $i \neq j$ devo dire: $\neg Z_1^i \vee \neg Z_1^j, \neg Z_2^i \vee \neg Z_2^j, \neg Z_3^i \vee \neg Z_3^j, \neg Z_4^i \vee \neg Z_4^j$.
- Infine devo mettere i vincoli di attacco diagonale: per $i \neq j$. Poniamo $i = 2, j = 4$, devo dire: $\neg Z_1^2 \vee \neg Z_3^4, \neg Z_2^2 \vee \neg Z_4^4, \neg Z_3^2 \vee \neg Z_1^4, \neg Z_4^2 \vee \neg Z_2^4$.

- Dobbiamo dire che le variabili X_1, X_2, X_3, X_4 hanno valore da 1 a 4.
- Per ognuna di esse (poniamo X_i) introduciamo 4 variabili Booleane: $Z_1^i, Z_2^i, Z_3^i, Z_4^i$.
- Devo dire che una di esse è vera: $Z_1^i \vee Z_2^i \vee Z_3^i \vee Z_4^i$.
- Devo dire che due non sono vere: $\neg Z_1^i \vee \neg Z_2^i, \neg Z_1^i \vee \neg Z_3^i, \neg Z_1^i \vee \neg Z_4^i, \ldots, \neg Z_3^i \vee \neg Z_4^i$.
- Poi devo mettere i vincoli di attacco orizzontale: per $i \neq j$ devo dire: $\neg Z_1^i \vee \neg Z_1^j, \neg Z_2^i \vee \neg Z_2^j, \neg Z_3^i \vee \neg Z_3^j, \neg Z_4^i \vee \neg Z_4^j$.
- Infine devo mettere i vincoli di attacco diagonale: per $i \neq j$. Poniamo $i = 2, j = 4$, devo dire: $\neg Z_1^2 \vee \neg Z_3^4, \neg Z_2^2 \vee \neg Z_4^4, \neg Z_3^2 \vee \neg Z_1^4, \neg Z_4^2 \vee \neg Z_2^4$.

SAT

ESEMPIO: LE 4 REGINE

- Dobbiamo dire che le variabili X_1, X_2, X_3, X_4 hanno valore da 1 a 4.
- Per ognuna di esse (poniamo X_i) introduciamo 4 variabili Booleane: $Z_1^i, Z_2^i, Z_3^i, Z_4^i$.
- Devo dire che una di esse è vera: $Z_1^i \vee Z_2^i \vee Z_3^i \vee Z_4^i$.
- Devo dire che due non sono vere: $\neg Z_1^i \vee \neg Z_2^i, \neg Z_1^i \vee \neg Z_3^i, \neg Z_1^i \vee \neg Z_4^i, \ldots, \neg Z_3^i \vee \neg Z_4^i$.
- Poi devo mettere i vincoli di attacco orizzontale: per $i \neq j$ devo dire: $\neg Z_1^i \vee \neg Z_1^j, \neg Z_2^i \vee \neg Z_2^j, \neg Z_3^i \vee \neg Z_3^j, \neg Z_4^i \vee \neg Z_4^j$.
- Infine devo mettere i vincoli di attacco diagonale: per $i \neq j$. Poniamo $i = 2, j = 4$, devo dire: $\neg Z_1^2 \vee \neg Z_3^4, \neg Z_2^2 \vee \neg Z_4^4, \neg Z_3^2 \vee \neg Z_1^4, \neg Z_4^2 \vee \neg Z_2^4$.

SAT

Esempio: Le 4 regine

- Il tutto va poi codificato (parte facile) in formato DIMACS (suffisso .cnf)
- Le variabili sono numeri interi (non 0)
- Serve un mapping tra var e interi

```
c ****************************
c Codifica in SAT del SUDOKU
c ****************************
p cnf 729 12015
54 0
85 0
161 0
....
1 2 3 4 5 6 7 8 9 0
-1 -2 0
-1 -3 0
-1 -4 0
....
```

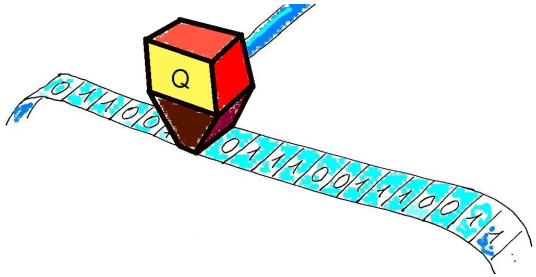
SAT

Quando si usa

- Problemi non (troppo) numerici
- Comunque problemi di cui si conosce già bene la formalizzazione ad alto livello (ogni cambiamento ci potrebbe costringere a riformulare il tutto)

ANSWERS SET PROGRAMMING

Sappiamo che già un piccolo sottoinsieme di Prolog (definite clause programming) è *Turing completo*.




```
delta(q0,0,qi,1,left).
...
delta(qn,1,qj,0,right).
```

```
turing(Left,halt,S,Right,Left,halt,S,Right).

turing([L|L_i],Q,S,R_i,L_o,Q_o,S_o,R_o) :-
     delta(Q,S,Q1,S1,left),
     turing(L_i,Q1,L,[S1|R_i],L_o,Q_o,S_o,R_o).
turing(L_i,Q,S,[R|R_i],L_o,Q_o,S_o,R_o) :-
     delta(Q,S,Q1,S1,right),
     turing([S1|L_i],Q1,R,R_i,L_o,Q_o,S_o,R_o).
turing([],Q,S,R_i,L_o,Q,S_o,R_o) :-
      turing([0],Q,S,R_i,L_o,Q,S_o,R_o).
turing(L_i,Q,S,[],L_o,Q,S_o,R_o) :-
      turing(L_i,Q,S,[0],L_o,Q,S_o,R_o).
```

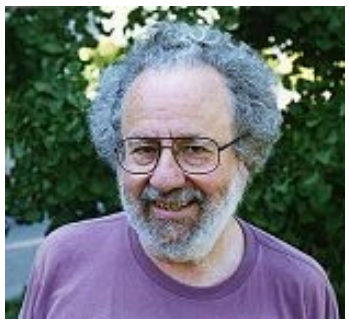
Answers Set Programming

Definite clause programming ha delle proprietà semantiche entusiasmanti

$$P \text{ has a model} \Leftrightarrow P \text{ has a Herbrand model}$$

$$M_P = \bigcap_{M \text{ is a Herbrand model of } P} = T_P \uparrow \omega(\emptyset)$$

$$M_P = \{A \ : \ \text{there is a SLD resolution for } A \text{ from } P\}$$

ANSWERS SET PROGRAMMING

Ammettendo la negazione LP può essere usato per *Knowledge representation* and *non monotonic reasoning*.

```
flies(X) :- bird(X), not abnormal_bird(X).
abnormal_bird(Y) :- penguin(Y).
abnormal_bird(Y) :- roadrunner(Y).
bird(Z) :- penguin(Z).
bird(tweety).                    penguin(pingu).
```

Ma la semantica ora non ha la stessa eleganza. Stable model semantics (Gelfond-Lifschitz) è la tecnica accettata in questo caso. È NP-computable nei programmi che non usano simboli di funzione arbitrari.

Answers Set Programming

Ammettendo la negazione LP può essere usato per *Knowledge representation* and *non monotonic reasoning*.

```
flies(X) :- bird(X), not abnormal_bird(X).
abnormal_bird(Y) :- penguin(Y).
abnormal_bird(Y) :- roadrunner(Y).
bird(Z) :- penguin(Z).
bird(tweety).                    penguin(pingu).
```

Ma la semantica ora non ha la stessa eleganza. Stable model semantics (Gelfond-Lifschitz) è la tecnica accettata in questo caso. È NP-computable nei programmi che non usano simboli di funzione arbitrari.

ANSWER SET PROGRAMMING

- ASP = Logic Programming CON negazione SENZA simboli di funzione, SEMANTICA del modello stabile.
- È un linguaggio per NP
- Diversi solvers (gratutiti) sono stati sviluppati negli ultimi anni (smodels, cmodels, clasp, DLV, ...) per calcolare i modelli stabili.
- In particolare clasp ha vinto diverse *tracks* dell'ultima SAT competition.
- Alcune estensioni sintattiche sono state introdotte.
- Ad esempio `:- A1,...,An` che significa che in ogni modello NON può valere la congiunzione `A1,...,An` (i.e. almeno uno è falso).
- Ad esempio `a{A1,...,An}b` che significa che in ogni modello devono valere un numero tra `a` e `b` di atomi `A1,...,An`.

ANSWER SET PROGRAMMING

CODIFICA DELLE N-REGINE


```
numero(1..n).

%%% queen( -colonna- , -riga.)
1{queen(I,J) : numero(I)}1 :- numero(J).

%%% vincoli orizzontali
:- queen(I,J), queen(I,J1),
   numero(I),numero(J),numero(J1),
   J != J1.

%%% vincoli diagonali
:- queen(I,J), queen(I1,J1),
   numero(I),numero(I1),numero(J),numero(J1),
   I != I1, abs(I1-I)== abs(J1-J).
```

Answer Set Programming

Quando si usa?

- La tecnica risolutiva assomiglia a quella di SAT. Dunque applicabilità simile (problemi non troppo numerici).
- Ma scrivere un programma ASP è molto più naturale e semplice che scrivere un file `.cnf` per un SAT solver.