Constraint Programming & Planning

Agostino Dovier

Università di Udine
Dipartimento di Matematica e Informatica

Udine, Aprile 2011

Sommario

Programmi Generali

- Completamento
- Modelli well-founded
- Grounding
- Modelli Stabili
- Riassunto

SEMANTICA DEI PROGRAMMI GENERALI

"Quando si deve fornire risposta positiva ad un goal della forma: $\leftarrow$ `not` A ?"

CWA: quando non esiste alcuna SLD-derivazione per il goal $\leftarrow A$ dal programma P
In generale indecidibile

SEMANTICA DEI PROGRAMMI GENERALI

"Quando si deve fornire risposta positiva ad un goal della forma: $\leftarrow$ `not` A ?"

NaF: quando l'SLD-albero per $\leftarrow A$ è un albero di fallimento finito.
Implementata in Prolog con cut & fail.

"Quando si deve fornire risposta positiva ad un goal della forma: $\leftarrow$ `not` A ?"

Herbrand: quando A è falso in tutti i modelli di Herbrand del completamento $Comp(P)$ di P.

Semantica dei Programmi Generali

- CWA è indecidibile e basata su SLD risoluzione: Non è estendibile pertanto a programmi generali.
- NaF lavora anche per programmi generali, ma è molto 'debole'. Basta un ciclo per farla entrare in loop:

```
p(a) :- not p(b).
p(b) :- not p(a).
```

- Sembra fatto apposta, ma cicli di negazione sono dietro l'angolo in KR.
- La piu' generalizzabile sembra quella di Herbrand.

Dato un programma *P*:

```
r(a, c).
r(a, d).
q(X) :- r(X,Y), not s(Y).
p(a) :- not p(b).
p(b) :- not p(a).
```

Lo normalizzo ottenendo *norm*(P):

```
r(X1,X2) :- X1=a, X2=c.
r(X1,X2) :- X1=a, X2=d.
q(X1) :- r(X1,Y), not s(Y).
p(X1) :- X1=a, not p(b).
p(X1) :- X1=b, not p(a).
```

Posso raccogliere le teste uguali e mettere dei se e solo se, e le quantificazioni esplicite ottenendo *iff*(*P*):

$$\texttt{r}(X_1,X_2) \leftrightarrow (X_1=\texttt{a} \wedge X_2=\texttt{c}) \vee (X_1=\texttt{a} \wedge X_2=\texttt{d})$$
$$\texttt{q}(X_1) \leftrightarrow \exists\ \texttt{Y}\ (\texttt{r}(X_1,\texttt{Y}) \wedge \neg\ \texttt{s}(\texttt{Y}))$$
$$\texttt{p}(X_1) \leftrightarrow (X_1=\texttt{a} \wedge \neg\ \texttt{p}(\texttt{b})) \vee (X_1=\texttt{b} \wedge \neg\ \texttt{p}(\texttt{a}))$$
$$\texttt{s}(X_1) \leftrightarrow \texttt{false}$$

Il completamento di P è dunque:

$$\texttt{r}(X_1,X_2) \leftrightarrow (X_1=\texttt{a} \wedge X_2=\texttt{c}) \vee (X_1=\texttt{a} \wedge X_2=\texttt{d})$$
$$\texttt{q}(X_1) \leftrightarrow \exists\ \texttt{Y}\ (\texttt{r}(X_1,\texttt{Y}) \wedge \neg\ \texttt{s}(\texttt{Y}))$$
$$\texttt{p}(X_1) \leftrightarrow (X_1=\texttt{a} \wedge \neg\ \texttt{p}(\texttt{b})) \vee (X_1=\texttt{b} \wedge \neg\ \texttt{p}(\texttt{a}))$$
$$\texttt{s}(X_1) \leftrightarrow \texttt{false}$$

Piu' i freeness axioms **(F1)**, **(F2)** e **(F3)**.

SEMANTICA DEI PROGRAMMI GENERALI

MODELLI DI HERBRAND DEL COMPLETAMENTO

$$r(X_1,X_2) \leftrightarrow (X_1=a \wedge X_2=c) \vee (X_1=a \wedge X_2=d)$$
$$q(X_1) \leftrightarrow \exists\ Y\ (r(X_1,Y) \wedge \neg\ s(Y))$$
$$p(X_1) \leftrightarrow (X_1=a \wedge \neg\ p(b)) \vee (X_1=b \wedge \neg\ p(a))$$
$$s(X_1) \leftrightarrow \mathtt{false}$$

Gli atomi da considerare sono 28:

s(a)	s(b)	s(c)	s(d)
p(a)	p(b)	p(c)	p(d)
q(a)	q(b)	q(c)	q(d)
r(a,a)	r(a,b)	r(a,c)	r(a,d)
r(b,a)	r(b,b)	r(b,c)	r(b,d)
r(c,a)	r(c,b)	r(c,c)	r(c,d)
r(d,a)	r(d,b)	r(d,c)	r(d,d)

Quali stanno in tutti i modelli del completamento?

Quali in nessuno?

Semantica dei Programmi Generali

Modelli di Herbrand del completamento

$r(X_1,X_2) \leftrightarrow (X_1=a \wedge X_2=c) \vee (X_1=a \wedge X_2=d)$

$q(X_1) \leftrightarrow \exists\ Y\ (r(X_1,Y) \wedge \neg\ s(Y))$

$p(X_1) \leftrightarrow (X_1=a \wedge \neg\ p(b)) \vee (X_1=b \wedge \neg\ p(a))$

$s(X_1) \leftrightarrow$ false

s(a) F	s(b) F	s(c) F	s(d) F
p(a)	p(b)	p(c)	p(d)
q(a)	q(b)	q(c)	q(d)
r(a,a)	r(a,b)	r(a,c)	r(a,d)
r(b,a)	r(b,b)	r(b,c)	r(b,d)
r(c,a)	r(c,b)	r(c,c)	r(c,d)
r(d,a)	r(d,b)	r(d,c)	r(d,d)

Studiamo s

Semantica dei Programmi Generali

Modelli di Herbrand del completamento

$r(X_1,X_2) \leftrightarrow (X_1=a \wedge X_2=c) \vee (X_1=a \wedge X_2=d)$

$q(X_1) \leftrightarrow \exists\ Y\ (r(X_1,Y) \wedge \neg\ s(Y))$

$p(X_1) \leftrightarrow (X_1=a \wedge \neg\ p(b)) \vee (X_1=b \wedge \neg\ p(a))$

$s(X_1) \leftrightarrow$ false

s(a) F	s(b) F	s(c) F	s(d) F
p(a)	p(b)	p(c)	p(d)
q(a)	q(b)	q(c)	q(d)
r(a,a) F	r(a,b) F	r(a,c) T	r(a,d) T
r(b,a) F	r(b,b) F	r(b,c) F	r(b,d) F
r(c,a) F	r(c,b) F	r(c,c) F	r(c,d) F
r(d,a) F	r(d,b) F	r(d,c) F	r(d,d) F

Studiamo r.

Semantica dei Programmi Generali

Modelli di Herbrand del completamento

$r(X_1,X_2) \leftrightarrow (X_1=a \wedge X_2=c) \vee (X_1=a \wedge X_2=d)$

$q(X_1) \leftrightarrow \exists\ Y\ (r(X_1,Y) \wedge \neg\ s(Y))$

$p(X_1) \leftrightarrow (X_1=a \wedge \neg\ p(b)) \vee (X_1=b \wedge \neg\ p(a))$

$s(X_1) \leftrightarrow$ false

s(a) F	s(b) F	s(c) F	s(d) F
p(a)	p(b)	p(c)	p(d)
q(a) T	q(b) F	q(c) F	q(d) F
r(a,a) F	r(a,b) F	r(a,c) T	r(a,d) T
r(b,a) F	r(b,b) F	r(b,c) F	r(b,d) F
r(c,a) F	r(c,b) F	r(c,c) F	r(c,d) F
r(d,a) F	r(d,b) F	r(d,c) F	r(d,d) F

Studiamo q.

Semantica dei Programmi Generali

Modelli di Herbrand del completamento

$r(X_1,X_2) \leftrightarrow (X_1=a \wedge X_2=c) \vee (X_1=a \wedge X_2=d)$
$q(X_1) \leftrightarrow \exists\ Y\ (r(X_1,Y) \wedge \neg\ s(Y))$
$p(X_1) \leftrightarrow (X_1=a \wedge \neg\ p(b)) \vee (X_1=b \wedge \neg\ p(a))$
$s(X_1) \leftrightarrow$ false

s(a) F	s(b) F	s(c) F	s(d) F
p(a) ?	p(b) ?	p(c) F	p(d) F
q(a) T	q(b) F	q(c) F	q(d) F
r(a,a) F	r(a,b) F	r(a,c) T	r(a,d) T
r(b,a) F	r(b,b) F	r(b,c) F	r(b,d) F
r(c,a) F	r(c,b) F	r(c,c) F	r(c,d) F
r(d,a) F	r(d,b) F	r(d,c) F	r(d,d) F

Studiamo p.

Semantica dei Programmi Generali

Modelli di Herbrand del completamento

- Si nota che vi sono tre insiemi di atomi:
  - Quelli veri in tutti i modelli di Herbrand del completamento.
  - Quelli falsi in tutti i modelli di Herbrand del completamento.
  - Quelli veri in alcuni e falsi in altri.
- Ciò suggerisce di determinare l'insieme dei I^+ sempre veri e quello I^- dei sempre falsi (semantica di Fitting).
- Tali insiemi possono essere calcolati (talvolta in tutto o talvolta in parte) usando la nozione di *modello well-founded*.
- Il modello well-founded è una coppia $\langle I^+, I^- \rangle$ ed è unico.
- In realtà, se $I^+ \cup I^- \neq B_P$ non è un vero modello !
- Se lo è sarà anche l'unico modello stabile (vedremo cos'è un modello stabile)

- Sia P ground (per semplicità)
- Una interpretazione è una coppia $\langle I^+, I^- \rangle$, con $I^+ \cap I^- = \emptyset$
- $\langle I^+, I^- \rangle$ è totale se $I^+ \cup I^- = B_P$
- Una regola

$$a \leftarrow b_1, \ldots, b_m, \texttt{not}\ c_1, \ldots, \texttt{not}\ c_n$$

  è soddisfatta da $\langle I^+, I^- \rangle$ se $a \in I^+$ oppure se $\{b_1, \ldots, b_m\} \not\subseteq I^+$ oppure se $\{c_1, \ldots, c_n\} \not\subseteq I^-$.
- Se $\langle I^+, I^- \rangle$ è totale e soddisfa tutte le regole di P allora è un modello.

SEMANTICA DEI PROGRAMMI GENERALI

MODELLO WELL-FOUNDED

- Data una interpretazione $\langle I^+, I^- \rangle$, un insieme $J \subseteq B_P$ è infondato rispetto a I se

$$\forall A \in J \forall (A \leftarrow \vec{B}) \in P(\langle I^+, I^- \rangle \not\models \vec{B} \vee J \cap \vec{B}^+ \neq \emptyset)$$

- $U_P(\langle I^+, I^- \rangle)$ è l'unione di tutti gli insiemi infondati.
- Mediante iterazione transfinita di T_P e U_P si ottiene un unico modello (in generale parziale) che viene detto modello well-founded. [Van Gelder et al., 1991]
- Vediamo un approccio più semplice [Zukowski, Brass, and Freitag, 1997]

SEMANTICA DEI PROGRAMMI GENERALI

MODELLO WELL-FOUNDED

- Date due interpretazioni (insiemi di atomi) I e J, definiamo l'operatore:

$$T_{P,J}(I) = \{a \in \mathcal{A} \mid (a \leftarrow bd^+, \texttt{not}\ \ bd^-) \in P, \\ I \models bd^+, (\forall p \in bd^-)(J \not\models p)\}$$

- Per calcolare il modello well-founded dobbiamo fare:

$$\begin{cases} K_0 = \mathrm{lfp}(T_{P^+,\emptyset}) & U_0 = \mathrm{lfp}(T_{P,K_0}) \\ K_i = \mathrm{lfp}(T_{P,U_{i-1}}) & U_i = \mathrm{lfp}(T_{P,K_i}) \quad i > 0 \end{cases}$$

  dove P^+ (usato per K_0) è l'insieme delle clausole definite di P.
- Se $(K_i, U_i) = (K_{i+1}, U_{i+1})$ ci fermiamo e il modello well-founded è: $I^+ = K_i, I^- = B_P \setminus U_i$
- Esempio ...

- La nozione di modello well-founded e quella che vedremo (meglio) di modello stabile sono fondate sulla nozione di programma ground
- Dato un programma generale P, il corrispondente programma ground $ground(P)$ è l'insieme di tutte le istanze ground su $\mathcal{H}_P$ delle clausole di P.

Semantica dei Programmi Generali

Grounding

$P =$

```
p(a).  p(b).  q(b).  q(c).
p(X,Y) :- p(X),q(Y).
```

$ground(P) =$

```
p(a).  p(b).  q(b).  q(c).
p(a,a) :- p(a),q(a).
p(a,b) :- p(a),q(b).
p(a,c) :- p(a),q(c).
p(b,a) :- p(c),q(a).
p(b,b) :- p(c),q(b).
p(b,c) :- p(c),q(c).
p(c,a) :- p(c),q(a).
p(c,b) :- p(c),q(b).
p(c,c) :- p(c),q(c).
```

Semantica dei Programmi Generali

Grounding

- Quant'è costoso il grounding?
- Assumiamo che H_P sia finito (e sia $|H_P| = c$).
- Siano $c_1, \ldots, c_n$ le clausole di P ($|P| = n$), e siano, rispettivamente, $\alpha_1, \ldots, \alpha_n$ i numeri delle variabili occorrenti in esse.
- Allora

$$|ground(P)| = \sum_{i=1}^{n} c^{\alpha_i}$$

- Se $k = \max_i\{\alpha_i\}$, abbiamo che

$$|ground(P)| \leq nc^k$$

- c può essere definito implicitamente (p.es. $p(1..100)$). In questi casi i numeri si sentono.

- Quant'è costoso il grounding?
- Assumiamo che H_P sia finito (e sia $|H_P| = c$).
- Siano $c_1, \ldots, c_n$ le clausole di P ($|P| = n$), e siano, rispettivamente, $\alpha_1, \ldots, \alpha_n$ i numeri delle variabili occorrenti in esse.
- Allora

$$|ground(P)| = \sum_{i=1}^{n} c^{\alpha_i}$$

- Se $k = \max_i\{\alpha_i\}$, abbiamo che

$$|ground(P)| \leq nc^k$$

- c può essere definito implicitamente (p.es. $p(1..100)$). In questi casi i numeri si sentono.

Semantica dei Programmi Generali

Grounding

- Quant'è costoso il grounding?
- Assumiamo che H_P sia finito (e sia $|H_P| = c$).
- Siano $c_1, \ldots, c_n$ le clausole di P ($|P| = n$), e siano, rispettivamente, $\alpha_1, \ldots, \alpha_n$ i numeri delle variabili occorrenti in esse.
- Allora

$$|ground(P)| = \sum_{i=1}^{n} c^{\alpha_i}$$

- Se $k = \max_i\{\alpha_i\}$, abbiamo che

$$|ground(P)| \leq nc^k$$

- c può essere definito implicitamente (p.es. $p(1..100)$). In questi casi i numeri si sentono.

- Quant'è costoso il grounding?
- Assumiamo che H_P sia finito (e sia $|H_P| = c$).
- Siano $c_1, \ldots, c_n$ le clausole di P ($|P| = n$), e siano, rispettivamente, $\alpha_1, \ldots, \alpha_n$ i numeri delle variabili occorrenti in esse.
- Allora

$$|ground(P)| = \sum_{i=1}^{n} c^{\alpha_i}$$

- Se $k = \max_i\{\alpha_i\}$, abbiamo che

$$|ground(P)| \leq nc^k$$

- c può essere definito implicitamente (p.es. $p(1..100)$). In questi casi i numeri si sentono.

SEMANTICA DEI PROGRAMMI GENERALI

MODELLI STABILI

- Come detto, consideriamo i programmi ground ($ground(P)$)
- I modelli stabili di un programma P vanno ricercati tra i sottoinsiemi *minimali* di $\mathcal{B}_P$ che siano modello di $ground(P)$.
- Un programma generale privo di negazioni (naf-literals) è un programma definito e quindi ha un unico modello minimale (il modello minimo): quello sarà anche il suo modello stabile.

- Se P è

```
p :- not q.
```

  Sappiamo che P ha due modelli minimali: $\{p\}$ e $\{q\}$.
- Tuttavia, osserviamo che il modello $\{q\}$ non riflette il significato intuitivo che attribuiamo al programma P (ovvero "se si dimostra che q è falso allora vale p"). Infatti in P non vi è nessuna informazione che fornisca una giustificazione a sostegno della verità di q.
- In questo caso dunque vorremmo come modello stabile $\{p\}$ (è anche well-founded).
- Se P invece è:

```
p :- not q.   q :- not p.
```

  well-founded dice $I^+ = \emptyset, I^- = \emptyset$

- Dato un programma generale P e un suo modello S, definiamo il programma P^S ridotto di P rispetto a S nel seguente modo:
  1. rimuovendo ogni regola il cui corpo contiene un naf-literal `not` L tale che $L \in S$;
  2. rimuovendo tutti i naf-literal dai corpi delle restanti regole.
- Per costruzione P^S è un programma definito. Se il suo modello minimo coincide con S, allora S è un modello stabile (anche detto answer set) per P.

Consideriamo il programma P

```
p :- a.
a :- not b.
b :- not a.
```

I candidati modelli stabili di P sono i sottoinsiemi di $\mathcal{B}_P = \{a, b, p\}$.

Consideriamo il programma P

```
p :- a.
a :- not b.
b :- not a.
```

I candidati modelli stabili di P sono i sottoinsiemi di $\mathcal{B}_P = \{a, b, p\}$.

$\emptyset$ Abbiamo che $P^\emptyset = \{p \leftarrow a.\ \ a.\ \ b.\}$ ma $\emptyset$ non è answer set per $P^\emptyset$. Quindi $\emptyset$ non è un answer set di P.

Consideriamo il programma P

```
p :- a.
a :- not b.
b :- not a.
```

I candidati modelli stabili di P sono i sottoinsiemi di $\mathcal{B}_P = \{a, b, p\}$.

$\{a\}$ Abbiamo che $P^{\{a\}} = \{p \leftarrow a.\ \ a.\}$ ma $\{a\}$ non è answer set per $P^{\{a\}}$. Quindi neanche $\{a\}$ è un answer set di P.

Consideriamo il programma P

```
p :- a.
a :- not b.
b :- not a.
```

I candidati modelli stabili di P sono i sottoinsiemi di $\mathcal{B}_P = \{a, b, p\}$.

$\{b\}$ Abbiamo che $P^{\{b\}} = \{p \leftarrow a.\ \ b.\}$ e $\{b\}$ è l'answer set di $P^{\{b\}}$ (ovvero, è il modello minimo). Quindi $\{b\}$ è un answer set di P.

Consideriamo il programma P

```
p :- a.
a :- not b.
b :- not a.
```

I candidati modelli stabili di P sono i sottoinsiemi di $\mathcal{B}_P = \{a, b, p\}$.

$\{p\}$ Abbiamo che $P^{\{p\}} = \{p \leftarrow a.\ \ a.\ \ b.\}$ ma $\{p\}$ non è answer set per $P^{\{p\}}$. Quindi neanche $\{p\}$ è un answer set di P.

Consideriamo il programma P

```
p :- a.
a :- not b.
b :- not a.
```

I candidati modelli stabili di P sono i sottoinsiemi di $\mathcal{B}_P = \{a, b, p\}$.

$\{p, a\}$ Abbiamo che $P^{\{p,a\}} = \{p \leftarrow a.\ \ a.\}$ e $\{p, a\}$ è l'answer set di $P^{\{p,a\}}$. Quindi $\{p, a\}$ è un answer set di P.

Consideriamo il programma P

```
p :- a.
a :- not b.
b :- not a.
```

I candidati modelli stabili di P sono i sottoinsiemi di $\mathcal{B}_P = \{a, b, p\}$.

$\{a, b\}$, $\{b, p\}$ e $\{a, b, p\}$, non sono answer set di P perchè includono propriamente un answer set (ad esempio $\{b\}$)

Consideriamo il programma P

```
p :- a.
a :- not b.
b :- not a.
```

I candidati modelli stabili di P sono i sottoinsiemi di $\mathcal{B}_P = \{a, b, p\}$.

Quindi P ha due answer set distinti: $\{b\}$ e $\{p, a\}$.

Consideriamo il programma P Consideriamo il programma P

```
a :- not b.
b :- not c.
d.
```

L'insieme $S_1 = \{b, d\}$ è un answer set di P. Infatti $P^{S_1} = \{b.\ \ d.\}$ che ha S_1 come answer set.
Invece l'insieme $S_2 = \{a, d\}$ non è un answer set di P. Infatti $P^{S_2} = \{a.\ \ b.\ \ d.\}$ che non ha S_2 come answer set.

Consideriamo il programma P

```
p :- not p,  d.
d.
```

Questo programma ammette il modello $\{p, d\}$. Ogni modello di P deve contenere d. Q uindi abbiamo due possibili candidati ad essere modello stabile:

- $S_1 = \{d\}$: allora $P^{S_1} = \{p \leftarrow d.\ \ d.\}$ ha come modello minimo $\{d, p\}$ che è diverso da S_1. Quindi S_1 non è answer set per P.
- $S_2 = \{d, p\}$: allora $P^{S_2} = \{\ d.\}$ ha come answer set $\{d\}$ che è diverso da S_2. Quindi S_2 non è answer set per P.

Questo Programma ha modelli, ma non ha modelli stabili.

Consideriamo il programma P

```
p :- not p,  d.
d.
```

Questo programma ammette il modello $\{p, d\}$. Ogni modello di P deve contenere d. Q uindi abbiamo due possibili candidati ad essere modello stabile:

- $S_1 = \{d\}$: allora $P^{S_1} = \{p \leftarrow d.\ \ d.\}$ ha come modello minimo $\{d, p\}$ che è diverso da S_1. Quindi S_1 non è answer set per P.
- $S_2 = \{d, p\}$: allora $P^{S_2} = \{\ d.\}$ ha come answer set $\{d\}$ che è diverso da S_2. Quindi S_2 non è answer set per P.

Questo Programma ha modelli, ma non ha modelli stabili.

SEMANTICA DEI PROGRAMMI GENERALI

"VINCOLI"

- Pensiamo al corpo di una regola ASP come ad una giustificazione per supportare la verità della testa della regola.
- L'idea intuitiva per decidere se un modello sia o meno un answer set è la seguente: *"Un qualsiasi p è presente nell'answer set solo se è forzato ad esserlo in quanto testa di una regola con corpo vero. Tuttavia, la verità della testa p di una regola non può essere giustificata in base alla verità di* `not` *p nel corpo."*
- Quindi la regola

```
p :- not p, d.
```

  non può supportare la verità di *p*.
- (potrebbe comunque essere supportata da una altra regola, se ci fosse)

Semantica dei Programmi Generali

"Vincoli"

- Dal punto di vista della ricerca di answer set, pertanto, se `p` non occorre altrove, la regola

  ```
  p :- not p, d.
  ```

  equivale ad imporre che *d* sia falso.
- Potremmo pertanto far ciò scrivendo il vincolo (goal)

  ```
  :- d
  ```

- Ammettiamo di usare vincoli di questo tipo. Sappiamo che sono uno zucchero sintattico.

Consideriamo il seguente programma

```
a :- not n_a.   n_a :- not a.
b :- not n_b.   n_b :- not b.
c :- not n_c.   n_c :- not c.
d :- not n_d.   n_d :- not d.
```

Gli answer set di questo programma rappresentano tutti i modi possibili di scegliere una e una sola alternativa

- tra `a` e `n_a`,
- tra `b` e `n_b`,
- tra `c` e `n_c`,
- tra `d` e `n_d`.

Theorem

Il problema di stabilire se un programma generale ground P ammette modelli stabili è NP-completo.

NP Dato P ground, un possibile modello stabile S conterrà necessariamente solo atomi presenti in P. Dunque $|S| \leq |P|$. Per verificare se S sia o meno modello stabile è sufficiente calcolare P^S e il modello minimo dello stesso. Entrambe le operazioni si effettuano in tempo polinomiale su $|P|$. Dunque il problema appartiene ad NP.

Theorem

Il problema di stabilire se un programma generale ground P ammette modelli stabili è NP-completo.

Hardness Consideriamo un'istanza φ di 3SAT:

$$\underbrace{(A \vee \neg B \vee C)}_{\texttt{c1}} \wedge \underbrace{(\neg A \vee B \vee \neg C)}_{\texttt{c2}}$$

e si costruisca il seguente programma P:

```
a :- not na.   na :- not a.
b :- not nb.   nb :- not b.
c :- not nc.   nc :- not c.
c1 :- a.   c1 :- nb.   c1 :- c.
c2 :- na.   c2 :- b.   c2 :- nc.
:- not c1.   :- not c2.
```

P ha modello stabile sse φ è soddisfacibile.

Semantica dei Programmi Generali

Riassunto

- P definito: unico modello minimo (che è anche well-founded e stabile). Se P ground finito si calcola in tempo polinomiale (no Turing)
- P generale. Ha sempre modello B_P, ma non è interessante.
- P generale. Ammette unico "modello" well-founded. Se P ground finito si calcola in tempo polinomiale (no Turing).
  Se è totale, allora è anche unico modello stabile.
- Se invece è parziale (cicli tra negazioni), sempre se P ground finito, stabilire l'esistenza di un modello stabile è NP-completo.
- Abbiamo un formalismo per il calcolo fatto apposta per i problemi in NP !!!!!!!

Semantica dei Programmi Generali

Come si calcolano in pratica i modelli stabili?

- Si usa un front-end (lparse, Gringo, DLV grounder) che *groundizza* il programma.
- Poi si chiama un ASP solver (smodels, cmodels, clasp, DLV)
- Esiste anche GASP!, un solver che rimanda il più possibile la fase di grounding, in ogni caso applicandola ad una clausola alla volta.