Logic Programming, Knowledge Representation, and Non Monotonic Reasoning

Agostino Dovier   Andrea Formisano

Department of Mathematics and Computer Science,
University of Udine, Italy

Udine, Fall 2011

SOMMARIO

I programmi definiti sono congiunzioni di regole del tipo:

$$A \leftarrow B_1, \ldots, B_m$$

con A, B_i atomi.
Se P è programma definito, ha un unico modello stabile, che è il suo modello minimo rispetto all'inclusione insiemistica tra atomi veri.

I programmi generali sono congiunzioni di regole del tipo:

$$A \leftarrow B_1, \ldots, B_m, \mathtt{not}\ \ C_1, \ldots, \mathtt{not}\ \ C_n$$

con A, B_i, C_j atomi.

I modelli stabili vanno ricercati tra i modelli di Herbrand minimali rispetto all'inclusione.

Precisamente, S è modello stabile di P sse S è modello minimo (e dunque stabile) di P^S ridotto di P rispetto a S, definito nel seguente modo:

1. rimuovendo ogni regola il cui corpo contiene un naf-literal $\mathtt{not}\ \ L$ tale che $L \in S$;
2. rimuovendo tutti i naf-literal dai corpi delle restanti regole.

I programmi generali sono congiunzioni di regole del tipo:

$$A \leftarrow B_1, \ldots, B_m, \texttt{not}\ \ C_1, \ldots, \texttt{not}\ \ C_n$$

con A, B_i, C_j atomi.
I modelli stabili vanno ricercati tra i modelli di Herbrand minimali rispetto all'inclusione.
Precisamente, S è modello stabile di P sse S è modello minimo (e dunque stabile) di P^S ridotto di P rispetto a S, definito nel seguente modo:

1. rimuovendo ogni regola il cui corpo contiene un naf-literal $\texttt{not}\ \ L$ tale che $L \in S$;
2. rimuovendo tutti i naf-literal dai corpi delle restanti regole.

I programmi generali sono congiunzioni di regole del tipo:

$$A \leftarrow B_1, \ldots, B_m, \texttt{not}\ \ C_1, \ldots, \texttt{not}\ \ C_n$$

con A, B_i, C_j atomi.
I modelli stabili vanno ricercati tra i modelli di Herbrand minimali rispetto all'inclusione.
Precisamente, S è modello stabile di P sse S è modello minimo (e dunque stabile) di P^S ridotto di P rispetto a S, definito nel seguente modo:

1. rimuovendo ogni regola il cui corpo contiene un naf-literal $\texttt{not}\ \ L$ tale che $L \in S$;
2. rimuovendo tutti i naf-literal dai corpi delle restanti regole.

I programmi disgiuntivi positivi senza `not` sono congiunzioni di regole del tipo:

$$A_1 \text{or} \cdots \text{or } A_m \leftarrow B_1, \ldots, B_n$$

con A_i, B_j atomi. m potrebbe anche essere 0, nel qual caso la regola si interpreta come un vincolo.
Se P è programma disgiuntivo positivo senza `not`, i modelli stabili di P sono tutti i modelli logici minimali rispetto all'inclusione (in cui or viene interpretato come $\vee$ e "," come $\wedge$).

```
vivo(X) or morto(X) :- uomo(X).
uomo(liga).
```

I modelli stabili sono: `{uomo(liga),vivo(liga)}` e `{uomo(liga),morto(liga)}`.

Il programma generale:

```
morto(X) :- uomo(X), not vivo(X).
uomo(liga).
```

logicamente è equivalente, ma il suo modello stabile è solo il secondo (non abbiamo giustificazioni per affermare la vita di `liga`).

```
vivo(X) or morto(X) :- uomo(X).
uomo(liga).
```

I modelli stabili sono: `{uomo(liga),vivo(liga)}` e `{uomo(liga),morto(liga)}`.

Il programma generale:

```
morto(X) :- uomo(X), not vivo(X).
uomo(liga).
```

logicamente è equivalente, ma il suo modello stabile è solo il secondo (non abbiamo giustificazioni per affermare la vita di `liga`).

```
a or b.
b or c :- a.
```

Scelgo `a` nella prima, dalla seconda ottengo `b` oppure `c`. Dunque sembrerebbe che `{a,b}` e `{a,c}` siano modelli stabili.

Scelgo `b` nella prima, `b` va bene anche per la seconda, dunque `{b}` sembra stabile.

Ma allora non lo era `{a,b}` (non sarebbe minimale).

```
a or b.
b or c :- a.
```

Scelgo `a` nella prima, dalla seconda ottengo `b` oppure `c`. Dunque sembrerebbe che `{a,b}` e `{a,c}` siano modelli stabili.
Scelgo `b` nella prima, `b` va bene anche per la seconda, dunque `{b}` sembra stabile.
Ma allora non lo era `{a,b}` (non sarebbe minimale).

```
a or b.
b or c :- a.
```

Scelgo `a` nella prima, dalla seconda ottengo `b` oppure `c`.
Dunque sembrerebbe che `{a,b}` e `{a,c}` siano modelli stabili.
Scelgo `b` nella prima, `b` va bene anche per la seconda, dunque `{b}` sembra stabile.
Ma allora non lo era `{a,b}` (non sarebbe minimale).

I programmi disgiuntivi generali sono congiunzioni di regole del tipo:

$$A_1 \text{or } \cdots \text{or } A_m \leftarrow B_1, \ldots, B_m, \texttt{not}\ \ C_1, \cdots, \texttt{not}\ \ C_n$$

con A_i, B_j, C_k atomi.
Se P è programma disgiuntivo, i modelli stabili di P vanno ricercati tra i modelli logici minimali rispetto all'inclusione. Precisamente, S è modello stabile di P sse S è modello minimale (e dunque stabile) di P^S ridotto di P rispetto a S, definito nel seguente modo:

1. rimuovendo ogni regola il cui corpo contiene un naf-literal $\texttt{not}\ \ L$ tale che $L \in S$;
2. rimuovendo tutti i naf-literal dai corpi delle restanti regole.

I programmi disgiuntivi generali sono congiunzioni di regole del tipo:

$$A_1 \text{or} \cdots \text{or}\ A_m \leftarrow B_1, \ldots, B_m, \texttt{not}\ \ C_1, \cdots, \texttt{not}\ \ C_n$$

con A_i, B_j, C_k atomi.
Se P è programma disgiuntivo, i modelli stabili di P vanno ricercati tra i modelli logici minimali rispetto all'inclusione.
Precisamente, S è modello stabile di P sse S è modello minimale (e dunque stabile) di P^S ridotto di P rispetto a S, definito nel seguente modo:

1. rimuovendo ogni regola il cui corpo contiene un naf-literal $\texttt{not}\ \ L$ tale che $L \in S$;
2. rimuovendo tutti i naf-literal dai corpi delle restanti regole.

I programmi disgiuntivi generali sono congiunzioni di regole del tipo:

$$A_1 \text{or} \ \cdots \text{or} \ A_m \leftarrow B_1, \ldots, B_m, \texttt{not} \ \ C_1, \cdots, \texttt{not} \ \ C_n$$

con A_i, B_j, C_k atomi.
Se P è programma disgiuntivo, i modelli stabili di P vanno ricercati tra i modelli logici minimali rispetto all'inclusione. Precisamente, S è modello stabile di P sse S è modello minimale (e dunque stabile) di P^S ridotto di P rispetto a S, definito nel seguente modo:

1. rimuovendo ogni regola il cui corpo contiene un naf-literal $\texttt{not} \ \ L$ tale che $L \in S$;
2. rimuovendo tutti i naf-literal dai corpi delle restanti regole.

CONSISTENCY PROBLEM

Dato un programma ground P, stabilire se ammette almeno un modello stabile.

- Un programma P definito ammette sempre il modello minimo M_P
- M_P è anche modello stabile per definizione
- Dunque ogni programma definito ammette sempre modello stabile.

Un problema/linguaggio L appartiene alla classe **NP** se, per ogni $x \in L$, esiste un certificato (guess) c per x che ci permette di dimostrare che $x \in L$ verify in tempo polinomiale.

Trovare il certificato opportuno può essere difficile, tipicamente visitando uno spazio esponenziale rispetto a $|x|$.

Un problema è **NP**-completo se ogni problema in **NP** può essere ridotto a lui. SAT (ovvero stabilire se una formula Booleana x è soddisfacibile) è **NP**-completo. Il certificato c per x è un assegnamento per le variabili in x.

Un problema/linguaggio L appartiene alla classe **NP** se, per ogni $x \in L$, esiste un certificato (guess) c per x che ci permette di dimostrare che $x \in L$ verify in tempo polinomiale.

Trovare il certificato opportuno può essere difficile, tipicamente visitando uno spazio esponenziale rispetto a $|x|$.

Un problema è **NP**-completo se ogni problema in **NP** può essere ridotto a lui. SAT (ovvero stabilire se una formula Booleana x è soddisfacibile) è **NP**-completo. Il certificato c per x è un assegnamento per le variabili in x.

Un problema/linguaggio L appartiene alla classe **NP** se, per ogni $x \in L$, esiste un certificato (guess) c per x che ci permette di dimostrare che $x \in L$ verify in tempo polinomiale.

Trovare il certificato opportuno può essere difficile, tipicamente visitando uno spazio esponenziale rispetto a $|x|$.

Un problema è **NP**-completo se ogni problema in **NP** può essere ridotto a lui. SAT (ovvero stabilire se una formula Booleana x è soddisfacibile) è **NP**-completo. Il certificato c per x è un assegnamento per le variabili in x.

THEOREM

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

NP Dato P ground, un certificato S, possibile modello stabile, conterrà 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:

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

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.
phi :- c1, c2.   :- not phi.
```

P ha modello stabile sse φ è soddisfacibile.

Theorem

Il problema di stabilire se un programma disgiuntivo senza `not` *P ammette modelli stabili è* **NP***-completo.*

NP. Dato P ground, un possibile modello stabile S conterrà solo atomi presenti in P. Dunque $|S| \leq |P|$. Verificare se S sia o meno modello si effettua in tempo polinomiale su $|P|$. S potrebbe non essere minimale, ma se non lo fosse, un altro modello minimale esisterebbe. Dunque il problema appartiene ad **NP**.

Si osservi che se non vi sono "vincoli" allora B_P è sempre modello e dunque esiste sempre un modello stabile come per i programmi definiti.

Theorem

Il problema di stabilire se un programma disgiuntivo senza `not` *P ammette modelli stabili è* **NP***-completo.*

Hardness Consideriamo un'istanza φ di 3SAT:

$$\underbrace{(A \vee \neg B \vee C)}_{\texttt{c1}} \wedge \underbrace{(\neg A \vee B \vee \neg C)}_{\texttt{c2}} \wedge \underbrace{(\neg A \vee \neg B)}_{\texttt{c3}}$$

e si costruisca il seguente programma *P*:

```
a or c :- b.   b :- a,c.   :- a,b.
```

E' immediato mostrare che *P* ha modello (e, dunque, anche un modello stabile) sse φ è soddisfacibile.

Parentesi

La gerarchia polinomiale

La gerarchia polinomiale delle classi di complessità viene definita in analogia con la gerarchia aritmetica di Kleene per la computabilità.

$$\Sigma_0^{\mathbf{P}} = \Pi_0^{\mathbf{P}} = \mathbf{P}$$

$$\Sigma_{k+1}^{\mathbf{P}} = \mathbf{NP}^{\Sigma_k^{\mathbf{P}}}, \Pi_{k+1}^{\mathbf{P}} = \text{co-}\Sigma_{k+1}^{\mathbf{P}}$$

Un linguaggio L appartiene a $\mathbf{NP}^C$ se esiste un algoritmo in grado di verificare l'appartenenza di x a L dato un certificato per x con un numero polinomiale di passi. In ogni passo si può chiedere aiuto ad un oracolo che risponde in tempo costante ad un test di appartenenza ad un linguaggio nella classe C.
Un linguaggio L appartiene a co-$\mathbf{NP}^C$ se esiste un algoritmo in grado di verificare la non appartenenza ad L con le stesse regole sopra.

Parentesi
La gerarchia polinomiale

- $\Sigma_1^{\mathbf{P}} = \mathbf{NP}^{\mathbf{P}} = \mathbf{NP}$. Tipico problema: SAT, ovvero stabilire se
$$\exists x_1 \cdots \exists x_n B[\vec{x}]$$
B formula Booleana sulle variabili $\vec{x}$.
- $\Pi_1^{\mathbf{P}} = \text{co-}\mathbf{NP}^{\mathbf{P}} = \text{co-}\mathbf{NP}$. Tipico problema: VALIDITY, ovvero data B, stabilire se
$$\forall x_1 \cdots \forall x_n B[\vec{x}]$$
- $\Sigma_2^{\mathbf{P}} = \mathbf{NP}^{\mathbf{NP}}$. Tipico problema: stabilire la validità di formule del tipo:
$$\exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m B[\vec{x}, \vec{y}]$$
- $\Pi_2^{\mathbf{P}} = \text{co-}\mathbf{NP}^{\mathbf{NP}}$. Tipico problema: stabilire la validità di formule del tipo:
$$\forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m B[\vec{x}, \vec{y}]$$

Theorem

Il problema di stabilire se un programma disgiuntivo generale ground P ammette modelli stabili è $\Sigma_2^{\mathbf{P}}$-completo.

Per mostrare che il problema appartiene a $\Sigma_2^{\mathbf{P}}$, dato un potenziale modello stabile S dovrei saper verificarlo in tempo polinomiale con chiamate ad un oracolo per un linguaggio nella classe **NP** che posso immaginare rispondere in tempo costante. In tempo polinomiale costruisco il ridotto P^S che è un programma senza `not` . Si deve ora verificare che S è un modello di P^S (polinomiale) e che è minimale, ovvero che non esiste $S' \subset S$ modello di P^S. Dire se esiste un tale S' è **NP**, dunque se l'oracolo risponde no, ho verificato la proprietà.

Consistency Problem

Programmi disgiuntivi generali

Theorem

Il problema di stabilire se un programma disgiuntivo generale ground P ammette modelli stabili è $\Sigma_2^{\mathbf{P}}$-completo.

Per la $\Sigma_2^{\mathbf{P}}$ hardness, si tratta di ridurre il problema di validità di una formula

$$\varphi \equiv \exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m B[\vec{x}, \vec{y}]$$

al problema dell'esistenza di un modello stabile per un programma costruito sulla base di φ.

Consistency Problem

Programmi disgiuntivi generali: Σ_2^P hardness

Si prenda la formula φ:

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

Introduciamo le variabili $x'_1, x'_2, x'_3, y'_1, y'_2, w$ e scriviamo il programma $P(\varphi)$:

$$\begin{array}{llll}
x_1 \vee x'_1 \leftarrow & x_2 \vee x'_2 \leftarrow & x_3 \vee x'_3 \leftarrow & \\
y_1 \vee y'_1 \leftarrow & y_2 \vee y'_2 \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y'_1 \leftarrow w. & y'_2 \leftarrow w. \\
w \leftarrow x_1, x'_2, y_1. & & & \\
w \leftarrow x'_1, x_3, y'_1, y_2. & & & \\
w \leftarrow \texttt{not}\ \ w. & & &
\end{array}$$

x'_i sta per $\neg x_i$. Anche y'_i sta intuitivamente per $\neg y_i$, anche se w può renderle vere entrambe.

Consistency Problem

Programmi disgiuntivi generali: Σ_2^P hardness

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. \\
w \leftarrow x_1, x_2', y_1. & & & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & & & (D_2) \\
w \leftarrow \texttt{not}\ w. & & &
\end{array}$$

Supponiamo che M sia modello stabile. L'ultima regola dice che w non può essere falso. Allora w deve essere supportato da una delle due regole (D_1) o (D_2) (si pensi a P^M) e tutti i y_i e y_i' sono veri in M.

Supponiamo ora di prendere un'interpretazione I che coincide con M su x_i, x_i', e che "sceglie", per ogni i, uno ed un solo valore tra y_i e y_i' e che non contiene w (ne ho a disposizione 4 in questo esempio).
I non può essere modello di P^M, altrimenti $M \supset I$ non sarebbe stabile.
Per non essere modello (logico) renderà vero almeno uno dei corpi di (D_1) e (D_2) (unico punto in possibile contraddizione).
Ma allora abbiamo mostrato che dati x_1, x_2, X_3, ogni valore assegnato agli $\vec{y}$ rende vera la matrice di φ. Dunque φ è valida.

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{ll}
x_1 \vee x_1' \leftarrow \quad x_2 \vee x_2' \leftarrow \quad x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow \quad y_2 \vee y_2' \leftarrow & \\
y_1 \leftarrow w. \quad y_2 \leftarrow w. \quad y_1' \leftarrow w. \quad y_2' \leftarrow w. & \\
w \leftarrow x_1, x_2', y_1. & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & (D_2) \\
w \leftarrow \texttt{not}\ w. &
\end{array}$$

Supponiamo che M sia modello stabile. L'ultima regola dice che w non può essere falso. Allora w deve essere supportato da una delle due regole (D_1) o (D_2) (si pensi a P^M) e tutti i y_i e y_i' sono veri in M.

Supponiamo ora di prendere un'interpretazione I che coincide con M su x_i, x_i', e che "sceglie", per ogni i, uno ed un solo valore tra y_i e y_i' e che non contiene w (ne ho a disposizione 4 in questo esempio).
I non può essere modello di P^M, altrimenti $M \supset I$ non sarebbe stabile.
Per non essere modello (logico) renderà vero almeno uno dei corpi di (D_1) e (D_2) (unico punto in possibile contraddizione).
Ma allora abbiamo mostrato che dati x_1, x_2, X_3, ogni valore assegnato agli $\vec{y}$ rende vera la matrice di φ. Dunque φ è valida.

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. \\
w \leftarrow x_1, x_2', y_1. & & & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & & & (D_2) \\
w \leftarrow \texttt{not}\ w. & & &
\end{array}$$

Supponiamo che M sia modello stabile. L'ultima regola dice che w non può essere falso. Allora w deve essere supportato da una delle due regole (D_1) o (D_2) (si pensi a P^M) e tutti i y_i e y_i' sono veri in M.

Supponiamo ora di prendere un'interpretazione I che coincide con M su x_i, x_i', e che "sceglie", per ogni i, uno ed un solo valore tra y_i e y_i' e che non contiene w (ne ho a disposizione 4 in questo esempio).
I non può essere modello di P^M, altrimenti $M \supset I$ non sarebbe stabile.
Per non essere modello (logico) renderà vero almeno uno dei corpi di (D_1) e (D_2) (unico punto in possibile contraddizione).
Ma allora abbiamo mostrato che dati x_1, x_2, X_3, ogni valore assegnato agli $\vec{y}$ rende vera la matrice di φ. Dunque φ è valida.

Consistency Problem

Programmi disgiuntivi generali: Σ_2^P hardness

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. \\
w \leftarrow x_1, x_2', y_1. & & & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & & & (D_2) \\
w \leftarrow \texttt{not}\ w. & & &
\end{array}$$

Supponiamo che M sia modello stabile. L'ultima regola dice che w non può essere falso. Allora w deve essere supportato da una delle due regole (D_1) o (D_2) (si pensi a P^M) e tutti i y_i e y_i' sono veri in M.

Supponiamo ora di prendere un'interpretazione I che coincide con M su x_i, x_i', e che "sceglie", per ogni i, uno ed un solo valore tra y_i e y_i' e che non contiene w (ne ho a disposizione 4 in questo esempio).

I non può essere modello di P^M, altrimenti $M \supset I$ non sarebbe stabile. Per non essere modello (logico) renderà vero almeno uno dei corpi di (D_1) e (D_2) (unico punto in possibile contraddizione).

Ma allora abbiamo mostrato che dati x_1, x_2, X_3, ogni valore assegnato agli $\vec{y}$ rende vera la matrice di φ. Dunque φ è valida.

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. \\
w \leftarrow x_1, x_2', y_1. & & & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & & & (D_2) \\
w \leftarrow \texttt{not}\ w. & & &
\end{array}$$

Supponiamo che M sia modello stabile. L'ultima regola dice che w non può essere falso. Allora w deve essere supportato da una delle due regole (D_1) o (D_2) (si pensi a P^M) e tutti i y_i e y_i' sono veri in M.

Supponiamo ora di prendere un'interpretazione I che coincide con M su x_i, x_i', e che "sceglie", per ogni i, uno ed un solo valore tra y_i e y_i' e che non contiene w (ne ho a disposizione 4 in questo esempio).
I non può essere modello di P^M, altrimenti $M \supset I$ non sarebbe stabile.
Per non essere modello (logico) renderà vero almeno uno dei corpi di (D_1) e (D_2) (unico punto in possibile contraddizione).
Ma allora abbiamo mostrato che dati x_1, x_2, X_3, ogni valore assegnato agli $\vec{y}$ rende vera la matrice di φ. Dunque φ è valida.

LPKRNMR

A. Dovier, A. Formisano

KR e complessità
Linguaggi
Consistency Problem
Programmi Definiti
Programmi Generali
Programmi disgiuntivi senza not
Programmi disgiuntivi generali
Decision Problem
Programmi Definiti
Programmi Generali
Programmi disgiuntivi senza not
Programmi disgiuntivi generali
Conclusioni

CONSISTENCY PROBLEM

PROGRAMMI DISGIUNTIVI GENERALI: Σ_2^P HARDNESS

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. \\
w \leftarrow x_1, x_2', y_1. & & & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & & & (D_2) \\
w \leftarrow \texttt{not}\ w. & & &
\end{array}$$

Supponiamo che M sia modello stabile. L'ultima regola dice che w non può essere falso. Allora w deve essere supportato da una delle due regole (D_1) o (D_2) (si pensi a P^M) e tutti i y_i e y_i' sono veri in M.

Supponiamo ora di prendere un'interpretazione I che coincide con M su x_i, x_i', e che "sceglie", per ogni i, uno ed un solo valore tra y_i e y_i' e che non contiene w (ne ho a disposizione 4 in questo esempio).
I non può essere modello di P^M, altrimenti $M \supset I$ non sarebbe stabile.
Per non essere modello (logico) renderà vero almeno uno dei corpi di (D_1) e (D_2) (unico punto in possibile contraddizione).
Ma allora abbiamo mostrato che dati x_1, x_2, X_3, ogni valore assegnato agli $\vec{y}$ rende vera la matrice di φ. Dunque φ è valida.

Consistency Problem

Programmi disgiuntivi generali: Σ_2^P hardness

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w.
\end{array}$$

$$w \leftarrow x_1, x_2', y_1. \qquad (D_1)$$

$$w \leftarrow x_1', x_3, y_1', y_2. \qquad (D_2)$$

$$w \leftarrow \texttt{not}\ w.$$

Supponiamo che M sia modello stabile. L'ultima regola dice che w non può essere falso. Allora w deve essere supportato da una delle due regole (D_1) o (D_2) (si pensi a P^M) e tutti i y_i e y_i' sono veri in M.

Supponiamo ora di prendere un'interpretazione I che coincide con M su x_i, x_i', e che "sceglie", per ogni i, uno ed un solo valore tra y_i e y_i' e che non contiene w (ne ho a disposizione 4 in questo esempio).
I non può essere modello di P^M, altrimenti $M \supset I$ non sarebbe stabile.
Per non essere modello (logico) renderà vero almeno uno dei corpi di (D_1) e (D_2) (unico punto in possibile contraddizione).
Ma allora abbiamo mostrato che dati x_1, x_2, X_3, ogni valore assegnato agli $\vec{y}$ rende vera la matrice di φ. Dunque φ è valida.

Consistency Problem

Programmi disgiuntivi generali: Σ_2^P hardness

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. \quad y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. & \\
w \leftarrow x_1, x_2', y_1. & & & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & & & (D_2) \\
w \leftarrow \texttt{not}\ w. & & &
\end{array}$$

Supponiamo che M sia modello stabile. L'ultima regola dice che w non può essere falso. Allora w deve essere supportato da una delle due regole (D_1) o (D_2) (si pensi a P^M) e tutti i y_i e y_i' sono veri in M.

Supponiamo ora di prendere un'interpretazione I che coincide con M su x_i, x_i', e che "sceglie", per ogni i, uno ed un solo valore tra y_i e y_i' e che non contiene w (ne ho a disposizione 4 in questo esempio).
I non può essere modello di P^M, altrimenti $M \supset I$ non sarebbe stabile.
Per non essere modello (logico) renderà vero almeno uno dei corpi di (D_1) e (D_2) (unico punto in possibile contraddizione).
Ma allora abbiamo mostrato che dati x_1, x_2, X_3, ogni valore assegnato agli $\vec{y}$ rende vera la matrice di φ. Dunque φ è valida.

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. \quad y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. & \\
w \leftarrow x_1, x_2', y_1. & & & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & & & (D_2) \\
w \leftarrow \texttt{not}\ w. & & &
\end{array}$$

Viceversa, se φ è valida, con un dato assegnamento per le variabili $\vec{x}$ è immediato trovare un modello stabile.

(esercizio . . . pensate al lucido precedente).

Consistency Problem

Programmi disgiuntivi generali: Σ_2^P hardness

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \wedge \neg x_2 \wedge y_1) \vee (\neg x_1 \wedge x_3 \wedge \neg y_1 \wedge y_2))$$

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. \\
w \leftarrow x_1, x_2', y_1. & & & (D_1) \\
w \leftarrow x_1', x_3, y_1', y_2. & & & (D_2) \\
w \leftarrow \texttt{not}\ w. & & &
\end{array}$$

Viceversa, se φ è valida, con un dato assegnamento per le variabili $\vec{x}$ è immediato trovare un modello stabile.
(esercizio ... pensate al lucido precedente).

Decision Problem

Dato un programma ground P, e un letterale L, il problema è quello di stabilire se $P \models_{sm} L$, ovvero che L è vero in *ogni* modello stabile di P.

(Se $L = A$ allora L è vero in S sse $A \in S$. Se $L = \neg A$ allora L è vero in S sse $A \notin S$)

Se P è inconsistent (o incoerente), allora $P \models_{sm} L$ per ogni letterale L (e per ogni altra formula)

Decision Problem

Programmi Definiti

Dato un programma definito ground P, e un atomo A, stabilire se $P \models_{sm} A$ equivale a dire se $A \in M_P$

Usando la T_P questo si verifica in tempo polinomiale.

In più, sappiamo che $A \in M_P$ sse $P \cup \{\neg A\}$ è insoddisfacibile. Ma la verifica di soddisfacibilità di una teoria di clausole di Horn (HORNSAT) è un problema lineare [Dowling and Gallier, JLP 1:267–284, 1984—riduzione ad un problema di cammini su grafo.]

Il problema $P \models_{sm} \neg A$ si riduce in questo caso a $A \notin M_P$: stesse considerazioni di sopra.

Decision Problem
Programmi Definiti

Dato un programma definito ground P, e un atomo A, stabilire se $P \models_{sm} A$ equivale a dire se $A \in M_P$

Usando la T_P questo si verifica in tempo polinomiale.

In più, sappiamo che $A \in M_P$ sse $P \cup \{\neg A\}$ è insoddisfacibile. Ma la verifica di soddisfacibilità di una teoria di clausole di Horn (HORNSAT) è un problema lineare [Dowling and Gallier, JLP 1:267–284, 1984—riduzione ad un problema di cammini su grafo.]

Il problema $P \models_{sm} \neg A$ si riduce in questo caso a $A \notin M_P$: stesse considerazioni di sopra.

Dato un programma definito ground P, e un atomo A, stabilire se $P \models_{sm} A$ equivale a dire se $A \in M_P$

Usando la T_P questo si verifica in tempo polinomiale.

In più, sappiamo che $A \in M_P$ sse $P \cup \{\neg A\}$ è insoddisfacibile. Ma la verifica di soddisfacibilità di una teoria di clausole di Horn (HORNSAT) è un problema lineare [Dowling and Gallier, JLP 1:267–284, 1984—riduzione ad un problema di cammini su grafo.]

Il problema $P \models_{sm} \neg A$ si riduce in questo caso a $A \notin M_P$: stesse considerazioni di sopra.

THEOREM

*Dato un programma generale ground P, e un atomo A, stabilire se $P \models_{sm} A$ è co-**NP** completo.*

co-NP Ragioniamo su $P \not\models_{sm} A$. Ciò accade quando esiste un modello stabile S di P tale che $A \notin S$.

Dato S (guess) verificare che S è modello stabile di P e $A \notin S$ è polinomiale in P (ed in S, ma S contiene un sottoinsieme degli atomi presenti in P).

Dunque stabilire se $P \not\models_{sm} A$ è **NP** e pertanto $P \models_{sm} A$ è co-**NP**.

Theorem

Dato un programma generale ground P, e un atomo A, stabilire se $P \models_{sm} A$ è co-**NP** *completo.*

Mostriamo che stabilire se $P \not\models_{sm} A$ è **NP hard**.
Uso la riduzione da SAT usata per la **NP**-completezza dell'esistenza del modello stabile.
Aggiungiamo: `nonphi :- not phi.`
Se P ha modelli stabili, in nessuno ci può essere `nonphi`:
$P \not\models_{sm}$ `nonphi`.
Se P non ha modelli stabili, allora tutti contengono `nonphi` (banalmente): $P \models_{sm}$ `nonphi` .
Dunque, per la riduzione già studiata, φ è soddisfacibile sse
$P \not\models_{sm}$ `nonphi`.

Theorem

*Dato un programma generale ground P, e un letterale negativo $\neg A$, stabilire se $P \models_{sm} \neg A$ è co-**NP** completo.*

Appartenenza: per dire che $P \not\models_{sm} \neg A$ dobbiamo mostrare che esiste un modello stabile S tale che $A \in S$. Dato S questa verifica si fa in tempo polinomiale.

Completezza: si pensi alla riduzione di prima. Aggiungiamo p. Se ci sono modelli stabili, questi contengono p. Dunque φ soddisfacibile implica esistenza di modelli stabili, perciò $P \not\models_{sm} \neg\mathrm{p}$. φ insoddisfacibile implica assenza di modelli stabili, perciò da P si deduce banalmente tutto, in particolare: $P \models_{sm} \neg\mathrm{p}$.

THEOREM

*Dato un programma generale ground P, e un letterale negativo $\neg A$, stabilire se $P \models_{sm} \neg A$ è co-**NP** completo.*

Appartenenza: per dire che $P \not\models_{sm} \neg A$ dobbiamo mostrare che esiste un modello stabile S tale che $A \in S$. Dato S questa verifica si fa in tempo polinomiale.

Completezza: si pensi alla riduzione di prima. Aggiungiamo `p.` Se ci sono modelli stabili, questi contengono `p`. Dunque φ soddisfacibile implica esistenza di modelli stabili, perciò $P \not\models_{sm} \neg \mathtt{p}$. φ insoddisfacibile implica assenza di modelli stabili, perciò da P si deduce banalmente tutto, in particolare: $P \models_{sm} \neg \mathtt{p}$.

Theorem

Sia P programma disgiuntivo senza `not` *e A atomo.*

1. *Stabilire se $P \models_{sm} A$ è co-NP completo.*
2. *Stabilire se $P \models_{sm} \neg A$ è Π_2^P completo.*

Si noti l'asimmetria rispetto ai programmi generali.

Per mostrare che $P \not\models_{sm} A$ verifico che un certificato S sia modello di P tale che $A \notin S$. S potrebbe non essere minimale, ma se $A \notin S$ allora A non apparterrà nemmeno al minimale.

THEOREM

Sia P programma disgiuntivo senza `not` *e A atomo.*

1. *Stabilire se $P \models_{sm} A$ è co-NP completo.*
2. *Stabilire se $P \models_{sm} \neg A$ è Π_2^P completo.*

Mostrare che $P \models_{sm} \neg A$ è più complicato. Ragioniamo al solito su $P \not\models_{sm} \neg A$. Significa che devo verificare se un certificato S è modello stabile di P e contiene A. Che sia modello e che $A \in S$ si verifica in tempo polinomiale. Il problema è che S potrebbe non essere minimale e dunque esistere $S' \subseteq S \setminus \{A\}$ modello (non necessariamente stabile, ma in caso ce n'è uno stabile incluso in lui e che dunque non contiene A). Verificare che esista un tale S' è proprietà **NP**.

Dunque, per mostrare che $P \models_{sm} \neg A$ (proprietà co-), analizzo un certificato (proprietà **NP**). Per dire che il certificato va bene uso un oracolo in co-**NP** (che è lo stesso di un oracolo in **NP**).

Theorem

Sia P programma disgiuntivo senza `not` *e A atomo.*

1. *Stabilire se $P \models_{sm} A$ è co-NP completo.*
2. *Stabilire se $P \models_{sm} \neg A$ è Π_2^P completo.*

Sorvolo la completezza (Si basa sulle stesse riduzioni viste prima).

THEOREM

Sia P programma disgiuntivo generale e L letterale. Stabilire se $P \models_{sm} L$ è $\Pi_2^{\mathbf{P}}$ completo.

Ragioniamo prima con $L = A$ positivo e concentriamoci su: $P \not\models_{sm} A$. Significa che deve esistere S è modello stabile di P e non contiene A. Dato S (guess), verifico che $A \notin S$ e dunque calcolo P^S in tempo polinomiale. Verifico che S sia modello di P^S in tempo polinomiale. Ma S deve essere minimale e dunque non deve esistere $S' \subset S$ modello di P^S. Dire se esiste $S' \subseteq S$ è proprietà **NP** e dunque ci serve un oracolo in co-**NP** (dunque in **NP**).

Si noti che non basta, in questo caso, dire che S è modello di P^S allora ce n'è uno stabile incluso in lui che non contiene A! E' proprio la stabilità si S che va mostrata. D'altro canto un $S' \subset S$ potrebbe non essere modello di $P^{S'}$.

Theorem

Sia P programma disgiuntivo generale e L letterale. Stabilire se $P \models_{sm} L$ è $\Pi_2^{\mathbf{P}}$ completo.

Con $L = \neg A$ il ragionamento è (in questo caso) analogo, ovvero, dato S che contiene A, devo dimostrare che non esiste $S' \subseteq S$ modello di P^S.

Theorem

Sia P programma disgiuntivo generale e L letterale. Stabilire se $P \models_{sm} L$ è $\Pi_2^{\mathbf{P}}$ completo.

La completezza deriva aggiungendo `p.` al programma usato per il teorema analogo per i programmi senza `not` .

Conclusioni

- Abbiamo studiato espressività di diverse classi di programmi logici rispetto alla stable model semantics.
- Vi sono alcune varianti per il decision problem (brave/credulous reasoning, cautious/skeptical reasoning) su cui ho sorvolato.
- Per ognuno dei linguaggi visti ci sono diversi solver. Clasp è di sicuro il più veloce per i programmi generali. Clasp è anche esteso per la disgiunzione ma lì se la gioca con DLV.