AUTOMATED REASONING

Agostino Dovier

Università di Udine
CLPLAB

Udine, January 2017

Sommario

A definite program is a set of rules:

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

where A, B_i are (positive) atoms.
If P is definite, it has a unique stable model, which is its minimum (w.r.t. $\subseteq$) Herbrand model.

A general program is a set of rules:

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

where A, B_i, C_j are atoms.

Stable models are looked for in the Herbrand models that are minimal (w.r.t. $\subseteq$)

Let us recall that S is a stable model of P if and only if it is the minimum Herbrand model of P^S (reduct of P w.r.t. S). P^S is obtained:

1. removing any rule whose body contains a naf-literal $\texttt{not}\ L$ s.t. $L \in S$;
2. removing any naf-literal from the bodies of the remaining rules.

A general program is a set of rules:

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

where A, B_i, C_j are atoms.
Stable models are looked for in the Herbrand models that are minimal (w.r.t. $\subseteq$)
Let us recall that S is a stable model of P if and only if it is the minimum Herbrand model of P^S (reduct of P w.r.t. S). P^S is obtained:

1. removing any rule whose body contains a naf-literal $\texttt{not}\ L$ s.t. $L \in S$;
2. removing any naf-literal from the bodies of the remaining rules.

A general program is a set of rules:

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

where A, B_i, C_j are atoms.
Stable models are looked for in the Herbrand models that are minimal (w.r.t. $\subseteq$)
Let us recall that S is a stable model of P if and only if it is the minimum Herbrand model of P^S (reduct of P w.r.t. S). P^S is obtained:

1. removing any rule whose body contains a naf-literal $\texttt{not}\ L$ s.t. $L \in S$;
2. removing any naf-literal from the bodies of the remaining rules.

Disjunctive programs without `not` are conjunctions of rules:

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

where A_i, B_j are (positive) atoms.
If $m = 0$ the rule is interpreted as a constraint.

If P is a Disjunctive programs without `not` , its stable models are all its minimal (w.r.t. $\subseteq$) Herbrand models
(of course, or is interpreted as $\vee$, while "," is interpreted as $\wedge$).

LANGUAGES

DISJUNCTIVE PROGRAMS WITHOUT not

```
living(X) or dead(X) :- man(X).
man(lazzarus).
```

Its Herbrand models are:

1. {man(lazzarus),living(lazzarus)}
2. {man(lazzarus),dead(lazzarus)}
3. {man(lazzarus),dead(lazzarus),living(lazzarus) }.

The latter is not minimal.

The general program:

```
dead(X) :- man(X), not living(X).
man(lazzarus).
```

is “logically” equivalent, but its unique stable model is only the second one (we have no justification for stating that lazzarus is living).

```
living(X) or dead(X) :- man(X).
man(lazzarus).
```

Its Herbrand models are:

1. {man(lazzarus),living(lazzarus)}
2. {man(lazzarus),dead(lazzarus)}
3. {man(lazzarus),dead(lazzarus),living(lazzarus) }.

The latter is not minimal.

The general program:

```
dead(X) :- man(X), not living(X).
man(lazzarus).
```

is “logically” equivalent, but its unique stable model is only the second one (we have no justification for stating that lazzarus is living).

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

Choose a in the first rule. Then b or c can be chosen from the second rule. It seems that {a,b} and {a,c} are stable models.

Choose b in the first rule, the second rule is true since the body is false, thus {b} seems a minimimal model, hence stable.

Therefore {a,b} is not stable!

Similarly, {b,c} is not stable (not minimal).

Languages

Disjunctive programs without `not`

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

Choose `a` in the first rule. Then `b` or `c` can be chosen from the second rule. It seems that {`a`,`b`} and {`a`,`c`} are stable models.
Choose `b` in the first rule, the second rule is true since the body is false, thus {`b`} seems a minimimal model, hence stable.
Therefore {`a`,`b`} is not stable!
Similarly, {`b`,`c`} is not stable (not minimal).

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

Choose a in the first rule. Then b or c can be chosen from the second rule. It seems that {a,b} and {a,c} are stable models.
Choose b in the first rule, the second rule is true since the body is false, thus {b} seems a minimimal model, hence stable.
Therefore {a,b} is not stable!
Similarly, {b,c} is not stable (not minimal).

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

Choose a in the first rule. Then b or c can be chosen from the second rule. It seems that {a,b} and {a,c} are stable models.
Choose b in the first rule, the second rule is true since the body is false, thus {b} seems a minimimal model, hence stable.
Therefore {a,b} is not stable!
Similarly, {b,c} is not stable (not minimal).

Consider a graph *G*

```
node(a). node(b). node(c). ....
edge(a,b). edge(a,c). ....
```

and a set of (three) colors RGB

```
color(red). color(green). color(blue).
```

Let us model the 3-coloring problem as follows:

```
colored(X,red) or  colored(X,green) or colored(X,blue) :-  node(X).
:- color(C), edge(A,B), colored(A,C), colored(B,C).
```

Its stable models (if any) are the solutions to the 3-coloring problem.
(So What?)

Consider a graph *G*

```
node(a). node(b). node(c). ....
edge(a,b). edge(a,c). ....
```

and a set of (three) colors RGB

```
color(red). color(green). color(blue).
```

Let us model the 3-coloring problem as follows:

```
colored(X,red) or  colored(X,green) or colored(X,blue) :-  node(X).
:- color(C), edge(A,B), colored(A,C), colored(B,C).
```

Its stable models (if any) are the solutions to the 3-coloring problem.
(So What?)

Languages

Disjunctive programs without `not`

Consider a graph *G*

```
node(a). node(b). node(c). ....
edge(a,b). edge(a,c). ....
```

and a set of (three) colors RGB

```
color(red). color(green). color(blue).
```

Let us model the 3-coloring problem as follows:

```
colored(X,red) or  colored(X,green) or colored(X,blue) :-  node(X).
:- color(C), edge(A,B), colored(A,C), colored(B,C).
```

Its stable models (if any) are the solutions to the 3-coloring problem.
(So What?)

Consider a graph *G*

```
node(a). node(b). node(c). ....
edge(a,b). edge(a,c). ....
```

and a set of (three) colors RGB

```
color(red). color(green). color(blue).
```

Let us model the 3-coloring problem as follows:

```
colored(X,red) or  colored(X,green) or colored(X,blue) :-  node(X).
:- color(C), edge(A,B), colored(A,C), colored(B,C).
```

Its stable models (if any) are the solutions to the 3-coloring problem. (So What?)

General disjunctive programs are sets of rules:

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

where A_i, B_j, C_k are atoms.

If P is a general disjunctive program, its stable models should be looked for in the minimal (w.r.t. $\subseteq$) Herbrand models.

S is a stable model of P if and only if S is a minimal model of P^S reduct of P w.r.t. S, defined as follows:

1. removing any rule whose body contains a naf-literal $\texttt{not } L$ s.t. $L \in S$;
2. removing any naf-literal from the bodies of the remaining rules.

General disjunctive programs are sets of rules:

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

where A_i, B_j, C_k are atoms.
If P is a general disjunctive program, its stable models should be looked for in the minimal (w.r.t. $\subseteq$) Herbrand models.
S is a stable model of P if and only if S is a minimal model of P^S reduct of P w.r.t. S, defined as follows:

1. removing any rule whose body contains a naf-literal $\texttt{not}\ L$ s.t. $L \in S$;
2. removing any naf-literal from the bodies of the remaining rules.

General disjunctive programs are sets of rules:

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

where A_i, B_j, C_k are atoms.
If P is a general disjunctive program, its stable models should be looked for in the minimal (w.r.t. $\subseteq$) Herbrand models.
S is a stable model of P if and only if S is a minimal model of P^S reduct of P w.r.t. S, defined as follows:

1. removing any rule whose body contains a naf-literal $\texttt{not } L$ s.t. $L \in S$;
2. removing any naf-literal from the bodies of the remaining rules.

$C = \{c_1, \dots, c_m\}$ ($m \geq 1$) is a set of companies; each company produces some goods in a set G, and each company $c_i \in C$ is possibly controlled by a set of owner companies $O_i \subseteq C$.
A set $S \subseteq C$ is a strategic set iff it is a minimal set satisfying:

- companies in S produce all goods in G
- If $O_i \subseteq S$ then $c_i \in S$

In the instances proposed in the ASP competition, moreover, each product is produced by at most four companies and each company is controlled by at most four companies. Even with these restriction the problem is NP^{NP} complete.

Instances:

- `producedBy`(p, c_1, c_2, c_3, c_4) if product p is produced by companies c_1, c_2, c_3, c_4 Repetitions are used if less than 4 companies produce p.
- `controlledBy`(c, c_1, c_2, c_3, c_4) if a company c is controlled by companies c_1, c_2, c_3, c_4. Repetitions are used if less than 4 companies control c.
- A fact `strategic_pair`(c_i, c_j) forces to look for a a strategic S such that $\{c_i, c_j\} \subseteq S$.

```
strategic(W) :-   controlled_by(W,X1,X2,X3,X4),
        strategic(X1), strategic(X2), strategic(X3), strategic(X4).
:- strategic_pair(X,Y), not strategic(X).
:- strategic_pair(X,Y), not strategic(Y).
% DISJUNCTIVE RULE:
strategic(X1) or strategic(X2) or strategic(X3) or strategic(X4) :-
        produced_by(X,X1,X2,X3,X4).
```

Given a ground program P establish whether there is (or not) a stable model for P.

- We have seen that a definite program P admits always a unique minimum Herbrand model M_P
- M_P is also its stable model.
- Thus the answer is simply yes.
- Trivial problem (computing M_P is another problem).

BACKGROUND

THE CLASS **NP**

A problem/language L belongs to the class **NP** if, for each $x \in L$, there exists a succint certificate (guess) c for x that allows us to prove that $x \in L$ (verify) in polynomial time.

Finding the certificate is typically the hard task. For NP complete problems (currently) we need to visit a search space of exponential size w.r.t. $|x|$.

A problem L in **NP** is **NP**-complete if any problem in **NP** can be reduced to L. SAT is **NP**-complete. The certificate c for x is the Boolean assignment for the variables.

BACKGROUND
THE CLASS **NP**

A problem/language L belongs to the class **NP** if, for each $x \in L$, there exists a succint certificate (guess) c for x that allows us to prove that $x \in L$ (verify) in polynomial time.

Finding the certificate is typically the hard task. For NP complete problems (currently) we need to visit a search space of exponential size w.r.t. $|x|$.

A problem L in **NP** is **NP**-complete if any problem in **NP** can be reduced to L. SAT is **NP**-complete. The certificate c for x is the Boolean assignment for the variables.

A problem/language L belongs to the class **NP** if, for each $x \in L$, there exists a succint certificate (guess) c for x that allows us to prove that $x \in L$ (verify) in polynomial time.

Finding the certificate is typically the hard task. For NP complete problems (currently) we need to visit a search space of exponential size w.r.t. $|x|$.

A problem L in **NP** is **NP**-complete if any problem in **NP** can be reduced to L. SAT is **NP**-complete. The certificate c for x is the Boolean assignment for the variables.

Theorem

The problem of establishing whether a general ground program admits a stable model is **NP***-complete.*

NP Let P ground program, a candidate stable model S will contain only atoms occurring in P, thus $|S| \leq |P|$. Computing P^S, the fixpoint computation of M_{P^S}, and checking if $S = M_{P^S}$ therefore polynomial w.r.t. $|P|$. Thus the problem is in NP.

Theorem

The problem of establishing whether a general ground program admits a stable model is **NP***-complete.*

Hardness Let us consider an instance φ of 3SAT:

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

and define accordingly the program P_φ:

```
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_φ can be computed in LOGSPACE and it is immediate to check that it admits a stable model iff φ is satisfiable.

Theorem

The problem of establishing whether a disjunctive programs without `not` *P admits a stable model is* **NP***-complete.*

NP. Given P ground, a candidate stable model S will contain only atoms occurring in P. Thus, $|S| \leq |P|$.
Checking whether S is or not a model of P can be made in polynomial time on P. S could be not minimal. However, if this is the case, another minimal model would exist. Thus, checking S can be made in polynomial time (hence the problem is in **NP**).

Let us observe that in absence of constraints B_P is always a model and therefore a minimal model would exist always.

Theorem

The problem of establishing whether a disjunctive programs without `not` *P admits a stable model is* **NP***-complete.*

Hardness Let us consider an instance φ of 3SAT:

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

Let us define the program P:

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

It is immediate to see that P has a model (and, therefore it has a minimal model) iff φ is satisfiable.

The polynomial hierarchy (for complexity) is defined in a similar way as Kleene defined the arithmetical hierarchy (for computability).

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

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

A problem/language L belongs to $\mathbf{NP}^C$ if there is an algorithm such that for all x, it allows to prove that $x \in L$ (verify) in a polynomial number of steps. At every step the algorithm can require the help of a oracle that answers in constant time a membership check in the class C.
A problem/language L belongs to co-$\mathbf{NP}^C$ if there is an algorithm capable of verifying the non memberhip to L with the same rules above.

- $B[\vec{v}]$ denotes a Boolean formula (without quantifiers) on the variables $\vec{v} = v_1, \ldots, v_k$
- $\Sigma_1^{\mathbf{P}} = \mathbf{NP}^{\mathbf{P}} = \mathbf{NP}$. Typical problem: SAT, namely establishing whether
$$\exists x_1 \cdots \exists x_n B[\vec{x}]$$
- $\Pi_1^{\mathbf{P}} = \text{co-}\mathbf{NP}^{\mathbf{P}} = \text{co-}\mathbf{NP}$. Typical problem: VALIDITY, namely establishing whether
$$\forall x_1 \cdots \forall x_n B[\vec{x}]$$
- $\Sigma_2^{\mathbf{P}} = \mathbf{NP}^{\mathbf{NP}}$. Typical problem: establishing whether
$$\exists x_1 \cdots \exists x_n \forall y_1 \cdots \forall y_m B[\vec{x}, \vec{y}]$$
- $\Pi_2^{\mathbf{P}} = \text{co-}\mathbf{NP}^{\mathbf{NP}}$. Typical problem: establishing whether
$$\forall x_1 \cdots \forall x_n \exists y_1 \cdots \exists y_m B[\vec{x}, \vec{y}]$$

THEOREM

The problem of establishing whether a general disjunctive program admits a stable model is $\Sigma_2^{\mathbf{P}}$-complete.

We first show that the problem belongs to $\Sigma_2^{\mathbf{P}}$.
Given a candidate stable model S we should be able to check it in a polynomial number of steps possibly querying an oracle in **NP**.
The reduct P^S is obtained in polynomial time and it is a disjunctive program without `not` .
Now we need to check if S is a model of P^S (again, polynomial). It remains to check if it is minimal, namely that does not exist $S' \subset S$ that is a model of P^S.
Stating whether such a S' exists is **NP** (if you have it, you can verify it easily), thus, if the oracle answers no, we have checked that S is a stable model with a polynomial number of steps.

THEOREM

The problem of establishing whether a general disjunctive program admits a stable model is $\Sigma_2^{\mathbf{P}}$-complete.

For the $\Sigma_2^{\mathbf{P}}$ hardness, we reduce the problem of validity of

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

to the problem of existence of a stable model for a program obtained from φ.

To fix the ideas, let us consider φ:

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

Let us introduce variables $x_1', x_2', x_3', y_1', y_2', w$ and let us define $P(\varphi)$:

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

x_i' stands for $\neg x_i$.
y_i' stands intuitively for $\neg y_i$; in this case w can force both of them to be true.

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

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

Assume S is a stable model of $P(\varphi)$. The last rule ensures that w cannot be false. Thus, w must be supported by one of the rules (D_1), (D_2) (think to $P(\varphi)^S$) and therefore all y_i and y_i' are true in S for $i = 1$ and $i = 2$.

Consider now an interpretation (set of atoms) I that coincides with S on x_j, x_j' and such that, for every i choose one and only one in y_i and y_i' and that does not contain w (there are 4 of them in this example).

I is not a model of $P(\varphi)^S$, otherwise, since $S \supset I$, S would not be stable contradicting the hypothesis.

For being not a model (by case analysis) I makes true (at least) one of the bodies of (D_1) and (D_2).

Therefore, we have proved that there are x_1, x_2, x_3 such that any value assigned to $\vec{y}$ makes true the Boolean disjunction of φ. Thus φ is valid.

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

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

Assume S is a stable model of $P(\varphi)$. The last rule ensures that w cannot be false. Thus, w must be supported by one of the rules $(D_1), (D_2)$ (think to $P(\varphi)^S$) and therefore all y_i and y_i' are true in S for $i = 1$ and $i = 2$.

Consider now an interpretation (set of atoms) I that coincides with S on x_j, x_j' and such that, for every i choose one and only one in y_i and y_i' and that does not contain w (there are 4 of them in this example).

I is not a model of $P(\varphi)^S$, otherwise, since $S \supset I$, S would not be stable contradicting the hypothesis.

For being not a model (by case analysis) I makes true (at least) one of the bodies of (D_1) and (D_2).

Therefore, we have proved that there are x_1, x_2, x_3 such that any value assigned to $\vec{y}$ makes true the Boolean disjunction of φ. Thus φ is valid.

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

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

Assume S is a stable model of $P(\varphi)$. The last rule ensures that w cannot be false. Thus, w must be supported by one of the rules (D_1), (D_2) (think to $P(\varphi)^S$) and therefore all y_i and y_i' are true in S for $i = 1$ and $i = 2$.

Consider now an interpretation (set of atoms) I that coincides with S on x_j, x_j' and such that, for every i choose one and only one in y_i and y_i' and that does not contain w (there are 4 of them in this example).

I is not a model of $P(\varphi)^S$, otherwise, since $S \supset I$, S would not be stable contradicting the hypothesis.

For being not a model (by case analysis) I makes true (at least) one of the bodies of (D_1) and (D_2).

Therefore, we have proved that there are x_1, x_2, x_3 such that any value assigned to $\vec{y}$ makes true the Boolean disjunction of φ. Thus φ is valid.

Consistency Problem

Programmi disgiuntivi generali: Σ_2^{P} hardness

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

$$\begin{array}{llll}
x_1 \vee x_1' \leftarrow & x_2 \vee x_2' \leftarrow & x_3 \vee x_3' \leftarrow & \\
y_1 \vee y_1' \leftarrow & y_2 \vee y_2' \leftarrow & & \\
y_1 \leftarrow w. & y_2 \leftarrow w. & y_1' \leftarrow w. & y_2' \leftarrow w. \\
\multicolumn{4}{l}{w \leftarrow x_1, x_2', y_1. \qquad (D_1)} \\
\multicolumn{4}{l}{w \leftarrow x_1', x_3, y_1', y_2. \qquad (D_2)} \\
\multicolumn{4}{l}{w \leftarrow \texttt{not } w.}
\end{array}$$

Assume S is a stable model of $P(\varphi)$. The last rule ensures that w cannot be false. Thus, w must be supported by one of the rules $(D_1), (D_2)$ (think to $P(\varphi)^S$) and therefore all y_i and y_i' are true in S for $i = 1$ and $i = 2$.

Consider now an interpretation (set of atoms) I that coincides with S on x_j, x_j' and such that, for every i choose one and only one in y_i and y_i' and that does not contain w (there are 4 of them in this example).

I is not a model of $P(\varphi)^S$, otherwise, since $S \supset I$, S would not be stable contradicting the hypothesis.

For being not a model (by case analysis) I makes true (at least) one of the bodies of (D_1) and (D_2).

Therefore, we have proved that there are x_1, x_2, x_3 such that any value assigned to $\vec{y}$ makes true the Boolean disjunction of φ. Thus φ is valid.

Consistency Problem

Programmi disgiuntivi generali: Σ_2^P hardness

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

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

Assume S is a stable model of $P(\varphi)$. The last rule ensures that w cannot be false. Thus, w must be supported by one of the rules $(D_1), (D_2)$ (think to $P(\varphi)^S$) and therefore all y_i and y_i' are true in S for $i = 1$ and $i = 2$.
Consider now an interpretation (set of atoms) I that coincides with S on x_j, x_j' and such that, for every i choose one and only one in y_i and y_i' and that does not contain w (there are 4 of them in this example).
I is not a model of $P(\varphi)^S$, otherwise, since $S \supset I$, S would not be stable contradicting the hypothesis.
For being not a model (by case analysis) I makes true (at least) one of the bodies of (D_1) and (D_2).

Therefore, we have proved that there are x_1, x_2, x_3 such that any value assigned to $\vec{y}$ makes true the Boolean disjunction of φ. Thus φ is valid.

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

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

Assume S is a stable model of $P(\varphi)$. The last rule ensures that w cannot be false. Thus, w must be supported by one of the rules (D_1), (D_2) (think to $P(\varphi)^S$) and therefore all y_i and y_i' are true in S for $i = 1$ and $i = 2$.

Consider now an interpretation (set of atoms) I that coincides with S on x_j, x_j' and such that, for every i choose one and only one in y_i and y_i' and that does not contain w (there are 4 of them in this example).

I is not a model of $P(\varphi)^S$, otherwise, since $S \supset I$, S would not be stable contradicting the hypothesis.

For being not a model (by case analysis) I makes true (at least) one of the bodies of (D_1) and (D_2).

Therefore, we have proved that there are x_1, x_2, x_3 such that any value assigned to $\vec{y}$ makes true the Boolean disjunction of φ. Thus φ is valid.

Consistency Problem

Programmi disgiuntivi generali: $\Sigma_2^{\mathbf{P}}$ hardness

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

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

Assume S is a stable model of $P(\varphi)$. The last rule ensures that w cannot be false. Thus, w must be supported by one of the rules $(D_1), (D_2)$ (think to $P(\varphi)^S$) and therefore all y_i and y_i' are true in S for $i = 1$ and $i = 2$.
Consider now an interpretation (set of atoms) I that coincides with S on x_j, x_j' and such that, for every i choose one and only one in y_i and y_i' and that does not contain w (there are 4 of them in this example).
I is not a model of $P(\varphi)^S$, otherwise, since $S \supset I$, S would not be stable contradicting the hypothesis.
For being not a model (by case analysis) I makes true (at least) one of the bodies of (D_1) and (D_2).

Therefore, we have proved that there are x_1, x_2, x_3 such that any value assigned to $\vec{y}$ makes true the Boolean disjunction of φ. Thus φ is valid.

$$\exists x_1 \exists x_2 \exists x_3 \forall y_1 \forall y_2 ((x_1 \land \neg x_2 \land y_1) \lor (\neg x_1 \land x_3 \land \neg y_1 \land y_2))$$

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

Assume S is a stable model of $P(\varphi)$. The last rule ensures that w cannot be false. Thus, w must be supported by one of the rules $(D_1), (D_2)$ (think to $P(\varphi)^S$) and therefore all y_i and y_i' are true in S for $i = 1$ and $i = 2$.

Consider now an interpretation (set of atoms) I that coincides with S on x_j, x_j' and such that, for every i choose one and only one in y_i and y_i' and that does not contain w (there are 4 of them in this example).

I is not a model of $P(\varphi)^S$, otherwise, since $S \supset I$, S would not be stable contradicting the hypothesis.

For being not a model (by case analysis) I makes true (at least) one of the bodies of (D_1) and (D_2).

Therefore, we have proved that there are x_1, x_2, x_3 such that any value assigned to $\vec{y}$ makes true the Boolean disjunction of φ. Thus φ is valid.

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

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

Conversely, if φ is valid, given an assignment for $\vec{x}$ it is immediate to find a stable model (complete the details looking at the previous slide).

CONCLUSIONS

- We have seen 4 families of logic programs for KR
- Each of them allows different expressivity and allows to deal with problems at different complexity
- All these languages are correctly interpreted by modern ASP solvers (DLV and clingo)
- Other families of logics have been used in AI. E.g., modal/temporal logics (subject of the course on verification) and description logics (foundation of Semantic Web).
- For the sake of completeness in the slides you can find the results of the four languages seen today on a different problem: the *Decision Problem*. [Remaining slides are in Italian but you can find the material in Eiter and Gottlob. On the Computational Cost of Disjunctive Logic Programming: Propositional Case. Annals of Mathematics and Artificial Intelligence, 15(3/4):289-323, 1995]

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.

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 $\mathrm{p}.$ 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`.