Abstract Answer Set Solver

Todolist

- Print the rules of Fig 1.

- An abstract framework for describing algorithms to find answer sets of a logic program using “constraint propagation”, backjumping, learning and forgetting.

Outline

- Notations
- Abstract Answer Set Solver
  - A first definition of graph associted with a program
  - An extended graph (catering for backjump)
  - Answer set solver
- Appendix
  - Generate reasons (in extended records)
  - Generate backjump clause

Abstract Answer Set Solver

- *States* and *transition rules* on *states* will be used, instead of pseudo-code, to describe ASP algorithms employing propagation, backjumping, learning and forgetting

States

- State: M||T or *FailState*
  - M is a record: A **record** relative to a program P is *a list of literals* over atoms of P without *repetitions* where each literal has an *annotation*, a bit that marks it as a **decision** literal or not.
  - T is a (multi-)set of denials

- Example

$FailState$	$\emptyset \| \| \emptyset$	$a\ b \| \| \emptyset$	$\neg a^{\Delta} \| \| \emptyset$	$\neg a\ a \| \| \emptyset$
	$\emptyset \| \| \leftarrow b$	$a\ b \| \| \leftarrow b$	$\neg a^{\Delta} \| \| \leftarrow b$	$\neg a\ a \| \| \leftarrow b$
	$\emptyset \| \| \leftarrow b,\ \leftarrow b$	$a\ b \| \| \leftarrow b,\ \leftarrow b$	$\neg a^{\Delta} \| \| \leftarrow b,\ \leftarrow b$	$\neg a\ a \| \| \leftarrow b,\ \leftarrow b$

Record

- Record
  - By ignoring the annotations and ordering, a record M can be taken as a set of literals, i.e., a "partial assignment"
  - *l* is *unassigned* if neither *l* nor its complemet is in M
  - A *decision* literal: supscripted with \Delta
  - Non-decision literal: no supscription

Transition rules

- Example

Unit Propagate:

$$M||\Gamma \implies M\ a||\Gamma \quad \text{if} \quad \begin{cases} a \leftarrow B \in \Pi \ \text{ and} \\ B \subseteq M \end{cases}$$

Unfounded:

$$M||\Gamma \implies M\ \neg a||\Gamma \quad \text{if} \begin{cases} M \text{ is consistent and} \\ a \in U \text{ for a set } U \text{ unfounded on } M \text{ w.r.t. } \Pi \end{cases}$$

Graph associated to a program

- For any program P, we define a graph G_P whose
  - Nodes are the states of P
  - Edges are “transition rules”
    - If there is a transition rule *S → S’ followed by a condition* such that S and S’ are states and the condition is satisfied, there is an edge between S and S’ in the graph

Graph and answer set

- Transition rules
- Semi-terminal state
- Result

Transition rules

- Basic rules
  - Rules based on satisfying the program rules
  - Rules based on *unfounded set*
  - Backjump (backtrack)
  - Decide
  - Fail
- Rules about learning
- Rules about forgetting

Basic rules

Unit Propagate:

$$M||\Gamma \implies M\ a||\Gamma \ \text{ if } \begin{cases} a \leftarrow B \in \Pi \ \text{ and} \\ B \subseteq M \end{cases}$$

All Rules Cancelled:

$$M||\Gamma \implies M\ \neg a||\Gamma \ \text{ if } \overline{B} \cap M \neq \emptyset \text{ for all } B \in \mathit{Bodies}(\Pi, a)$$

Backchain True:

$$M||\Gamma \implies M\ l||\Gamma \ \text{ if } \begin{cases} a \leftarrow B \in \Pi, \\ a \in M, \\ \overline{B'} \cap M \neq \emptyset \text{ for all } B' \in \mathit{Bodies}(\Pi, a) \setminus B\ , \\ l \in B \end{cases}$$

Backchain False:

$$M||\Gamma \implies M\ \bar{l}||\Gamma \ \text{ if } \begin{cases} a \leftarrow l, B\ \in \Pi, \\ \neg a \in M \text{ or } a = \bot, \\ B \subseteq M \end{cases}$$

?

Unfounded:

$$M||\Gamma \implies M\ \neg a||\Gamma \ \text{ if } \begin{cases} M \text{ is consistent and} \\ a \in U \text{ for a set } U \text{ unfounded on } M \text{ w.r.t. } \Pi \end{cases}$$

- A clause $l \vee C$ is a *reason* for l to be in a list of literals $P\ l\ Q$ w.r.t P if P satisfies $l \vee C$ and $\overline{C} \subseteq P$.
- P *satisfies* a formula F when for any consistent and complete set M of literals, if M+ is an answer set for P, then M |= F.

Backjump:

$$P\ l^{\Delta}\ Q||\Gamma \implies P\ l'||\Gamma \ \text{ if } \begin{cases} P\ l^{\Delta}\ Q \text{ is inconsistent and} \\ \text{there exists a reason for } \bar{l}' \text{ to be in } P\ l' \text{ w.r.t. } \Pi \end{cases}$$

what's $\bar{l}'$?

Decide:

$$M||\Gamma \implies M\ l^{\Delta}||\Gamma \ \text{ if } \begin{cases} M \text{ is consistent and} \\ l \text{ is unassigned by } M \end{cases}$$

Fail:

$$M||\Gamma \implies \mathit{FailState} \ \text{ if } \begin{cases} M \text{ is inconsistent and} \\ M \text{ contains no decision literals} \end{cases}$$

Rules on learning and forgetting

Learn:
$$M||\Gamma \implies M|| \leftarrow B,\ \Gamma \ \text{ if } \ \Pi \text{ satisfies } \overline{B}$$

Forget:
$$M|| \leftarrow B,\ \Gamma \implies M||\Gamma$$

- *Semi-terminal* state: there is no edge due to one of the *basic* transition rules leaving this node.

Graph and Answer sets

- Given a program P and its graph G_P
  - every path in G_P contains only finitely many edges labeled by Basic transition rules,
  - for any semi-terminal state M||Γ of G_P reachable from ∅||∅, M+ is an answer set of P,
  - *FailState* is reachable from ∅||∅ in G_P if and only if P has no answer sets.

Example

$a \leftarrow not\ b$

$b \leftarrow not\ a$

$c \leftarrow a$

$d \leftarrow d.$

program

$\emptyset \| \| \emptyset$	$\Longrightarrow$	(Decide)
$a^{\Delta} \| \| \emptyset$	$\Longrightarrow$	(Unit Propagate)
$a^{\Delta}\ c \| \| \emptyset$	$\Longrightarrow$	(All Rules Cancelled)
$a^{\Delta}\ c\ \neg b \| \| \emptyset$	$\Longrightarrow$	(Decide)
$a^{\Delta}\ c\ \neg b\ d^{\Delta} \| \| \emptyset$	$\Longrightarrow$	(Unfounded)
$a^{\Delta}\ c\ \neg b\ d^{\Delta}\ \neg d \| \| \emptyset$	$\Longrightarrow$	(Backjump)
$a^{\Delta}\ c\ \neg b\ \neg d \| \| \emptyset$		

A path

Extended graph of a program

- Backjump: in contrast to backtrack to the previous decision literal, it can backtrack to the earlier decision literal which "causes" the current conflict. Efficient in SAT solvers
- Learning and forgetting: clauses are learned from the current conflict. They can be used to prune the search space. Forgetting is necessary because too many learned clauses may slow down the solver. Again, very useful techniques in SAT solvers

- Extended graph: extended state || denials, or *FailState*
- An *extended record M* relative to a program P is a list of literals over atoms in P without repetitions where
  - (i) each literal *l* in *M* is annotated either by or by a reason for *l* to be in *M*,
  - (ii) for any inconsistent prefix of *M* its last literal is annotated by a reason.

Example: extended record

program

$$\boxed{\begin{array}{l} a \leftarrow \; not\ b \\ c. \end{array}}$$

an extended state

$$b^{\Delta} \; a^{\Delta} \; \neg\, b^{\neg b \vee \neg a}$$

Example: non extended record

$a^{\Delta} \; \neg a^{\Delta}$ $\quad a^{\Delta} \; \neg\, b^{\neg b \vee \neg a} \; b^{\Delta}$ $\quad b^{\Delta} \; a^{\Delta} \; \neg\, b^{\neg b \vee \neg a} \; c^{\Delta}$

Extended graph

- We now define a graph G↑_P for any program P. Its nodes are the extended states relative to P. The transition rules of G_P are extended to G↑_P as follows: S1 → S2 is an edge in G↑_P justified by a transition rule T if and only if $S_1^\downarrow \to S_2^\downarrow$ is an edge in G_P justified by T .

$S_1^\downarrow$: the state obtained by dropping reasons from S_1

e.g., S_1: $a^\Delta \neg b^{\neg b \vee \neg a} \,||\, \phi$ $\quad S_1^\downarrow$: $a^\Delta \neg b \,||\, \phi$

Proposition 1 ↑

- For any program P,
  - a) every path in G ↑ _P contains only finitely many edges labeled by Basic transition rules,
  - b) for any semi-terminal state M││Γ of G ↑ _P, M+ is an answer set of P,
  - c) G ↑ _P contains an edge leading to *FailState* if and only if P has no answer sets.
- Note
  - Any semi-terminal state and *FailState* is reachable from $\phi || \phi$ in G ↑ _P?

(Lemma 1 proof may help answer this).

Answer set solver

- Consider finding only one answer set
- A solver using the same inference rules (unit propagate etc.) as those of G_P (or G ↑ _P) can be characterized by its strategies of traversing the graph to find a path from $\phi || \phi$ to a semi-terminal or FailState.

?

SMODELS_cc

1. edges corresponding to the applications of transition rules Unit Propagate, All Rules Cancelled, Backchain True, Backchain False, and Unfounded to a state in G_P are considered if Backjump is not applicable in this state,
2. an edge corresponding to an application of a transition rule Decide to a state in G_P is considered if and only if none of the rules among Unit Propagate, All Rules Cancelled, Backchain True, Backchain False, Unfounded, and Backjump is applicable in this state,
3. an edge corresponding to an application of a transition rule Learn to a state in G_P is considered if and only if this state was reached by the edge Backjump and a FirstUIP backjump clause is learned

SUP

1 – 3 of SMODELS_cc

- an edge corresponding to an application of transition rule Unfounded to a state in G_P is considered only if a state assigns all atoms of P
- Remove unfounded from 2.

Generate the reasons

Unit Propagate:

$$M||\Gamma \implies M\ a||\Gamma \ \text{ if } \begin{cases} a \leftarrow B \in \Pi \ \text{ and} \\ B \subseteq M \end{cases}$$

reason:

$a \vee \overline{B}$ (if B then a)

All Rules Cancelled:

$$M||\Gamma \implies M\ \neg a||\Gamma \ \text{ if } \overline{B} \cap M \neq \emptyset \text{ for all } B \in Bodies(\Pi, a)$$

Reason: $\forall B$, let $f(B)$ be a literal $\in B \cap \overline{M}$

$\neg a \vee \bigvee_{B \in Bodies(\Pi,a)} f(B)$ (if $\forall B, f(B)$, then $\neg a$)

Backchain True:

$$M||\Gamma \implies M\ l||\Gamma \ \text{ if } \begin{cases} a \leftarrow B \in \Pi, \\ a \in M, \\ \overline{B'} \cap M \neq \emptyset \text{ for all } B' \in Bodies(\Pi, a) \setminus B, \\ l \in B \end{cases}$$

Reason: $l \vee \neg a \vee \bigvee_{B' \in Bodies(\Pi,a) \setminus B} f(B')$.

Backchain False:

$$M||\Gamma \implies M\ \overline{l}||\Gamma \ \text{ if } \begin{cases} a \leftarrow l, B \in \Pi, \\ \neg a \in M \text{ or } a = \bot, \\ B \subseteq M \end{cases}$$

reason: $\overline{l} \vee a \vee \overline{B}$

Unfounded:

$$M||\Gamma \implies M\ \neg a||\Gamma \ \text{ if } \begin{cases} M \text{ is consistent and} \\ a \in U \text{ for a set } U \text{ unfounded on } M \text{ w.r.t. } \Pi \end{cases}$$

If $\forall B \in Bodies(\Pi, U)$ such that $U \cap B^+ = \emptyset$, $\overline{B} \cap M \neq \emptyset$, then $\forall a \in U$, $\neg a$.

$$\neg a \vee \bigvee_{B \in Bodies(\Pi,U),\ B^+ \cap U = \emptyset} f(B)$$

Backjump:

$$P\ l^{\Delta}\ Q||\Gamma \implies P\ l'||\Gamma \ \text{ if } \begin{cases} P\ l^{\Delta}\ Q \text{ is inconsistent and} \\ \text{there exists a reason for } l' \text{ to be in } P\ l' \text{ w.r.t. } \Pi \end{cases}$$

Reason:

latter, we discuss how to find a reason.

Decide:

$$M||\Gamma \implies M\ l^{\Delta}||\Gamma \ \text{ if } \begin{cases} M \text{ is consistent and} \\ l \text{ is unassigned by } M \end{cases}$$

Fail:

$$M||\Gamma \implies \mathit{FailState} \ \text{ if } \begin{cases} M \text{ is inconsistent and} \\ M \text{ contains no decision literals} \end{cases}$$

Rules on learning and forgetting

Learn:

$$M||\Gamma \implies M|| \leftarrow B,\ \Gamma \ \text{ if } \Pi \text{ satisfies } \overline{B}$$

Forget:

$$M|| \leftarrow B,\ \Gamma \implies M||\Gamma$$

Backjump related notations

- We call the reason in the backjump rule *backjump clause.*

Backjump:

$$P\ l^{\Delta}\ Q||\Gamma \Longrightarrow\ P\ l'||\Gamma \ \text{ if } \begin{cases} P\ l^{\Delta}\ Q \text{ is inconsistent and} \\ \text{there exists a reason for } l' \text{ to be in } P\ l' \text{ w.r.t. } \Pi \end{cases}$$

- We say that a state in the graph G↑_P is a *backjump state* if its record is inconsistent and contains a decision literal.

- For a record M, by lcp(M) we denote its largest consistent prefix.
- A clause C is *conflicting* on a list M of literals if P satisfies C, and C ⊆ lcp(M). e.g.,

$a \leftarrow not\ b$	$k \leftarrow l,\ not\ b$
$b \leftarrow not\ a,\ not\ c$	$\leftarrow m,\ not\ l,\ not\ b$
$c \leftarrow not\ f$	$m \leftarrow\ not\ k,\ not\ l$
$\leftarrow k,\ d$	

$a^{\Delta}\ \neg b^{\neg b \vee \neg a}\ c^{\Delta}\ \neg f^{\neg f \vee \neg c}\ d^{\Delta}\ \neg k^{\neg k \vee \neg d}$

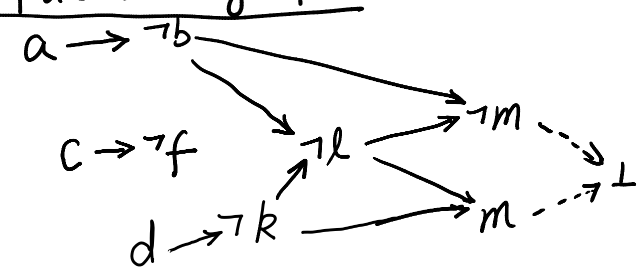
$\neg l^{\neg l \vee b \vee k}\ \neg m^{\neg m \vee l \vee b}\ m^{m \vee k \vee l}$ — Conflicting clause

inconsistent

$a^{\Delta}\ \neg b^{\neg b \vee \neg a}\ c^{\Delta}\ \neg f^{\neg f \vee \neg c}\ d^{\Delta}\ \neg k^{\neg k \vee \neg d}$

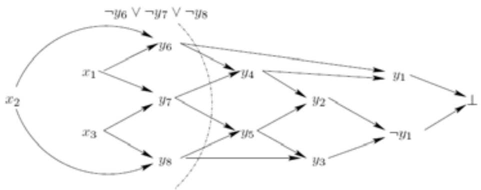
$\neg l^{\neg l \vee b \vee k}\ \neg m^{\neg m \vee l \vee b}\ m^{m \vee k \vee l}$

Implication graph

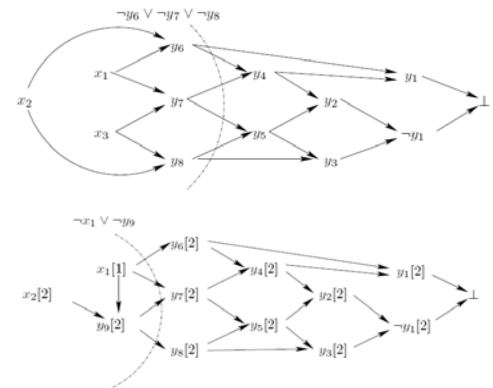


CUT

- What's a cut
  - A *cut* in the implication graph is a bipartition of the graph such that all decision variables are in one set while the conflict is in the other set.
  - There are many cuts

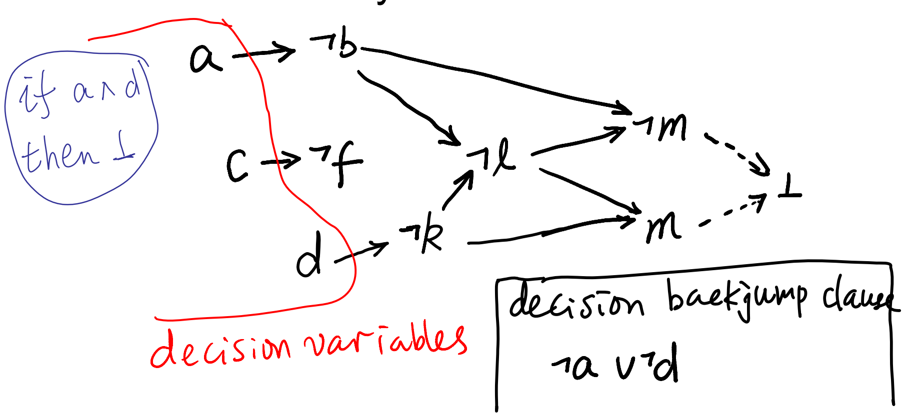


- Each cut results in a learned clause

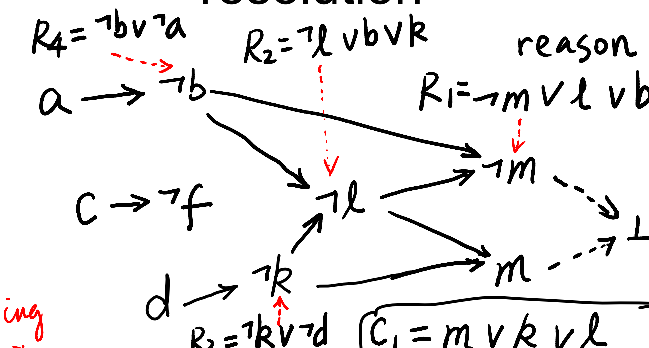


(Decision) backjump clause through graph

- Decision backjump clause: one set of the cut contains only decision variables



Obtain backjump clause through resolution



$C_1 = m \vee k \vee l$ Conflicting clause

Conflicting clause

$C_5: \neg d \vee \neg a$ (C_4, R_4)

$C_2: k \vee l \vee b$ (resolvent of C_1, R_1)

$C_3: k \vee b$ (. . . C_2, R_2)

$C_4: \neg d \vee b$ (C_3, R_3)

Apply backjump rule

$a^{\Delta} \neg b^{\neg b \vee \neg a}\ c^{\Delta} \neg f^{\neg f \vee \neg c}\ d^{\Delta} \neg k^{\neg k \vee \neg d} \neg l^{\neg l \vee b \vee k} \neg m^{\neg m \vee l \vee b}\ m^{m \vee k \vee l} \Longrightarrow$

$a^{\Delta}\ \neg b^{\neg b \vee \neg a}\ \neg d^{\neg d \vee \neg a} || \emptyset$

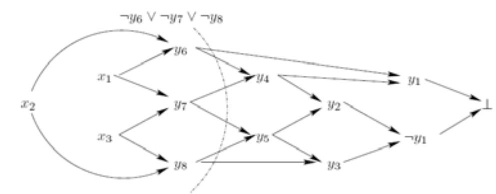
- backjump clause (learned clause / conflict clause)
- reason for ¬d.

SAT

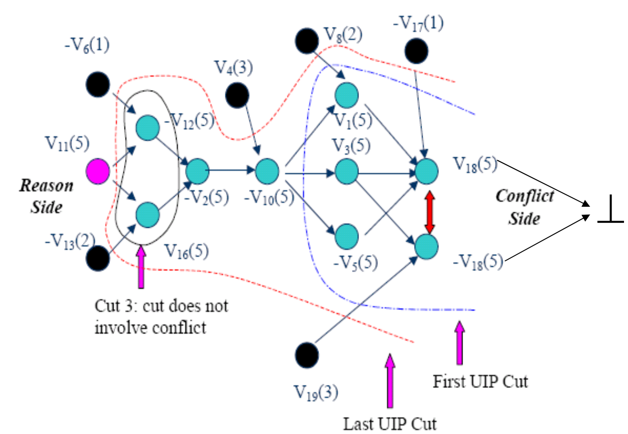
(see Lemma 6 on C^{Δ} is passed by backjump).
Let $\{l_1, l_2, \cdots, l_i\}$ be a learned clause, we backjump to the decision level of l_{i-1}.

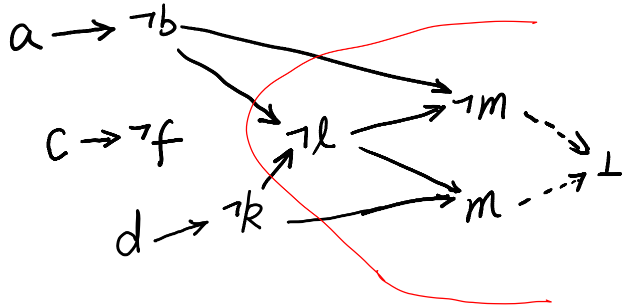
UIP

- What's *unique implication point* (UIP)
  - A literal *l* in a implication graph is called a *unique implication* point if every path from the decision literal at level *l* to the point of conflict passes through *l*.
  - A *decision level* of a literal *l* is the number of decision variables when *l* is assigned a value.



- First UIP cut
  - The *1UIP cut* of of an implication graph is the cut "generated from" the unique implication points closest to the point of conflict.
    - On one set (conflict side): all variables assigned after the first UIP of current decision level reachable to the conflict
    - On the other side: everything else.





Apply backjump rule

$$a^{\Delta} \neg b^{\neg b \vee \neg a} \; c^{\Delta} \neg f^{\neg f \vee \neg c} \; d^{\Delta} \neg k^{\neg k \vee \neg d} \neg l^{\neg l \vee b \vee k} \neg m^{\neg m \vee l \vee b} \; m^{m \vee k \vee l} \Longrightarrow$$

$$a^{\Delta} \; \neg b^{\neg b \vee \neg a} \; k^{k \vee b} || \emptyset$$

backjump clause : $k \vee b$

decision variables leading to $\perp$:

(a, d)

So, backjump to the decision level of a.

Application of learning rule

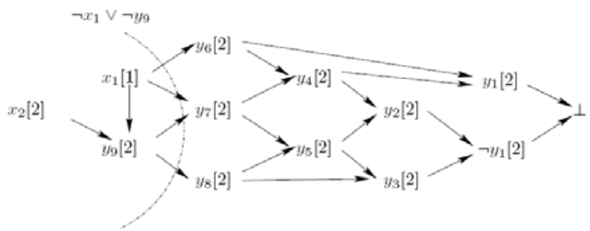
- Reason carried by the last literal can be put into the store

Refences

1. An abstract answer set solver by Yuliya
2. Efficient Conflict Driven Learning in a Boolean Satisfiability Solver, iccad 2001

Appendix

- Another implication graph

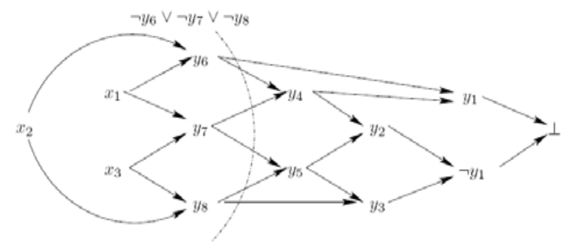


Backjump clause (conflict clause)

Clauses

$$\neg y_1 \vee \neg y_2 \vee \neg y_3 \quad \neg y_8 \vee \neg y_5 \vee y_3 \quad \neg y_7 \vee \neg y_8 \vee y_5$$
$$\neg y_4 \vee \neg y_6 \vee y_1 \quad \neg y_6 \vee \neg y_7 \vee y_4 \quad \neg x_1 \vee \neg x_2 \vee y_6$$
$$\neg y_4 \vee \neg y_5 \vee y_2 \quad \neg x_2 \vee \neg x_3 \vee y_8 \quad \neg x_1 \vee \neg x_3 \vee y_7$$

Implication graph (with decision literals x1, x2, x3)



Notations

- Unfounded set
  - A set U of atoms occurring in a program P is said to be unfounded on a consistent set M of literals w.r.t. P if for every $a \in U$ and every $B \in$ Bodies(P, a), $\bar{B} \cap M \neq \emptyset$ or $U \cap B^{+} \neq \emptyset$.