Chapter 11

Datalog Knowledge Bases II

NEGATION IN THE BODY: CYCLIC NEGATIVE DEPENDENCIES

A program whose dependency graph contains a *negative cycle* cannot be stratified.

- Consider the program $P = \{p(b) \leftarrow \neg p(a)\}$ (without any assured facts). It has three models, $\mathcal{M}_1 = \{p(b)\}$, $\mathcal{M}_2 = \{p(a)\}$, and $\mathcal{M}_3 = \{p(a), p(b)\}$.
  Both $\mathcal{M}_1$ and $\mathcal{M}_2$ are minimal.

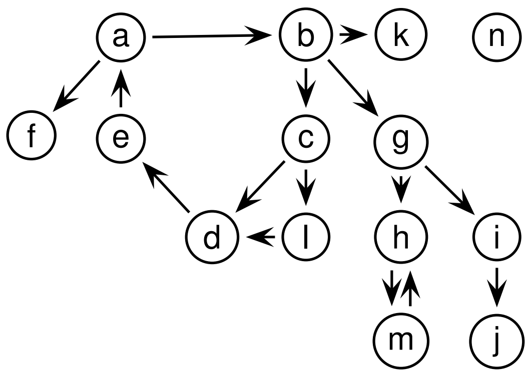
  Which of the models is "preferable", given P as a knowledge base?
- well-founded semantics (still polynomial)
- stable semantics (answer set programming) (exponential)
- the rule is logically equivalent to $p(a) \vee p(b)$ – but as a rule, it can be read to have a more "directed" meaning:
  "if $p(a)$ cannot be shown, then assume $p(b)$".

Example: Win-Move-Game

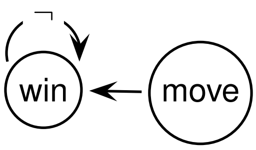
- 2 players,
- positions on a board that are connected by (directed) moves (relation "move(x,y)"),
- first player puts a pebble on a position,
- players alternately move the pebble from x to a connected y,
- if a player cannot move, he loses.
- Question: which positions are "winning" positions, "losing" position, or "drawn" positions?

The following program "describes" the game:

```
win(X) :-  move(X,Y), not win(Y).
```



- the dependency graph contains a negative cycle:



Well-Founded Semantics: Motivation

... switch from "stupid" bottom-up to well-founded argumentation "why or why not".

- every fact has an individual finite proof
  (positive/existential part: linear; not-exists/forall part: multiple ((finitely) failed) subproofs)
- but not stratified (but "dynamically stratified"/"ground-stratified")

1. basic facts,
2. apply rules based on existing knowledge
3. additional facts,
4. continue with (2);
5. including "negative facts" – under closed-world assumption (CWA).

- Does this need full reasoning? (tableau proofs obviously cover it)
- is resolution sufficient? (yes, it's only rule applications)
- theory: how to characterize the model?
- three-valued logic: yes-no-undefined (win-move: lost/won/drawn)
- how to compute the model efficiently?

Analysis

- which atoms are definitely true?
  - the facts
  - instantiations $\sigma(H)$ of rule heads of rules $H \leftarrow C_1 \wedge \ldots \wedge C_n \wedge \neg D_1 \wedge \ldots \wedge \neg D_k$
    - where all $\sigma(C_i)$ are definitely true, and
    - where all $\sigma(D_i)$ are definitely false.
- which atoms are definitely false (under CWA)?
  - instances of EDB predicates that are not amongst the given facts,
  - ground instances $p(...)$ of IDB predicates such that for all rules whose rule head H unifies with $p(...)$ as $\sigma(H)$ (there might be several such rules with $p(...)$ in their head): $H \leftarrow C_1 \wedge \ldots \wedge C_n \wedge \neg D_1 \wedge \ldots \wedge \neg D_k$
    - some $\sigma(C_i)$ is definitely false, or
    - some $\sigma(D_i)$ is definitely true.
- idea: start with nothing. Derive some definitively true things and some definitively false ones.
- based on the obtained knowledge, do "next round",
- care for "still unknown" things.

Well-Founded Semantics: For What

- Many real problems are stratified.
- Most (relational/SQL) queries are stratified.
- WFS goes beyond classical queries:
  many problems can be encoded in Datalog wrt. well-founded semantics

Non-Stratified examples:

- logical puzzles ;)
- planning problems
  can_start(Y) $\leftarrow$ completed(X), *additional conditions*.
- argumentation contexts
  holds(...) $\leftarrow$ holds(...), $\neg$ holds(...), *additional conditions*.

Let's have a look at the theory ...

REDUCT OF A PROGRAM

Consider a Herbrand interpretation (i.e., a set of ground facts) $\mathcal{H}$.

Definition 11.1 (Reduct of a Program)
The reduct $P^{\mathcal{H}}$ of a program P wrt. a Herbrand interpretation $\mathcal{H}$ is obtained as follows:

- *let P_g denote the grounding of P, i.e. the set of all ground instances of rules in P over elements of the Herbrand universe of $\mathcal{H} \cup P$.*
- *delete from P_g all rules that contain a negative literal $\neg a$ in the body such that $a \in \mathcal{H}$, (these rule bodies cannot be satisfied in $\mathcal{H}$)*
- *delete all remaining negative literals in the bodies of the remaining rules. (for those $\neg a$, $a \notin \mathcal{H}$, i.e., these literals are satisfied in $\mathcal{H}$)* □

Properties of $P^{\mathcal{H}}$

- $P^{\mathcal{H}}$ is a (ground) positive program.
- If $\mathcal{H}$ is a model of P, then $T_{P^{\mathcal{H}}}(\mathcal{H}) \subseteq \mathcal{H}$.
  (note: use $T_{P^{\mathcal{H}}}(\mathcal{H}$ here, not T^{ω}, but run it on $\mathcal{H}$)

11.1 Stable Models I

Definition 11.2 (M. Gelfond, V. Lifschitz, ICLP 1988)
A Herbrand interpretation $\mathcal{H}$ is a stable model *of a Datalog$^\neg$ program P, if*

$$T^\omega_{P^\mathcal{H}}(\emptyset) = \mathcal{H}.$$

□

- note that a program P can have several stable models.

Remark and Exercise

Note that the definition of stable models is based on $T^\omega_{P^\mathcal{H}}(\emptyset)$.

Consider $P = \{$p(a) :- p(a)$\}$ and $\mathcal{H} = \{p(a)\}$; $P^\mathcal{H} = P$.
$\mathcal{H}$ is a model of P, and $T^\omega_{P^\mathcal{H}}(\mathcal{H}) = \mathcal{H}$.

But, $T^\omega_{P^\mathcal{H}}(\emptyset) = \emptyset$, i.e., $\mathcal{H}$ is not a stable model ($p(a)$ is not “supported”).

$\mathcal{H}' = \{p(a), p(b), q(b)\}$ is also a model of P, which is also (obviously) not stable.

Obviously, $\emptyset$ is a stable model of P – and thus, is the only one.

Note that the above example is a positive Datalog program. For positive Datalog programs P, and any $\mathcal{H}$, $P^\mathcal{H} =$ ground(P) (i.e., all ground instances of rules of P) and $T^\omega_{\text{ground}(P)}(\emptyset) = T^\omega_P(\emptyset)$ is the only stable model.

Stable Models – Example

Consider the following program P:

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

[Filename: Datalog/qnotp.s]

Logically, the rule is equivalent to $p(a) \vee q(a)$.

- The program has one stable model:

```
> lparse -n 0 qnotp.s|smodels
Answer: 1
Stable Model: q(a)
True
```

  For $\mathcal{H} = \{q(a)\}$, $P^\mathcal{H} = \{$q(a) :- true$\}$ and $T^\omega_{P^\mathcal{H}}(\emptyset) = \{q(a)\}$, thus $\mathcal{H}$ is stable.

- Consider $\mathcal{H}' = \{p(a)\}$. It is a model of P.
  $P^{\mathcal{H}'} = \emptyset$ and $T^\omega_{P^{\mathcal{H}'}}(\emptyset) = \emptyset$.
  The derivation of $p(a)$ is “not supported” by P; $\mathcal{H}'$ is not stable.

- so, in Stable Models Semantics, the rule does not mean disjunction, but is directed.

Stable Models – Example

Consider the following program:

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

[Filename: Datalog/porq.s]

Logically, each of the rules is equivalent to $p(a) \vee q(a)$.

- The program has two total stable models, and one partial (which is the well-founded model):

```
> lparse -n 0 --partial porq.s|smodels
Answer: 1
Stable Model: q(a)
Answer: 2
Stable Model: p(a)
Answer: 3
Stable Model: q'(a) p'(a)
```

- thus, both rules together represent disjunction.
- Note that $\{p(a), q(a)\}$ is a model, but not a stable model.
- There is no possibility in Datalog$^\neg$ to assert $\neg q(a)$ to forbid one of the models. (in smodels, this will be allowed)

Stable Models – Example

Consider the following program:

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

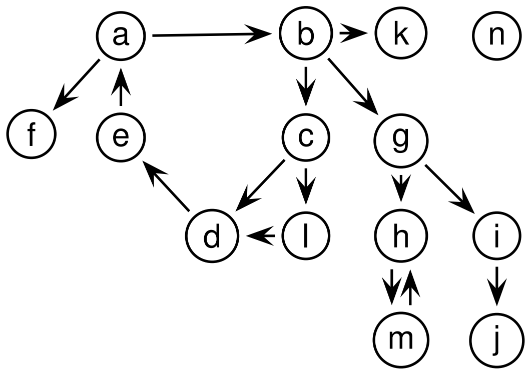
[Filename: Datalog/pporq.s]

- The program has only one stable model: $\{p(a)\}$.
- This model is also the well-founded model.

WinMove with Stable Models

- lparse does not accept don't-care-variables.

```
pos(a). pos(b). pos(c). pos(d). pos(e). pos(f). pos(g).
pos(h). pos(i). pos(j). pos(k). pos(l). pos(m). pos(n).
win(X) :-  move(X,Y), not win(Y).
lose(X) :- pos(X), not win(X).
move(a,b).   move(a,f).
move(b,c).   move(b,g).   move(b,k).
move(c,d).   move(c,l).
move(d,e).
move(e,a).
move(g,i).   move(g,h).
move(h,m).
move(i,j).
move(l,d).
move(m,h).
```



[Filename: Datalog/winmove.s]

- `lparse -n 0 -d none winmove.s | smodels` yields *two total* two-valued stable models.
- drawn cycle between h and m: once w/l, other l/w
- wfm = intersection of stable models, minimal 3-valued model.

Stable Models – First Summary

- A Datalog$^\neg$ program may have several stable models.
- Finding the stable models of a program is exponential (optimization strategies exist)
- come back to the well-founded semantics
  - cheaper (polynomial),
  - returns a *unique* reasonable result in cases where disjunction is not needed or not intended,
  - cf. win-move game: drawn positions are neither lost nor won.
- ... a closer investigation of stable models semantics will be given on Slides 671 ff.

11.2 Well-Founded Semantics

- recall the considerations from Slides 628 ff.:
  well-founded non-stratified "argumentation" which facts can be derived to be true or false

Main Problem:

How to deal with true-unknown-false:

- model-theoretic: three-valued logic
- practically: apply a trick to be able to use the existing 2-valued T_P operator for *positive* Datalog.

Definition

Definition 11.3 (A. Van Gelder, K.A. Ross, J.S. Schlipf, PODS 1988)
Given a Datalog$^\neg$ program P, the well-founded model *of P is the minimal 3-valued stable model of P.* □

- from the practical view not very promising ...
  not only to guess stable models, yet even 3-valued.
- have a look at this definition later.

Alternating Fixpoint Computation for WFS

The Alternating Fixpoint Computation [A. Van Gelder, PODS 1989] mirrors the well-foundedness of the derivation:

Definition 11.4
Given a Datalog$^\neg$ program P over a signature Σ, define the sequence $I_0, I_1, \ldots$ of Herbrand interpretations over Σ als follows:

$$\begin{array}{rcl} I_0 & := & \emptyset \\ I_{i+1} & := & T^{\omega}_{P^{I_i}}(\emptyset) \end{array}$$

□

- Does $((I_k))$ converge?
  No. And Yes.
- Is there a fixpoint?
  Yes. There are two fixpoints!

... let's have a look ...

Exercise

Evaluate $((I_k))$ for the win-move example.

Alternating Fixpoint: Analysis

- Consider first the program P^{facts} which consists only of the facts (= fact rules) in P:
  - $T^{\omega}_{P^{\text{facts}}}(\emptyset) = T^{1}_{P^{\text{facts}}}(\emptyset)$ makes all facts true that are contained in the program.
- Consider next the program P^{+} which is obtained from ground(P) by deleting all rules that contain any negative literal:
  - P^{+}: corresponds to "all negative literals are false".
    Recall that $\mathcal{HB}_P$ denotes the interpretation that makes all possible atoms over the Herbrand Universe of P true. With this, $P^{+} = P^{\mathcal{HB}_P}$.
  - (P^{+} can be equivalently obtained by first deleting all rules that contain a negative literal and then grounding the remaining (positive) rules)
  - P^{+} is the smallest possible reduct of P,
  - $T^{\omega}_{P^{+}}(\emptyset)$ derives all atoms that can be derived by only the remaining purely positive rules,
  - this includes all facts (recall fact rules of the form `p(...) :- true.`)
  - ⇒ these are atoms that hold in *all* models of P (facts+positive rules force them).
  - ⇒ a safe and very careful underestimate of true atoms.

$$\emptyset \subseteq T^{1}_{P^{\text{facts}}}(\emptyset) \subseteq T^{\omega}_{P^{+}}(\emptyset) \subseteq T^{\omega}_{P^{anyI}}(\emptyset) \subseteq \mathcal{HB}_P$$

Alternating Fixpoint: Analysis

Consider now the program P^{-} which is obtained from ground(P) by simply deleting all negative literals from all rules (corresponds to "all negative literals are satisfied"):

- P^{-} is the reduct wrt. the empty interpretation, the starting point of the whole process,
- P^{-} it is the biggest possible reduct of P
- $T^{\omega}_{P^{-}}(\emptyset)$ derives all atoms that can be derived by P if all negative literals are assumed to be satisfied.
- this includes again all facts (recall fact rules of the form `p(...) :- true.`)
- and everything that could by derived from them under "optimal" conditions

⇒ an *overestimate* of true atoms.

⇒ atoms that are not in $T_{P^{-}}(\emptyset)$ can definitely not be derived by P,

⇒ a safe *underestimate of false atoms* (in any stable model/wrt. Closed-World Assumption).

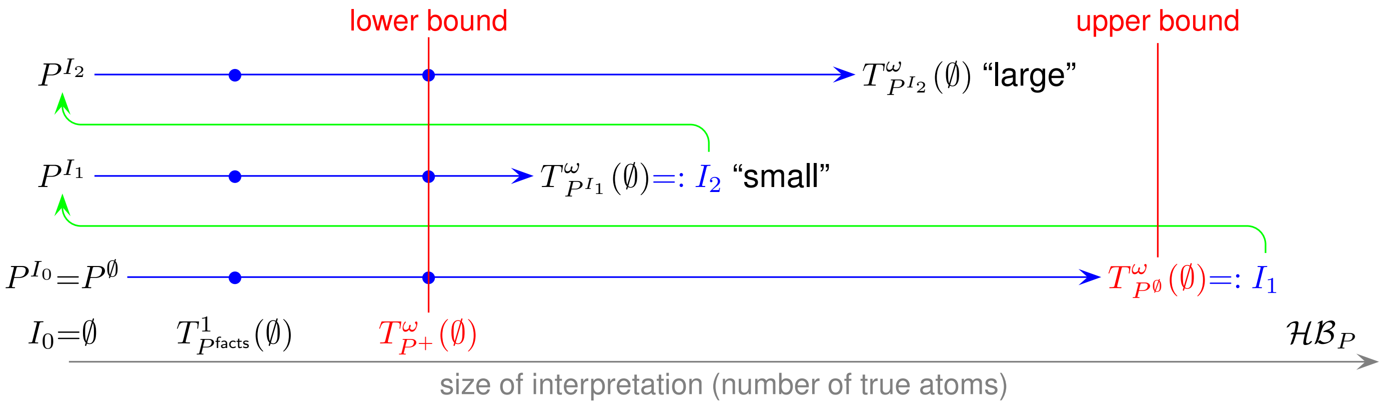
- Example: Consider $P = \{$p(a), p(b):- not p(a)$\}$. Then, $P^{-} = \{$p(a), p(b):-true$\}$ and $T^{\omega}_{P^{-}}(\emptyset) = \{p(a), p(b)\}$.
- use this for starting with $I_0 = \emptyset$ and thus considering $P^{\emptyset} = P^{-}$:

$$\emptyset \subseteq T^{1}_{P^{\text{facts}}}(\emptyset) \subseteq T^{\omega}_{P^{+}}(\emptyset) \subseteq T^{\omega}_{P^{anyI}}(\emptyset) \subseteq T^{\omega}_{P^{-}}(\emptyset) = T^{\omega}_{P^{\emptyset}}(\emptyset) \subseteq \mathcal{HB}_P$$

Well-Founded Semantics Computation: Intuitive Analysis

⇒ coming back to the inductive definition:
$I_0 = \emptyset$,
$I_1 = T^{\omega}_{P^{\emptyset}}(\emptyset)$ is an overestimate of true atoms and an underestimate of false atoms.

- observation: the larger I, the *smaller* the reduct P^I (delete non-satisfied negative literals), the smaller $T^{\omega}_{P^I}(\emptyset)$ ("antimonotonic")
- P^{I_1} is a "small" reduct program, $T^{\omega}_{P^{I_1}}$ is a "small" interpretation, but $\supseteq T^{\omega}_{P^+}(\emptyset)$
- P^{I_2} is a "large" reduct program, $T^{\omega}_{P^{I_2}}$ is a "large" interpretation, but $\subseteq T^{\omega}_{P^{\emptyset}}(\emptyset)$



Alternating Fixpoint: Analysis

$$\begin{aligned} I_0 &:= \emptyset \\ I_{i+1} &:= T^{\omega}_{P^{I_i}}(\emptyset) \end{aligned}$$

- in each step, P^{I_i} encodes the knowledge about false atoms from I_i into P.
- $T^{\omega}_{P^{I_i}}$ runs the resulting positive program under consideration of these false atoms:
- if I_i is an underestimate of false atoms:
  - only negative literals that are already proven to be true are assumed to be true.

  ⇒ underestimate of the satisfied rule bodies,

  ⇒ underestimate of the true heads.

  ⇒ $I_{i+1} = T^{\omega}_{P^{I_i}}$ is an underestimate of true atoms.
- Analogously, if I_i is an overestimate of false atoms, $I_{i+1} = T^{\omega}_{P^{I_i}}$ is an overestimate of true atoms.

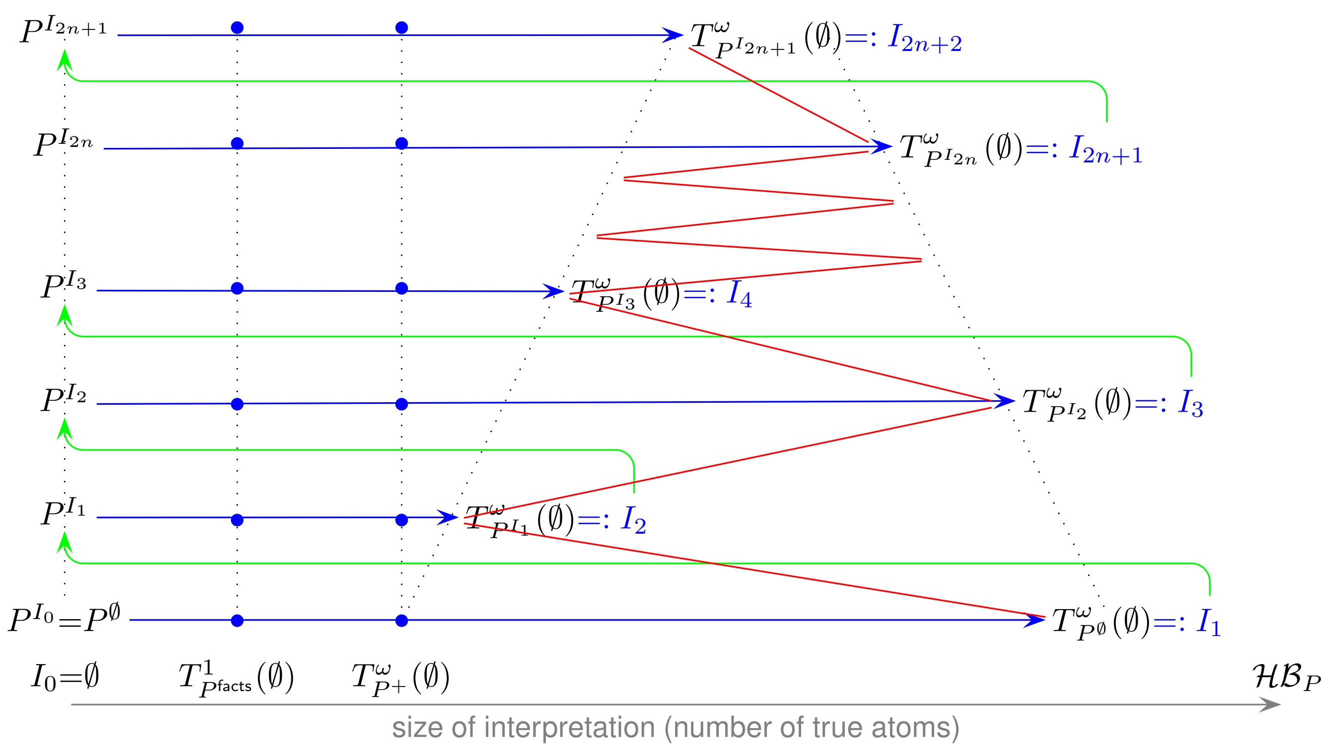
Alternating Fixpoint: Analysis

$$
\begin{aligned}
I_0 &= \emptyset \\
I_{i+1} &= T^{\omega}_{P^{I_i}}(\emptyset)
\end{aligned}
$$

- I_0 is an underestimate of true atoms and an overestimate of false atoms,
- I_1 is an overestimate of true atoms and an underestimate of false atoms,
- I_{2n} is an underestimate of true atoms and an overestimate of false atoms,
- I_{2n+1} is an overestimate of true atoms and an underestimate of false atom,
- and with each step, the estimates get better.
- To be proven by interleaved induction:
  - increasing sequence of underestimates:
    $I_{2(n+1)} \geq I_{2n}$ (base case obvious: $I_2 \geq I_0 = \emptyset$)
  - decreasing sequence of overestimates:
    $I_{2n+3} \geq I_{2n+1}$ (first element $I_1 = T^{\omega}_{P^{\emptyset}}(\emptyset) = T^{\omega}_{P-}(\emptyset)$ (cf. Slide 641)

Well-Founded Semantics Computation

- alternating sequence of growing underestimates and shrinking overestimates



Alternating Fixpoint: Analysis

Lemma 11.1
The mapping $I \to T^{\omega}_{P^I}(\emptyset)$ is antimonotonic:
If $I \leq J$, then $T^{\omega}_{P^I}(\emptyset) \geq T^{\omega}_{P^J}(\emptyset)$. □

Proof *$I \leq J$ means that $I \subseteq J$, i.e., in I more atoms evaluate fo false. Thus, in P_I more negative literals are removed (because they are satisfied in I), thus less rules are removed due to remaining negative literals (which are not satisfied). Thus, $P_I \supseteq P_J$ (as sets of ground rules), thus $T^{\omega}_{P^I}(\emptyset) \supseteq T^{\omega}_{P^J}(\emptyset)$.* □

Alternating Fixpoint: Analysis

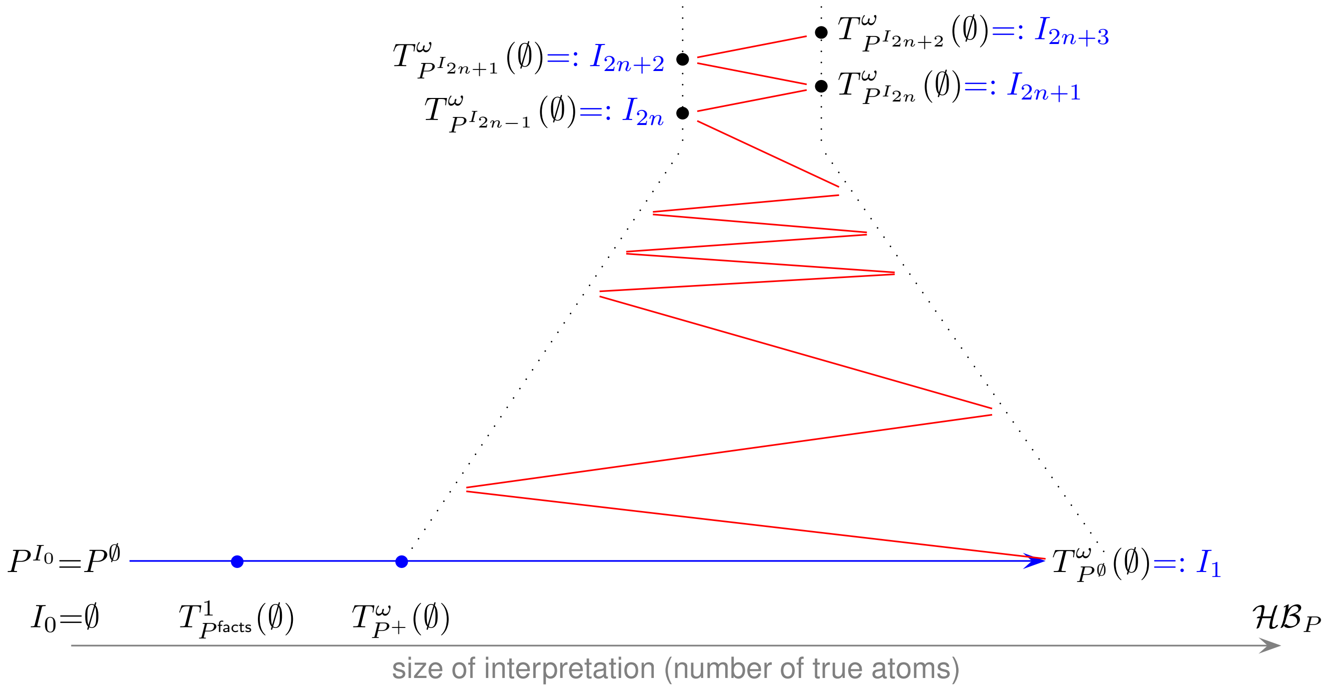
Theorem 11.1
With the above definition, $I_0 \leq I_2 \leq \ldots \leq I_{2n} \leq I_{2n+2} \leq \ldots \leq I_{2n+1} \leq I_{2n-1} \leq \ldots \leq I_1$. □

Proof *Obviously, $I_0 = \emptyset \leq I_1$ and $I_0 \leq I_2$. Thus, $I_2 = T^{\omega}_{P^{I_1}}(\emptyset) \leq T^{\omega}_{P^{I_0}}(\emptyset) = I_1$.*
$I_3 = T^{\omega}_{P^{I_2}}(\emptyset) \leq T^{\omega}_{P^{I_0}}(\emptyset) = I_1$.
Analogously by induction:
Since $I_{2n-1} \geq I_{2n+1}$: $I_{2n+2} = T^{\omega}_{P^{I_{2n+1}}}(\emptyset) \geq T^{\omega}_{P^{I_{2n-1}}}(\emptyset) = I_{2n}$.
Since $I_{2n-2} \leq I_{2n}$: $I_{2n+1} = T^{\omega}_{P^{I_{2n}}}(\emptyset) \leq T^{\omega}_{P^{I_{2n-2}}}(\emptyset) = I_{2n-1}$.
Since $I_{2n+1} \geq I_{2n}$: $I_{2n+2} = T^{\omega}_{P^{I_{2n+1}}}(\emptyset) \leq T^{\omega}_{P^{I_{2n}}}(\emptyset) = I_{2n+1}$.
Since $I_{2n} \leq I_{2n-1}$: $I_{2n+1} = T^{\omega}_{P^{I_{2n}}}(\emptyset) \geq T^{\omega}_{P^{I_{2n-1}}}(\emptyset) = I_{2n}$. □

- The I_{2n} are a monotonically increasing (and limited) sequence:
  the underestimates of true atoms.
- The I_{2n+1} are a monotonically decreasing (and limited) sequence:
  the overestimates of true atoms.
- $\lim\limits_{n\to\infty} I_{2n} \leq \lim\limits_{n\to\infty} I_{2n+1}$.
- do the limits coincide? – sometimes yes, but not always!

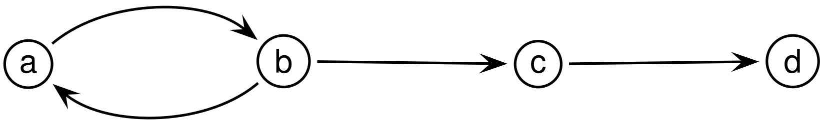
Well-Founded Semantics Computation: Alternating Fixpoints Case

- either there is an n such that $I_n = I_{n+1} = I_{n+2} = \ldots$ (single fixpoint), or
- there is an n such that $I_n = I_{n+2} = \ldots$ and $I_{n+1} = I_{n+3} = \ldots$ (alternating fixpoints):



Alternating Fixpoint: Example

Consider the small win-move game consisting of



- $I_0 = \emptyset$.
- $I_1 = \{$move(a,b), move(b,a), move(b,c), move(c,d), win(c), win(b), win(a)$\}$ – d is already $\neg$win(d) since there is no move from it.
- $I_2 = \{$move(a,b), move(b,a), move(b,c), move(c,d), win(c)$\}$ – now c is known to be won.
- $I_3 = \{$move(a,b), move(b,a), move(b,c), move(c,d), win(c), win(b), win(a)$\} = I_1$ win(b) is still there since there is the move to a.
- From then $(n \geq 2)$ on, $I_{2n} = I_2$ and $I_{2n+1} = I_1$.

How to interpret this?

- all facts in $\lim_{n\to\infty} I_{2n}$ have a well-founded derivation "to hold": win(c).
- all facts not in $\lim_{n\to\infty} I_{2n+1}$ have a well-founded derivation "not to hold": $\neg$win(d).
- all others: ?? – game: a and b are drawn positions.

What about a logical semantics? – three-valued logic: true/false/undefined.

Example: Win-Move-Game in Datalog

- XSB: use `tnot` (tabled!) – applies SLG resolution (SLD + memoing/tabling)

```
:- auto_table.
pos(a). pos(b). pos(c). pos(d).
move(a,b).  move(b,a).  move(b,c).  move(c,d).
win(X) :-  move(X,Y), tnot win(Y).
lose(X) :- pos(X), tnot win(X).
% ?- win(X)
```

[Filename: Datalog/winmovesmall.P]

```
?- win(X).
X = c
X = b undefined
X = a undefined
no
```

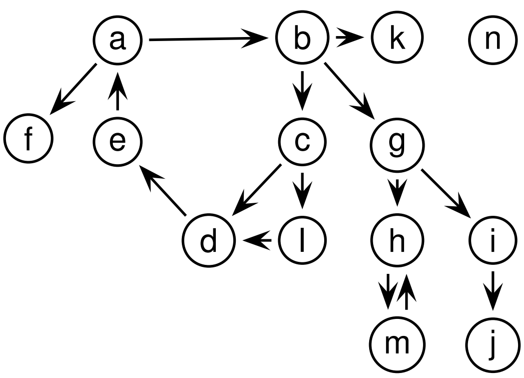
- c is won, d is lost, a and b are undefined (to be interpreted as drawn).

Aside: References

- The win-move game is used in the above-mentioned papers [M. Gelfond, V. Lifschitz, ICLP 1988], [A. Van Gelder, K.A. Ross, J.S. Schlipf, PODS 1988], [A. Van Gelder, PODS 1989].

Example: Win-Move-Game in Datalog

```
:- auto_table.
% :- table win/1.
pos(a). pos(b). pos(c). pos(d). pos(e). pos(f). pos(g).
pos(h). pos(i). pos(j). pos(k). pos(l). pos(m). pos(n).
win(X) :-  move(X,Y), tnot win(Y).
lose(X) :- pos(X), tnot win(X).
move(a,b).   move(a,f).
move(b,c).   move(b,g).   move(b,k).
move(c,d).   move(c,l).
move(d,e).
move(e,a).
move(g,i).   move(g,h).
move(h,m).
move(i,j).
move(l,d).
move(m,h).
```



[Filename: Datalog/winmove.P]

11.3 3-Valued Logic

- same syntax as FOL
- truth values t (true, 1), u (undefined, 0.5), f (false, 0), ordered by $t > u > f$.
- All three-valued logics coincide in the definition of $\wedge$, $\vee$, and $\neg$:

$A \wedge B = \min(A, B)$

$A \backslash B$	f	u	t
f	f	f	f
u	f	u	u
t	f	u	t

$A \vee B = \max(A, B)$

$A \backslash B$	f	u	t
f	f	u	t
u	u	u	t
t	t	t	t

$\neg A = 1 - a$

A	$\neg A$
f	t
u	u
t	f

- there is not a single 3-valued logic. There are multiple variants, depending on what should be done with the logic.

3-Valued Logic for Logic Programming Semantics

- does not require actual reasoning in a 3-valued world,
- define a model theory for Datalog with negation,
- express *partial models*:
  - consider Datalog with disjunction in the head (or similar situations e.g. in Description Logics/OWL):
    Consider an axiom $\forall X$: person$(X) \rightarrow$ (male$(X) \vee$ female(X)).
    Consider an interpretation $\mathcal{I}$ where there is an individual a s.t. $\mathcal{I} \models$ person(a). From $\mathcal{I} \models \forall X$: person$(X) \rightarrow$ (male$(X) \vee$ female(X)).
    the intended semantics of $\models$ and $\rightarrow$ (both must still be defined!) should imply that $\mathcal{I} \models$ male$(a) \vee$ female(a).
    Since it is not known whether a is male or female, the model theory for partial models with negation in the head should allow that neither male(a) nor female(a) belong to $\mathcal{I}$.
- this chapter: allow to define and compute $T_P(I)$ for rules with negation in the body:
  - evaluate conjunctive bodies with negation,
  - T_P for such rules: if the truth value of the body is u, that of the head should also be u.
  - an appropriate notion for $I \models P$ for partial interpretations wrt. such programs.

3-Valued Logic: Implication

For implication, there are different definitions (here, only two are listed):

1. Logic K$_3$, Stephen Kleene (1938):
   $A \to B = \neg A \vee B = \max(1 - A, B)$
   follows the definition of $\to$ as a derived operator from boolean logic.

A \ B	f	u	t
f	t	t	t
u	u	u	t
t	f	u	t

   - Fits with intuitive "if the truth value of the body is unknown and the truth value of the head is unknown, then the truth value of $A \to B$ is also unknown".
   - Does not fit with the intention to handle $\mathcal{I} \models head \leftarrow body$ where the truth value of the body (and that of the head) is u.

2. based on the ordering of the domain: $t > u > f$:
   $A \to B = (A \le B)$
   - the truth value of $A \to B$ is always t or f,
   - For a rule $head \leftarrow body$,
     if $I(body) = u$ and $I(head) = u$,
     then $I(head \leftarrow body) = t$.

$A : body$ \ $B : head$	f	u	t
f	t	t	t
u	f	t	t
t	f	f	t

$\Rightarrow$ use the second alternative.

3-Valued Logic: Notation and Minimal Models

Extend and adapt FOL notation:

- 3-valued Herbrand interpretations are given as tuples $I = (T, F)$ where T is the set of true atoms and F is the set of false atoms.
  All other atoms are undefined.
- $I_1 \le I_2$ is defined wrt. the amount of information:
  with partial order $u \prec t$ and $u \prec f$
  – $I_1 \le I_2$ if for all ground atoms a , $I_1(a) \preceq I_2(a)$,
  – or equivalently $(T_1, F_1) \le (T_2, F_2) \;\Leftrightarrow\; T_1 \subseteq T_2$ and $F_1 \subseteq F_2$.
- The minimal interpretation is thus formally correctly written as $(\emptyset, \emptyset)$.
- instead of $I \models \phi$ or $I \models_\beta \phi$ (which can only express true/false),
  write $I(\phi) = v$ or val$_{I,\beta}(\phi) = v$ for $v \in \{t, u, f\}$.
  Convention: write $I \models \phi$ ("I is a model of ϕ") in 3-valued context if $I(\phi) = t$.
  ($\models$ will only be applied to programs and rules, the semantics of $\to$ has been defined above to result in t or f.)

11.4 3-Valued Well-Founded Model

Given a program P, define a certain 3-valued Herbrand interpretation $I = (T, F)$ as follows;

Definition 11.5

For a Datalog$^\neg$ program P with $I_0 = \emptyset, I_1, \ldots, I_{2n}, I_{2n+1}, \ldots$ the Alternating Fixpoint Computation, let $\mathcal{W}_P := (\{a | a \in \lim_{n\to\infty} I_{2n}\}, \{a | a \in \mathcal{B}_P, a \notin \lim_{n\to\infty} I_{2n+1}\})$. □

- "true": all facts that are in the final underestimate of true atoms;
- "false": all facts that are outside of the final overestimate of true atoms – they are definitely false.

It will be proven later that $\mathcal{W}_P$ is the well-founded model of P (cf. Definition 11.3).

Example

Consider again the simple win-move game from Slide 649.

The corresponding program is $P =$

```
pos(a). pos(b). pos(c). pos(d).
move(a,b).  move(b,a).  move(b,c).  move(c,d).
win(X) :-  move(X,Y), not win(Y).
lose(X) :- pos(X), not win(X).
```

[Filename: Datalog/winmove-small.s]

With the sequence $((I_k))$ as given on Slide 649, the alternating fixpoint computation stops at $I_3 = I_1$ (EDB shown in gray):

$\mathcal{W}(P) =$ ({ pos(a), pos(b), pos(c), pos(d),
move(a,b), move(b,a), move(b,c), move(c,d), win(c), lose(d)},
{ move(a,a), move(a,c), move(a,d), move(b,a), move(b,b), move(b,d), move(c,a),
move(c,b), move(c,c), move(d,a), move(d,a), move(d,b), move(d,c), move(d,d),
win(d), lose(c)})

undefined: win(a), win(b), lose(a), lose(b)

(usually one omits the EDB predicates when listing well-founded or stable models).

3-VALUED T_P-OPERATOR

Definition 10.2 carries over to 3-valued interpretations as follows:

Definition 11.6 ($3T_P$-Operator)
For a ground Datalog$^\neg$ program P_g (which might contain the boolean atom undef in the body) and a 3-valued interpretation $I = (T, F)$, for each ground atom a,

$$3T_{P_g}(I)(a) := \max(\{I(\mathit{body}) : \ a \leftarrow \mathit{body} \in P_g\}$$

For a non-ground Datalog$^\neg$ program P and a 3-valued interpretation $I = (T, F)$, $3T_P(I) := 3T_{P_g}(I)$ where P_g is the grounding of P wrt. the Herbrand Universe of P (i.e., the set of all possible ground instances of the rules of P).

$$\begin{aligned}
3T_P^0(I) &:= I \\
3T_P^1(I) &:= 3T_P(I) \\
3T_P^{n+1}(I) &:= 3T_P(3T_P^n(I)) \\
3T_P^\omega(I) &:= \bigcup_{n \in \mathbb{N}} 3T_P^n(I) \\
3T_P^\omega &:= 3T_P^\omega(\emptyset, \emptyset).
\end{aligned}$$

□

3-VALUED REDUCT

Definition 11.1 (Slide 631) carries over to 3-valued interpretations as follows:

Definition 11.7 (3-Valued Reduct)
For a Datalog$^\neg$ program P, and a 3-valued interpretation $I = (T, F)$, the reduct P^I of P wrt. I is obtained as follows:

- *let P_g denote the grounding of P,*
- *delete from P_g all rules that contain a negative literal $\neg a$ in the body such that $I(a) = t$,*
- *replace all negative literals $\neg a$ in the remaining rules s.t. $I(a) = u$ by the boolean atom undef (since undef is neither in T nor in F it will be evaluated as I(undef) $= u$),*
- *delete all remaining negative literals in the bodies of the remaining rules.*

□

Properties of P^I

- P^I is a ground positive program.
- If I is a model of P, then for each ground atom a, $(3T_{P^I}^\omega(\emptyset))(a) \leq I(a)$.

3-STABLE MODELS

Definition 11.8
A 3-valued interpretation $I = (T, F)$ is a 3-stable *model of a Datalog$^\neg$ program P, if*

$$3T^\omega_{P^I}(\emptyset, \emptyset) = I.$$

□

For returning also partial models, invoke smodels with `--partial`.

- output `p(a)` means that $p(a)$ can be derived to be true
- output `p'(a)` means that $val(p(a) \geq u$ is at least undefined (`p(a)` might also be listed to be true)
- this avoids to have to list all possible ground instantiations of atoms that are false.

(see next slide)

Example/Syntax: Partial Stable Model in smodels

Example 11.1

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

[Filename: Datalog/pnotp.s]

... has only one partial stable model: $p(a)$ is undefined:

```
lparse -n 0 --partial pnotp.s|smodels
smodels version 2.34. Reading...done
Answer: 1
Stable Model: p'(a)
```

Interpretation of the result $M = \{p'(a)\}$ (smodels Section 4.8.2):

- *for every ground atom $p(\ldots)$, an atom $p'(\ldots)$ is added to the internal program, which means "$p(\ldots)$ is potentially true"*
- *if both $p(\ldots)$ and $p'(\ldots)$ are in M, then $val_M(p(\ldots)) = t$,*
- *if $p'(\ldots) \in M$ and $p(\ldots) \notin M$, then $val_M(p(\ldots)) = u$,*
- *otherwise $val_M(p(\ldots)) = f$.*

□

Example

Example 11.2
Consider once more the program from Slide 634:

```
q(a) :- not p(a).
p(a) :- not q(a).
```
[Filename: Datalog/porq.s]

Exercise: give the Alternating Fixpoint Computation for P.

- *$\mathcal{S} := (\emptyset, \emptyset)$, i.e., $\mathcal{S}(p(a)) = \mathcal{S}(q(a)) = u$, is a 3-stable model. It is the minimal 3-stable model.*
- *On Slide 634, $\{p(a)\}$ and $\{q(a)\}$ have been identified as total stable models of P.*
  Note: as partial models, these are written as (T, F)-pairs as $(\{p(a)\}, \{q(a)\})$ and $(\{q(a)\}, \{p(a)\})$ □

Example 11.3
Consider winmove.p with –partial.

- *here, the unique partial stable model (= the well-founded model) is the "intended" one with drawn positions.*
- *the total stable models arbitrarily "fix" some drawn positions to be won/lost (in an admissible way wrt. the program).* □

Well-Founded Model

Recall Definition 11.3 (638):

For a Datalog$^\neg$ program P, the (in general three-valued) *w*ell-founded model of P is the (unique) minimal 3-stable model of P.

Theorem 11.2
$\mathcal{W}_P$ (as defined on Slide 656) is the well-founded model of P. □

Proof:

- Show that $\mathcal{W}_P$ is 3-stable [Abiteboul, Hull, Vianu: Foundations of Databases, Thm. 15.3.9]
- minimality and uniqueness follow from Lemma 11.2:

Lemma 11.2
For a Datalog$^\neg$ program P, $\mathcal{W}_P = (T, F)$ is the intersection of all 3-stable models of P, i.e., for every 3-stable model (T', F'), $T' \supseteq T$ and $F' \supseteq F$. □

Proof: minimality of T wrt. all models and minimality of F wrt. all stable models follows from the properties proven for the AFP computation.

Comments: Well-Founded Model

- The AFP gives a (polynomial!) computation for the non-constructive definition of “well-founded model”.
- all stable models extend the well-founded model
  $\Rightarrow$ computation/guessing can be based on the well-founded model.
- starting the Alternating Fixpoint Computation with the contents of the EDB relations as initial interpretation J_0 leads to the same final result
  (but the intermediate J_i are different and J_0 serves as an underestimate).

Recall: Non-Monotonicity of Closed-World-Assumption

“Negation by default” is non-monotonous:

Consider a program P and its well-founded model $\mathcal{W}(P) = (T, F)$:

- recall that any program (we have only positive atoms in the head) cannot imply that any atom *must be* false in all models
  $\Rightarrow$ any positive fact can be added to a Datalog/Datalog$^\neg$ program without being inconsistent.
- there are non-stable models $\mathcal{M} = (T', F')$ of P where $T' - T \neq \emptyset$ (containing atoms that are not supported by P), and for these, often also $F - F' \neq \emptyset$
  - e.g. add an edge to the win-move game, and some other positions are won, but some that were won before are now lost, or
  - e.g. just fix that a certain (drawn or even lost) position is won.
  - $F - F' \neq \emptyset \Rightarrow$ Things that have been concluded before to hold do now turn out not to hold; “Belief Revision”.
- $\mathcal{M}$ is then a 3-stable model of a (more or less slightly) different program $P' \supsetneq P$.
  (e.g., $P' = P \cup \{$move(x,y)$\}$ or $P' = P \cup \{$win(x)$\}$)
  $\Rightarrow$ corresponds to “learning” about a new fact,
  $\Rightarrow$ requires to recompute the whole well-founded model from scratch.

Exercise: Well-Founded Model

- show that for every positive Datalog program P, the well-founded model is *total* (i.e., all ground atoms are either true or false).
- show that for every stratifiable Datalog$^{\neg}$ program P, the well-founded model is *total*.

Exercise: Well-Founded Model

- Are there non-stratifiable Datalog$^{\neg}$ programs that have a total well-founded model (i.e., no atoms undefined)?
- Are there (non-ground) non-stratifiable Datalog$^{\neg}$ programs that have a total well-founded model for *all* EDB instances?

Well-founded Semantics: Literature

- definition of reduct and stable model taken from documentation of smodels,
- alternating fixpoint taken from ??TO BE EXTENDED??
- further reading: [Abiteboul, Hull, Vianu: Foundations of Databases]
- Original Paper: Allen Van Gelder, Kenneth A. Ross, John S. Schlipf: Unfounded Sets and Well-Founded Semantics for General Logic Programs. PODS 1988: 221-230
- Long version: Allen Van Gelder, Kenneth A. Ross, John S. Schlipf: The Well-Founded Semantics for General Logic Programs. J. ACM 38(3): 620-650 (1991)
- Alternating Fixpoint: Allen Van Gelder: The Alternating Fixpoint of Logic Programs with Negation. PODS 1989: 1-10
- online literature database (started with database + logic programming, now for everything in CS): `http://dblp.uni-trier.de/`
  (from university computers, access to most pdfs is allowed)

Restrictions of the Datalog/Minimal/Well-founded Model Semantics

Given a Datalog/Datalog$^\neg$ program P, the minimal model, well-founded model, and the AFP procedure cannot decide the following:

- for a given general FOL formula ϕ, does ϕ hold in *all* models of P?
- if $p(c_1, \ldots, c_n)$ can not be confirmed by the minimal, stratified, or well-founded model, this does *not* mean, that there is no model of P where $p(c_1, \ldots, c_n)$ holds.
  Even more, any positive fact can be added to a Datalog/Datalog$^\neg$ program without being inconsistent.

Closed-World-Assumption (CWA)

- For all facts that are not given in the database and that are not derivable, it is assumed that they do not hold (more explicitly: that their negation holds).
- CWA not appropriate in the Web: for things that I do not find in the Web, simply nothing is said.
  [Example: travel planning]

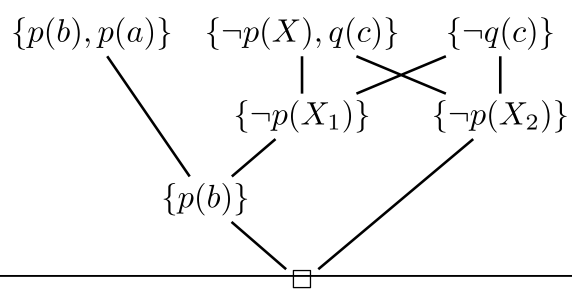
The Limits – no real Disjunction

```
:- auto_table.
p(a) :- tnot p(b).
p(b) :- tnot p(a).
q(c) :- p(X).
```

[Filename: Datalog/pq.P]

```
?-p(X).
X = b undefined
X = a undefined
?- q(X).
X = c undefined
X = c undefined
```

- "$q(c)$ undefined" is computed twice by SLG resolution, i.e. two proof paths exist.
- $\mathcal{W}(P) = (\emptyset, \{q(a), q(b), p(c)\})$, the "interesting" ground atoms $\{p(a), p(b), q(c)\}$ are undefined.
  The model theories of the minimal model and well-founded model define truth/entailment only for ground atoms.
- P as a FOL formula: $(p(b) \vee p(a)) \wedge \forall x : p(x) \rightarrow q(c) \models_{\text{FOL}} q(c)$
- (general) resolution proof: clauses $\{p(a), p(b)\}$ (which is the clause corresponding to both the two first rules) and $\{\neg p(X), q(c)\}$ together with query/goal clause $\neg q(c)$ allow to derive $\Box$:



- SLD/SLG resolution tries only linear proofs.

The Limits – no real Disjunction

The same program interpreted by stable models:

```
thing(a). thing(b). thing(c).
p(a) :- not p(b).
p(b) :- not p(a).
q(c) :- thing(X), p(X).
```

[Filename: Datalog/pq.s]

```
lparse -n 0 --partial pq.s|smodels
models version 2.34. Reading...done
Answer: 1
Stable Model: p(a) q(c)
Answer: 2
Stable Model: p(b) q(c)
## Answer: 3
## Stable Model: p'(a) p'(b) q'(c)
False
```

- two total stable models:
  - "either $p(a)$ or $p(b)$ hold"
  - "$q(c)$ holds in any case"
- the user can interpret the result as a 3-valued interpretation I where

  $val_I(p(a)) = val_I(p(b)) = u$ and $val_I(q(c)) = t$.

  I is a model of P (i.e., $3T^{\omega}_{P_I}(\emptyset) \leq I$), but I is *not* a *stable* model of P (i.e., $3T^{\omega}_{P_I}(\emptyset) \neq I$)!