Contents

1 Motivation

Nonmonotonic Queries

- Some simple queries cannot be written in positive Datalog.
- Example: $(\pi_1\ R) - S$
- This query is *nonmonotone*!
- Adding tuples to S may retract result tuples.
- Positive Datalog can express only monotone queries.

Nonmonotonic Queries

- In Relational Calculus $(\pi_1\ R) - S$ is written using negation.
- Introduce negation also for Datalog!
- *Problem*: Negation through recursion?

1.1 Introducing Negation

Closed World Assumption

- Atoms for which it is not necessary to be true should be considered as false.
- Only those items which are known should be true.
- Example: Timetable
- Reason for Minimal Model semantic!

Closed World Assumption

Definition 1. For a positive program $\mathcal{P}$, $CWA(\mathcal{P}) = \{A \mid \mathcal{P} \not\models A\}$.
Equivalently: $CWA(\mathcal{P}) = \mathbf{HB}(\mathcal{P}) - MM(\mathcal{P})$

Is this as simple if we allow rules with negative body literals?

1.2 Normal Programs

Normal Programs – Syntax

Definition 2. A *normal* rule is

$$h \leftarrow b_1, \ldots, b_m, \texttt{not}\ b_{m+1}, \ldots, \texttt{not}\ b_n.$$
$$1 \leq m \leq n$$

Let
$B^+(r) = \{b_1, \ldots, b_m\}$
$B^-(r) = \{b_{m+1}, \ldots, b_m\}$
$\texttt{not}.a = \texttt{not}\ a, \texttt{not}.\texttt{not}\ a = a$
$\texttt{not}.L = \{\texttt{not}.l \mid l \in L\}$
$B(r) = B^+(r) \cup \texttt{not}.B^-(r)$
$H(r), V(r), C(r)$ as before

Unsafe Queries
Recall: Using Negation it is easy to violate domain independence!

Example 3.
$$positive(X) \leftarrow \texttt{not}\ zero(X).$$

Definition 4 (Safety)**.** Each variable in a rule must occur in a positive body atom.

Example 5.
$$answer(X) \leftarrow mynumber(X), \texttt{not}\ zero(X).$$

1.3 Semantics

Normal Programs – Semantics

- Most concepts do not change.
- Satisfaction of a rule r with respect to M: If $B^+(r) \subseteq M$ and $M \cap B^-(r) = \emptyset$, then $H(r) \in M$
- Question: Minimal Model semantics suitable?

Normal Programs

In general there is no unique minimal model.

Example 6.

$$a \leftarrow \mathtt{not} b.$$

There are two models $M_1 = \{a\}$ und $M_2 = \{b\}$.
M_2 is not very intuitive.

Normal Programs

Semantics of "negative recursion"?

$$\begin{array}{l} person(nicola). \\ male(X) \leftarrow person(X), \mathtt{not}\ female(X). \\ female(X) \leftarrow person(X), \mathtt{not}\ male(X). \end{array}$$

$\{person(nicola), male(nicola)\}$ and $\{person(nicola), female(nicola)\}$ are minimal models

Both are equally intuitive.

Possibilities

1. Pragmatic: Do not allow "recursion through negation".
2. Three-valued: Stay with a unique model, which may leave some atoms undefined.
3. Two-valued: Abandon model uniqueness, stay with standard models.

2 Stratifiable Programs

2.1 Dependency Graph

Dependency Graph

Definition 7. For a negative Datalog program $\mathcal{P}$, we define a directed graph (V, E), where V are the predicate symbols of $\mathcal{P}$, and $(p, q) \in E$ if p is in the head and q is in the body of some rule. If q is in the negative body, we mark the arc.

Examples

Example 8.

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Example 9.

$$a \leftarrow b, c.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

2.2 Stratification

Stratification

Main idea: Partition the program along negation.

Definition 10. A stratification is a function λ, which maps predicate symbols to integers such that for each rule with p being the head predicate the following conditions hold:

1. For each predicate q in the positive body, $\lambda(p) \geq \lambda(q)$.
2. For each predicate r in the negative body, $\lambda(p) > \lambda(r)$.

Stratification

- λ induces a partition $\langle P_0, \ldots, P_n \rangle$ of $\mathcal{P}$ (assuming that λ maps to integers between 0 and n):
$$P_0 = \{r \mid \lambda(H(r)) = 0$$
$$\ldots$$
$$P_n = \{r \mid \lambda(H(r)) = n$$
- λ defines a partial ordering between partitions.
- We can evaluate the program along this ordering.

Examples

Example 11.

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Stratifiable: $\lambda(a) = 0, \lambda(b) = 0, \lambda(c) = 1$

Example 12.

$$a \leftarrow b, c.$$
$$c \leftarrow \texttt{not}\ b.$$
$$b \leftarrow a$$

Not stratifiable: $\lambda(c) > \lambda(b) \geq \lambda(a) \geq \lambda(c)$

Stratification

Theorem 13. *A program is stratifiable if and only if its dependency graph contains no cycle with a marked ("negative") edge.*

2.3 Perfect Models

Perfect Models

- Stratification specifies an order for evaluation.
- First fully compute the relations in the lowest stratum.
- Then move one stratum up and evaluate the relations there.
- Negation is evaluated only over fully computed relations.
- Can be treated like negation over EDB predicates.

Perfect Models und $\mathbf{T}_{\mathcal{P}}$

Modify operator $\mathbf{T}_{\mathcal{P}}$, as $\mathcal{P}$ may contain negation.

Definition 14.

$$\mathbf{T}_{\mathcal{P}}(I) = \{h \mid r \in Ground(\mathcal{P}), B^{+}(r) \subseteq I, h \in H(r), \\ \texttt{not}.B^{-}(r) \cap I = \emptyset\} \cup I$$

Perfect Models und $\mathbf{T}_{\mathcal{P}}$

Definition 15. Let $\langle P_0, \ldots, P_n \rangle$ be the partitions of a stratifiable program $\mathcal{P}$, induced by a stratification λ.

The sequence $M_0 = \mathbf{T}^{\infty}_{P_0}(\emptyset)$, $M_1 = \mathbf{T}^{\infty}_{P_1}(M_0)$, ..., $M_n = \mathbf{T}^{\infty}_{P_n}(M_{n-1})$ defines the *Perfect Model* M_n of $\mathcal{P}$.

Example – stratifiable

Easy case: Negation only on EDB predicates

Example 16.

$$color(yellow, k1).\ color(yellow, k2).\ color(blue, k3). \\ color(green, k4).\ color(red, k5).$$

$$block(K) \leftarrow color(F, K). \quad block(K) \leftarrow form(F, K). \\ diffcolor(K1, K2) \leftarrow \\ color(F, K1), block(K2), \texttt{not}\ color(F, K2).$$

Example – stratifiable

Example 17.

$$form(box, k1).\ form(cone, k2).\ form(disc, k3).$$
$$form(box, k4).\ form(pyramid, k5).$$

$$block(K) \leftarrow color(F, K). \quad block(K) \leftarrow form(F, K).$$
$$pointy\_top(K) \leftarrow block(K), form(cone, K).$$
$$pointy\_top(K) \leftarrow block(K), form(pyramid, K).$$
$$fits\_on(K1, K2) \leftarrow block(K1), block(K2), \texttt{not}\ pointy\_top(K2).$$

Example – stratifiable

Example 18.

$$form(box, k1).\ form(cone, k2).\ form(disc, k3).$$
$$form(box, k4).\ form(pyramid, k5).$$

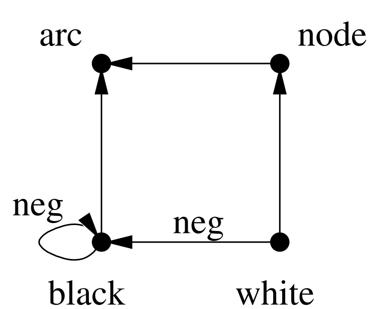
$$block(K) \leftarrow color(F, K). \quad block(K) \leftarrow form(F, K).$$
$$flat\_top(K) \leftarrow block(K), form(box, K).$$
$$flat\_top(K) \leftarrow block(K), form(disc, K).$$
$$pointy\_top(K) \leftarrow block(K), \texttt{not}\ flat\_top(K).$$
$$fits\_on(K1, K2) \leftarrow block(K1), block(K2), \texttt{not}\ pointy\_top(K2).$$

Example – unstratified

$$arc(a, b).\ arc(b, c).\ arc(b, d).$$
$$node(N) \leftarrow arc(N, Y).\ node(N) \leftarrow arc(X, N).$$
$$black(Y) \leftarrow arc(X, Y), \texttt{not}\ black(X).$$
$$white(X) \leftarrow node(X), \texttt{not}\ black(X).$$

Example – unstratified

Dependency Graph:



Perfect Models

- *Note*: Perfect Models are defined only on stratifiable programs.

Theorem 19. *For any stratifiable program, there exists a unique Perfect Model.*

Unstratifiable Programs

Example 20.

$$\begin{array}{l} person(nicola). \\ alive(X) \leftarrow person(X). \\ male(X) \leftarrow person(X), \texttt{not}\ female(X). \\ female(X) \leftarrow person(X), \texttt{not}\ male(X). \end{array}$$

Perfect Models are not defined.
But we would like to conclude at least $alive(nicola)$.

3 Recursive Negation

Recursive Negation

Example 21.

$$\begin{array}{l} person(nicola). \\ alive(X) \leftarrow person(X). \\ male(X) \leftarrow person(X), \texttt{not}\ female(X). \\ female(X) \leftarrow person(X), \texttt{not}\ male(X). \end{array}$$

Recursive Negation

Example 22. Using generalized $\mathbf{T}_{\mathcal{P}}$:

$$\begin{array}{l} \mathbf{T}_{\mathcal{P}}(\emptyset) = \{person(nicola)\} \\ \mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset)) = \{person(nicola), alive(nicola), male(nicola), female(nicola)\} \\ \mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset))) = \{person(nicola), alive(nicola)\} \\ \mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset)))) = \mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset)) \\ \mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\mathbf{T}_{\mathcal{P}}(\emptyset))))) = \mathbf{T}_{\mathcal{P}}(\emptyset) \\ \ldots \end{array}$$

Recursive Negation

Example 23. But there are two fixpoints:

$$\begin{array}{l} \mathbf{T}_{\mathcal{P}}(\{person(nicola), alive(nicola), male(nicola)\}) = \\ \quad \{person(nicola), alive(nicola), male(nicola)\} \\ \mathbf{T}_{\mathcal{P}}(\{person(nicola), alive(nicola), female(nicola)\}) = \\ \quad \{person(nicola), alive(nicola), female(nicola)\} \end{array}$$

Recursive Negation

Two ways of resolving this:

1. Be cautious and do not say anything about $male(nicola)$ and $female(nicola)$.
2. Consider two scenarios: One in which $male(nicola)$ is true, another in which $female(nicola)$ is true.

Problems to resolve:

1. needs another truth value *undefined*.
2. allows more than one model.

4 Well-founded Models

4.1 Unfounded Sets

Three-valued Interpretations

Definition 24. A *three-valued* (or *partial*) interpretation I is a set of ground `not` literals, such that for any ground atom a not both $a \in I$ and $\texttt{not}a \in I$.

Example 25. $I = \{\texttt{not}\ a, c\}$

- a is false in I
- b is undefined in I
- c is true in I

Unfounded Sets

Goal: Derive as much negative information as possible.

Example 26.

$$a \leftarrow \texttt{not}\ b.$$

b does not occur in any head, thus can never become true und should be false. a should therefore be true.

Unfounded Sets

Goal: Derive as much negative information as possible.

Example 27.

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ a.$$

Given the interpretation $\{\texttt{not}\ b\}$, a can never become true and should be false. c should be true in this case.

Unfounded Sets

Goal: Derive as much negative information as possible.

Example 28.

$$a \leftarrow b.$$
$$b \leftarrow a.$$
$$c \leftarrow \texttt{not}\ a.$$

a and b occur in some heads, but all bodies of these rules require one of a or b to become true. Therefore a and b can become true only via themselves and should be false, hence c should be true.

Unfounded Sets – Definition

Definition 29. A set $U \subseteq \mathbf{HB}(\mathcal{P})$ is *unfounded* with respect to a partial interpretation I if the following holds:

For each $a \in U$ and each rule $r \in Ground(\mathcal{P})$ with $H(r) = \{a\}$ at least one of the the following conditions holds:

1. $\exists \ell \in B(r) : \texttt{not}.\ell \in I$
2. $B^+(r) \cap U \neq \emptyset$

Unfounded Sets – Example

Example 30.

$$a \leftarrow \texttt{not}\ b.$$

For $I = \emptyset$, $\{b\}$ is an unfounded set.

Unfounded Sets – Example

Example 31.

$$a \leftarrow b.$$
$$c \leftarrow \texttt{not}\ a.$$

For $I = \{\texttt{not}\ b\}$, $\{a\}$ is an unfounded set.

Unfounded Sets – Example

Example 32.

$$a \leftarrow b.$$
$$b \leftarrow a.$$
$$c \leftarrow \texttt{not}\ a.$$

For $I = \emptyset$, $\{a, b\}$ is an unfounded set, because condition 2 holds for $a \leftarrow b.$ and $b \leftarrow a..$

4.2 Well-founded Operator

Unfounded Operator

Theorem 33. *For any program $\mathcal{P}$ and partial interpretation I, the greatest unfounded set $GUS_{\mathcal{P}}(I)$ (which is a superset of all unfounded sets) exists and is unique.*

Idea: Use $GUS_{\mathcal{P}}(I)$ to derive negative information.

Definition 34. Operator $\mathbf{U}_{\mathcal{P}}(I) = \{\texttt{not}.a \mid a \in GUS_{\mathcal{P}}(I)\}$

Well-Founded Operator

First generalize $\mathbf{T}_{\mathcal{P}}(I)$ for partial interpretations:

Definition 35. $\mathbf{T}_{\mathcal{P}}(I) := \{h \mid r \in Ground(\mathcal{P}), B(r) \subseteq I, h \in H(r)\}$

Define the *well-founded* operator $\mathbf{W}_{\mathcal{P}}(I)$ as a combination of $\mathbf{T}_{\mathcal{P}}(I)$ and $\mathbf{U}_{\mathcal{P}}(I)$.

Definition 36. $\mathbf{W}_{\mathcal{P}}(I) = \mathbf{T}_{\mathcal{P}}(I) \cup \mathbf{U}_{\mathcal{P}}(I)$

Well-Founded Model




Allen Van Gelder

Kenneth Ross John Schlipf

Well-Founded Model

Theorem 37. $\mathbf{W}_{\mathcal{P}}$ *is monotone and allows for a least fixpoint.*

Definition 38. The least fixpoint $\mathbf{W}_{\mathcal{P}}^{\infty}(\emptyset)$ is the Well-Founded Model of a normal program $\mathcal{P}$.

Well-Founded Model – Properties

Theorem 39. *Each normal program has a unique Well-Founded Model.*

Well-Founded Model – Properties

Definition 40. A partial interpretation I is total if $I \cup \texttt{not}.I = \mathbf{HB}(\mathcal{P})$ (each ground atom is true or false).

Theorem 41. *The Well-Founded Model for positive programs is total and corresponds to its Minimal Model.*

Theorem 42. *The Well-Founded Model for stratifiable programs is total and corresponds to its Perfect Model.*

Well-Founded Model – Example

Example 43.

$$
\begin{array}{l}
person(nicola). \\
alive(X) \leftarrow person(X). \\
male(X) \leftarrow person(X), \texttt{not}\ female(X). \\
female(X) \leftarrow person(X), \texttt{not}\ male(X).
\end{array}
$$

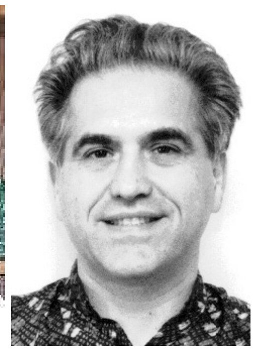
The Well-Founded Model is $\{person(nicola), alive(nicola)\}$ and it is not total.

5 Stable Models

Stable Models

- No longer a unique model.
- Use total models.
- Stability criterion instead of fixpoint semantics.

Stable Models

  Michael Gelfond

Vladimir Lifschitz

Stable Models


Nicole Bidoit Christine Froidevaux

5.1 Gelfond-Lifschitz Reduct

Gelfond-Lifschitz Reduct

Definition 44. The *Gelfond-Lifschitz reduct* of a program $\mathcal{P}^I$ is defined as follows, starting from $Ground(\mathcal{P})$:

1. Delete rules r, for which $B^-(r) \cap I \neq \emptyset$.
2. Delete the negative bodies of the remaining rules.

Gelfond-Lifschitz Reduct

Example 45.

$$\mathcal{P} = \{ \, male(g) \leftarrow \texttt{not}\ female(g). \quad female(g) \leftarrow \texttt{not}\ male(g). \}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{male(g).\ female(g).\}$ $I_2 = \{male(g)\}$, $\mathcal{P}^{I_2} = \{male(g).\}$ $I_3 = \{female(g)\}$, $\mathcal{P}^{I_3} = \{female(g).\}$ $I_4 = \{male(g), female(g)\}$, $\mathcal{P}^{I_4} = \emptyset$

Stable Models

Fact 46. *Gelfond-Lifschitz reducts are always positive, and have a unique Minimal Model.*

Definition 47. A total interpretation M is a Stable Model of $\mathcal{P}$, if $M = MM(\mathcal{P}^M)$.

Stable Models – Example

Example 48.

$$\mathcal{P} = \{ \, male(g) \leftarrow \texttt{not}\ female(g). \quad female(g) \leftarrow \texttt{not}\ male(g). \}$$

$I_1 = \emptyset$, $\mathcal{P}^{I_1} = \{male(g).\ female(g).\}$, $MM(\mathcal{P}^{I_1}) \neq I_1$ $I_2 = \{male(g)\}$, $\mathcal{P}^{I_2} = \{male(g).\}$, $MM(\mathcal{P}^{I_2}) = I_2$ I_2 is a stable model. $I_3 = \{female(g)\}$, $\mathcal{P}^{I_3} = \{female(g).\}$, $MM(\mathcal{P}^{I_3}) = I_3$ I_3 is a stable model. $I_4 = \{male(g).\ female(g)\}$, $\mathcal{P}^{I_4} = \emptyset$, $MM(\mathcal{P}^{I_4}) \neq I_4$

Stable Models – Example

Example 49.

$$\mathcal{P} = \{\ weird \leftarrow \texttt{not}\ weird.\}$$

$I_1 = \emptyset, \mathcal{P}^{I_1} = \{weird.\}, MM(\mathcal{P}^{I_1}) \neq I_1$ $I_2 = \{weird\}, \mathcal{P}^{I_2} = \emptyset, MM(\mathcal{P}^{I_2}) \neq I_2$
There is no stable model!

Stable Models

Theorem 50. *For positive programs there is exactly one Stable Model, which is equal to the Minimal Model.*

Theorem 51. *For stratifiable programs there is exactly one Stable Model, which is equal to the Perfect Model.*

Stable Models

Theorem 52. *If the Well-Founded Model of a program is total, then the program has a corresponding unique Stable Model.*

Theorem 53. *The positive part of the Well-Founded Model of a program is contained in each Stable Model of the program.*

Stable Models – Consequences

Definition 54 (Brave/Credulous Reasoning)**.** $\mathcal{P} \models_b l$, if l is true in some Stable Model of $\mathcal{P}$.

Definition 55 (Cautious/Skeptical Reasoning)**.** $\mathcal{P} \models_c l$, if l is true in all Stable Models of $\mathcal{P}$.

Note: If $\mathcal{P}$ admits no Stable Model, then all literals are cautious/skeptical consequences!

Stable Models – Example

Example 56 (Two-Colorability). Given a graph, can each vertex be assigned one of two colors, such that adjacent vertices do not have the same color?

$$vertex(V) \leftarrow arc(V,Y).\ vertex(V) \leftarrow arc(X,V).$$
$$color(V,white) \leftarrow vertex(V), \texttt{not}\ color(V,black).$$
$$color(V,black) \leftarrow vertex(V), \texttt{not}\ color(V,white).$$
$$bad \leftarrow color(V1,F), color(V2,F),$$
$$arc(V1,V2), \texttt{not}\ bad.$$