Ontology and Database Systems: Foundations of Database Systems

Part 6: Datalog with Negation

Werner Nutt

Faculty of Computer Science
Master of Science in Computer Science

A.Y. 2015/2016

Freie Universität Bozen
Libera Università di Bolzano
Università Liedia de Bulsan

The Issue

- In Relational Calculus and Relational Algebra, we have negation ($\neg$) as an operator
- Thus, queries like the complement of a relation or the difference between two relations are easily expressible
- These queries can not be expressed in datalog (monotonicity)
- $\leadsto$ Extension of datalog with negation!

Example

$$\mathit{ready}(D) \leftarrow \ \mathit{device}(D), \neg \mathit{busy}(D)$$

Giving a semantics is not straightforward because of possible cyclic definitions:

Example

$$\begin{aligned} \mathit{single}(X) &\leftarrow \mathit{man}(X), \neg \mathit{husband}(X) \\ \mathit{husband}(X) &\leftarrow \mathit{man}(X), \neg \mathit{single}(X) \end{aligned}$$

Datalog$^\neg$ Syntax

Definition

A *datalog$^\neg$ program* P is a finite set of datalog$^\neg$ rules r of the form

$$A \leftarrow B_1, \ldots, B_n \tag{1}$$

where $n \geq 0$ and

- A is an atom $R_0(\vec{x}_0)$
- each B_i is an atom $R_i(\vec{x}_i)$ or *a negated atom* $\neg R_i(\vec{x}_i)$
- $\vec{x}_0, \ldots, \vec{x}_n$ are tuples of variables and constants (from **dom**)
- every variable in $\vec{x}_0, \ldots, \vec{x}_n$ must occur in some atom $B_i = R_i(\vec{x}_i)$ ("safety")
- the head of r is A, denoted $H(r)$
- the body of r is $\{B_1, \ldots, B_n\}$, denoted $B(r)$, and
  $B^+(r) = \{R(\vec{x}) \mid \exists i\, B_i = R(\vec{x})\}$, $B^-(r) = \{R(\vec{x}) \mid \exists i\, B_i = \neg R(\vec{x})\}$

P has extensional and intensional relations, *edb*(P) resp. *idb*(P), like a datalog program.

Datalog$^\neg$ Semantics – First Attempt

- **Idea:** Naturally extend the minimal-model semantics of datalog (equivalently, the least fixpoint-semantics) to negation
- Generalize to this aim the immediate consequence operator

$$\mathbf{T}_P(\mathbf{K}) : \mathit{inst}(\mathit{sch}(P)) \to \mathit{inst}(\mathit{sch}(P))$$

Definition

Given a datalog$^\neg$ program P and $\mathbf{K} \in \mathit{inst}(\mathit{sch}(P))$,
a fact $R(\vec{t})$ is an *immediate* consequence for $\mathbf{K}$ and P, if either

- $R \in \mathit{edb}(P)$ and $R(\vec{t}) \in \mathbf{K}$, or
- there exists some ground instance r of a rule in P such that
  - $H(r) = R(\vec{t})$,
  - $B^+(r) \subseteq \mathbf{K}$, and
  - $B^-(r) \cap \mathbf{K} = \emptyset$

  (that is, evaluate "$\neg$" w.r.t. $\mathbf{K}$)

Problems with Least Fixpoints

Natural trial: Define the semantics of datalog$^\neg$ in terms of least fixpoint of $\mathbf{T}_P$.
However, this suffers from several problems:

1. $\mathbf{T}_P$ may not have a fixpoint:

$$P_1 = \{\, \mathit{known}(a) \leftarrow \neg \mathit{known}(a) \,\}$$

2. $\mathbf{T}_P$ may not have a least (i.e., single minimal) fixpoint:

$$P_2 = \{ \quad \begin{array}{r} \mathit{single}(X) \leftarrow \mathit{man}(X), \neg \mathit{husband}(X) \\ \mathit{husband}(X) \leftarrow \mathit{man}(X), \neg \mathit{single}(X) \end{array} \quad \}$$

$\mathbf{I} = \{\mathit{man}(\mathit{dilbert})\}$

3. The least fixpoint of $\mathbf{T}_P$ including $\mathbf{I}$ may not be constructible by fixpoint iteration (i.e., not as limit $\mathbf{T}_P^\omega(\mathcal{I})$ of $\{\mathbf{T}_P^i(\mathbf{I})\}_{i\geq 0}$):

$$P_3 = P_2 \cup \{\mathit{husband}(X) \leftarrow \neg \mathit{husband}(X), \mathit{single}(X)\}$$

$\mathbf{I} = \{\mathit{man}(\mathit{dilbert})\})$ as above

Note: The operator $\mathbf{T}_P$ is not monotonic!

Problems with Minimal Models

There are similar problems for model-theoretic semantics

- We can associate with P naturally a first-order theory Σ_P as in the negation-free case (write rules as implications):

$$R(\vec{x}) \leftarrow (\neg)R_1(\vec{x}_1), \ldots, (\neg)R_n(\vec{x}_n)$$

$$\rightsquigarrow$$

$$\forall\vec{x}\forall\vec{x}_1 \cdots \forall\vec{x}_n(((\neg)R_1(\vec{x}_1) \land \cdots \land (\neg)R_n(\vec{x}_n)) \rightarrow R(\vec{x}))$$

- Still, $\mathbf{K} \in \mathit{inst}(\mathit{sch}(P))$ is a model of Σ_P iff $\mathbf{T}_P(\mathbf{K}) \subseteq \mathbf{K}$ (and models are not necessarily fixpoints)
- However, multiple minimal models of Σ_P containing $\mathcal{I}$ might exist (*dilbert* example).

unibz

Solution Approaches

Different kinds of proposals have been made to handle the problems above

- **Give up single fixpoint/model semantics:** Consider alternative fixpoints (models), and define results by *intersection*, called *certain semantics*.
  Most well-known: Stable model semantics (Gelfond & Lifschitz, 1988;1991).
  Still suffers from 1.
- **Constrain the syntax of programs:** Consider only fragment where negation can be "naturally" evaluated to a single minimal model.
  Most well-known: semantics for stratified programs (Apt, Blair & Walker, 1988), perfect model semantics (Przymusinski, 1987).

Solution Approaches/2

- **Give up 2-valued semantics:** Facts might be true, false or *unknown*
  Adapt and refine the notion of immediate consequence.
  Most well-known: Well-founded semantics (Ross, van Gelder & Schlipf, 1991).
  Resolves all problems 1-3

- **Give up fixpoint/minimality condition:** Operational definition of result.
  Most well-known: Inflationary semantics (Abiteboul & Vianu, 1988)

Semi-Positive Datalog

"Easy" case: Datalog$^\neg$ programs where negation is applied only to *edb* relations.

- Such programs are called *semi-positive*
- For a semi-positive program, the operator $\mathbf{T}_P$ is monotonic if the *edb*-part is fixed, i.e.,

  $$\mathbf{I} \subseteq \mathbf{J} \text{ and } \mathbf{I}|\mathit{edb}(P) = \mathbf{J}|\mathit{edb}(P) \text{ implies } \mathbf{T}_P(\mathbf{I}) \subseteq \mathbf{T}_P(\mathbf{J})$$

Theorem

Let P be a semi-positive datalog program and $\mathbf{I} \in \mathit{inst}(\mathit{sch}(P))$. Then,

1. $\mathbf{T}_P$ has a unique minimal fixpoint $\mathbf{J}$ among all $\mathbf{I}$ such that $\mathbf{I}|\mathit{edb}(P) = \mathbf{J}|\mathit{edb}(P)$.
2. Σ_P has a unique minimal model $\mathbf{J}$ among all $\mathbf{I}$ such that $\mathbf{I}|\mathit{edb}(P) = \mathbf{J}|\mathit{edb}(P)$.

unibz

Example

Semi-positive datalog can express

the transitive closure of the complement of a graph G:

$$\mathit{neg\_tc}(x, y) \leftarrow \neg G(x, y)$$
$$\mathit{neg\_tc}(x, y) \leftarrow \neg G(x, z), \mathit{neg\_tc}(z, y)$$

Stratified Semantics

Intuition: For evaluating the body of a rule instance r containing $\neg R(\vec{t})$, the value of the "negated" relation $R(\vec{t})$ should be known.

1. Evaluate first R
2. if $R(\vec{t})$ is false, then $\neg R(\vec{t})$ is true,
3. if $R(\vec{t})$ is true, then $\neg R(\vec{t})$ is false and the rule is not applicable.

Example

$$\mathit{boring}(\mathit{chess}) \leftarrow \neg \mathit{interesting}(\mathit{chess})$$
$$\mathit{interesting}(X) \leftarrow \mathit{difficult}(X)$$

For $\mathbf{I} = \{\}$, we compute the result $\{\mathit{boring}(\mathit{chess})\}$.

Note: this introduces *procedurality* (which violates declarativity)!

unibz

Dependency Graph for Datalog$^\neg$ Programs

Associate with each datalog$^\neg$ program P a directed graph *DEP*$(P) = (N, E)$, called *dependency graph*, as follows:

- $N =$ *sch*(P), i.e., the nodes are the relations
- $E = \{\langle R, R'\rangle \mid \exists r \in P : H(r) = R \land R' \in B(r)\}$,
  i.e., there are edges $R \to R'$ from the relations in rule heads
  to the relations in the body
- Mark each arc $R \to R'$ with "*",
  if $R(\vec{x})$ is in the head of a rule in P
  whose body contains $\neg R'(\vec{y})$.

Remark: *edb* relations are often omitted in the dependency graph

Example

P: $\quad \mathit{husband}(X) \leftarrow \mathit{man}(X),\ \mathit{married}(X).$
$\quad\quad \mathit{single}(X) \leftarrow \mathit{man}(X),\ \neg\mathit{husband}(X).$

DEP(P):

```
husband        married

   *

 single         man
```

Definition (Stratification Principle)

If $R = R_0 \rightarrow R_1 \rightarrow R_2 \rightarrow \cdots R_{n-1} \rightarrow R_n = R'$ such that
some $R_i \rightarrow R_{i+1}$ is marked with "*",
then R' must be evaluated prior to R.

Stratification

Definition

A *stratification* of a datalog program P is a partitioning

$$\Sigma = \bigcup_{i \geq 1}^{n} P_i$$

of *sch*(P) into nonempty, pairwise disjoint sets P_i such that

(a) if $R \in P_i$, $R' \in P_j$, and $R \rightarrow R'$ is in *DEP*(P), then $i \geq j$;

(b) if $R \in P_i$, $R' \in P_j$, and $R \rightarrow R'$ is in *DEP*(P) marked with "*," then $i > j$.

$P_1, \ldots, P_n$ are called the *strata* of P w.r.t. Σ

Definition

A datalog program P is called *stratified*, if it has some stratification Σ.

Evaluation Order

A stratification Σ gives an *evaluation order* for the relations in P, given $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$:

1. First evaluate the relations in P_1 (which is $\neg$-free).

   $\Rightarrow$ All relations R in heads of P_1 are defined. This yields $\mathbf{J}_1 \in \mathit{inst}(\mathit{sch}(P_1))$.

2. Evaluate P_2 considering relations in $\mathit{edb}(P)$ and P_1 as $\mathit{edb}(P_1)$, where $\neg R(\vec{t})$ is true if $R(\vec{t})$ is false in $\mathbf{I} \cup \mathbf{J}_1$;

   $\Rightarrow$ All relations R in heads of P_2 are defined. This yields $\mathbf{J}_2 \in \mathit{inst}(\mathit{sch}(P_2))$.

   ...

3. Evaluate P_i considering relations in $\mathit{edb}(P)$ and $P_1, \ldots, P_{i-1}$ as $\mathit{edb}(P_i)$, where $\neg R(\vec{t})$ is true if $R(\vec{t})$ is false in $\mathbf{I} \cup \mathbf{J}_1 \cup \cdots \cup \mathbf{J}_{i-1}$;

4. The result of evaluating P on $\mathbf{I}$ w.r.t. Σ, denoted $P_\Sigma(\mathbf{I})$, is given by $\mathbf{I} \cup \mathbf{J}_1 \cup \cdots \cup \mathbf{J}_n$.

unibz

Example

$$P = \{ \quad \mathit{husband}(X) \leftarrow \mathit{man}(X),\ \mathit{married}(X)$$
$$\mathit{single}(X) \leftarrow \mathit{man}(X),\ \neg\mathit{husband}(X)\ \}$$

Stratification Σ:
$P_1 = \{\mathit{man}, \mathit{married}\}$, $P_2 = \{\mathit{husband}\}$, $P_3 = \{\mathit{single}\}$

$\mathbf{I} = \{\mathit{man}(\mathit{dilbert})\}$:

1. Evaluate P_1: $\mathbf{J}_1 = \{\}$
2. Evaluate P_2: $\mathbf{J}_2 = \{\}$
3. Evaluate P_3: $\mathbf{J}_3 = \{\mathit{single}(\mathit{dilbert})\}$
4. Hence, $P_\Sigma(\mathbf{I}) = \{\mathit{man}(\mathit{dilbert})\}, \mathit{single}(\mathit{dilbert})\}$

Formal Definition of Stratified Semantics

Let P be a stratified Datalog$^{\neg}$ program with stratification $\Sigma = \bigcup_{i=1}^{n} P_i$.

- Let P_i^* be the set of rules from P whose relations in the head are in P_i, and set $\mathit{edb}(P_1^*) = \mathit{edb}(P)$, $\mathit{edb}(P_i^*) = \mathit{rels}(\bigcup_{j=1}^{i-1} P_j^*) \cup \mathit{edb}(P)$, $i > 1$.
- For every $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$, let $\mathbf{I}_0^\Sigma = \mathbf{I}$ and define

$$
\begin{array}{ccccccc}
\mathbf{I}_1^\Sigma & = & \mathbf{T}_{P_1^*}^{\omega}(\mathbf{I}_0^\Sigma) & = & \mathit{lfp}(\mathbf{T}_{P_1^*}(\mathbf{I}_0^\Sigma)) & \supseteq & \mathbf{I}_0^\Sigma \\
\mathbf{I}_2^\Sigma & = & \mathbf{T}_{P_2^*}^{\omega}(\mathbf{I}_1^\Sigma) & = & \mathit{lfp}(\mathbf{T}_{P_2^*}(\mathbf{I}_1^\Sigma)) & \supseteq & \mathbf{I}_1^\Sigma \\
 & \dots & & & & & \\
\mathbf{I}_i^\Sigma & = & \mathbf{T}_{P_i^*}^{\omega}(\mathbf{I}_{i-1}^\Sigma) & = & \mathit{lfp}(\mathbf{T}_{P_i^*}(\mathbf{I}_{i-1}^\Sigma)) & \supseteq & \mathbf{I}_{i-1}^\Sigma \\
 & \dots & & & & & \\
\mathbf{I}_n^\Sigma & = & \mathbf{T}_{P_n^*}^{\omega}(\mathbf{I}_{n-1}^\Sigma) & = & \mathit{lfp}(\mathbf{T}_{P_n^*}(\mathbf{I}_{n-1}^\Sigma)) & \supseteq & \mathbf{I}_{n-1}^\Sigma
\end{array}
$$

  where $\mathbf{T}_Q^\omega(\mathbf{J}) = \lim\{\mathbf{T}_Q^i(\mathbf{J})\}_{i\geq 0}$ with $\mathbf{T}_Q^0(\mathbf{J}) = \mathbf{J}$ and $\mathbf{T}_Q^{i+1} = \mathbf{T}_Q(\mathbf{T}_Q^i(\mathbf{J}))$, and $\mathit{lfp}(\mathbf{T}_Q(\mathbf{J}))$ is the least fixpoint $\mathbf{K}$ of $\mathbf{T}_Q$ such that $\mathbf{K}|\mathit{edb}(Q) = \mathbf{J}|\mathit{edb}(Q)$.

- Denote $P_\Sigma(\mathbf{I}) = \mathbf{I}_n^\Sigma$

Formal Definition of Stratified Semantics/2

Proposition

For every $i \in \{1, \ldots, n\}$,

- *lfp*$(\mathbf{T}_{P_i^*}(\mathbf{I}_{i-1}^{\Sigma}))$ exists,
- *lfp*$(\mathbf{T}_{P_i^*}(\mathbf{I}_{i-1}^{\Sigma})) = \mathbf{T}_{P_i^*}^{\omega}(\mathbf{I}_{i-1}^{\Sigma})$ holds,
- $\mathbf{I}_{i-1}^{\Sigma} \subseteq \mathbf{I}_i^{\Sigma}$.

Therefore, $P_\Sigma(\mathbf{I})$ is always well-defined.

Theorem

$P_\Sigma(\mathbf{I})$ is a minimal model $\mathbf{K}$ of P such that $\mathbf{K}|$*edb*$(P) = \mathbf{I}$.

unibz

Dilbert Example cont'd

$$P = \{ \quad \mathit{husband}(X) \leftarrow \mathit{man}(X),\ \mathit{married}(X)$$
$$\mathit{single}(X) \leftarrow \mathit{man}(X),\ \neg\mathit{husband}(X)\ \}$$
$\mathit{edb}(P) = \{\mathit{man}\}$

Stratification Σ: $P_1 = \{\mathit{man}, \mathit{married}\}$, $P_2 = \{\mathit{husband}\}$, $P_3 = \{\mathit{single}\}$

1. $P_1 = \{\}$
2. $P_2 = \{\mathit{husband}(X) \leftarrow \mathit{man}(X),\ \mathit{married}(X)\}$
3. $P_3 = \{\mathit{single}(X) \leftarrow \mathit{man}(X),\ \neg\mathit{husband}(X)\}$

$\mathbf{I} = \{\mathit{man}(\mathit{dilbert})\}$:

1. $\mathbf{I}_1^\Sigma = \{\mathit{man}(\mathit{dilbert})\}$
2. $\mathbf{I}_2^\Sigma = \{\mathit{man}(\mathit{dilbert})\}$
3. $\mathbf{I}_3^\Sigma = \{\mathit{man}(\mathit{dilbert}), \mathit{single}(\mathit{dilbert})\}$

Hence, $P_\Sigma(\mathbf{I}) = \{\mathit{man}(\mathit{dilbert}), \mathit{single}(\mathit{dilbert})\}$

unibz

Stratification Theorem

The stratification Σ above is not unique

- Alternative stratification Σ':
  $P_1 = \{\textit{man}, \textit{married}, \textit{husband}\}$, $P_2 = \{\textit{single}\}$
- Evaluation with respect to Σ' yields same result!

The choice of a particular stratification is irrelevant:

Theorem (Stratification Theorem)

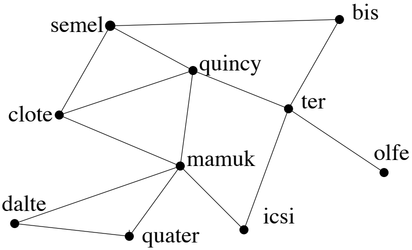
Let P be a stratifiable datalog$^\neg$ program. Then, for any stratifications Σ and Σ' and $\mathbf{I} \in \textit{inst}(\textit{sch}(P))$, $P_\Sigma(\mathbf{I}) = P_{\Sigma'}(\mathbf{I})$.

- Thus, syntactic stratification yields semantically a canonical way of evaluation.
- The result $P_{\text{str}}(\mathbf{I})$ is called the *perfect model* or *stratified model* of P for $\mathbf{I}$.

Remark: Prolog features SLDNF – SLD resolution with (finite) negation as failure

Example: Railroad Network

Determine whether safe connections between locations in a railroad network



- **Cutpoint c for a and b:** if c fails, there is no connection between a and b
- **Safe connection between a and b:** no cutpoints between a and b exist
- E.g., ter is a cutpoint for olfe and semel, while quincy is not

Example: Railroad Network/2

Relations:

$$
\begin{aligned}
\mathit{link}(X, Y)&: \text{ direct connection from station } X \text{ to } Y \text{ (edb facts)}\\
\mathit{linked}(A, B)&: \text{ symmetric closure of } \mathit{link}.\\
\mathit{connected}(A, B)&: \text{ there is path between } A \text{ and } B \text{ (one or more links)}\\
\mathit{cutpoint}(X, A, B)&: \text{ each path from } A \text{ to } B \text{ goes through station } X\\
\mathit{circumvent}(X, A, B)&: \text{ there is a path between } A \text{ and } B \text{ not passing } X\\
\mathit{has\_icut\_point}(A, B)&: \text{ there is at least one cutpoint between } A \text{ and } B.\\
\mathit{safely\_connected}(A, B)&: \ A \text{ and } B \text{ are connected with no cutpoint.}\\
\mathit{station}(X)&: \ X \text{ is a railway station.}
\end{aligned}
$$

Example: Railroad Network/3

Railroad program P:

r_1 : $linked(A, B) \leftarrow link(A, B).$

r_2: $linked(A, B) \leftarrow link(B, A).$

r_3: $connected(A, B) \leftarrow linked(A, B).$

r_4: $connected(A, B) \leftarrow connected(A, C), linked(C, B).$

r_5: $cutpoint(X, A, B) \leftarrow connected(A, B), station(X), \neg circumvent(X, A, B).$

r_6: $circumvent(X, A, B) \leftarrow linked(A, B), X \neq A, station(X), X \neq B.$

r_7: $circumvent(X, A, B) \leftarrow circumvent(X, A, C), circumvent(X, C, B).$

r_8: $has\_icut\_point(A, B) \leftarrow cutpoint(X, A, B), X \neq A, X \neq B.$

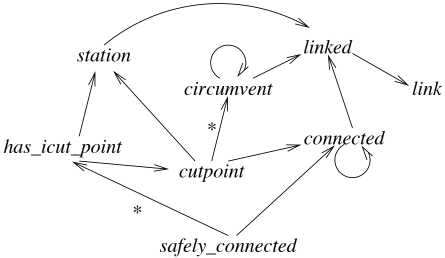
r_9: $safely\_connected(A, B) \leftarrow connected(A, B), \neg has\_icut\_point(A, B).$

r_{10}: $station(X) \leftarrow linked(X, Y).$

Remark: Inequality ($\neq$) is used here as built-in. It can be easily defined in stratified manner.

Example: Railroad Network/3

DEP(P):



Stratification Σ:

$P_1 = \{link, linked, station, circumvent, connected\}$
$P_2 = \{cutpoint, has\_icut\_point\}$
$P_3 = \{safely\_connected\}$

Example: Railroad Network/4

$$\mathbf{I}(link) = \{\ \langle semel, bis\rangle, \langle bis, ter\rangle, \langle ter, olfe\rangle, \langle ter, icsi\rangle, \langle ter, quincy\rangle, \langle quincy, semel\rangle, \langle quincy, clote\rangle, \langle quincy, mamuk\rangle, \ldots, \langle dalte, quater\rangle\ \}$$

Evaluation $P_\Sigma(\mathbf{I})$:

1. $P_1 = \{link, linked, station, circumvent, connected\}$:

   $\mathbf{J}_1 = \{linked(semel, bis),\ linked(bis, ter),\ linked(ter, olfe), \ldots, connected(semel, olfe),\ \ldots,\ circumvent(quincy, semel, bis), \ldots\}$

2. $P_2 = \{cutpoint, has\_icut\_point\}$:

   $\mathbf{J}_2 = \{cutpoint(ter, semel, olfe),\ has\_icut\_point(semel, olfe) \ldots\}$

3. $P_3 = \{safely\_connected\}$:

   $\mathbf{J}_3 = \{safely\_connected(semel, bis),\ safely\_connected(semel, ter)\}$
   But, $safely\_connected(semel, olfe) \notin \mathbf{J}_3$

unibz

Algorithm STRATIFY

Input: A datalog$^\neg$ program P
Output: A stratification Σ for P, or "no" if none exists

1. Construct the directed graph $G := \mathit{DEP}(P)$ $(=\langle N, E\rangle)$ with markers "*";
2. **For each** pair $(R, R') \in N \times N$ **do**
   if R reaches R' via some path containing a marked arc
   then $E := E \cup \{R \to R'\}$; mark $R \to R'$ with "*" ;
3. $i := 1$;
4. Identify the set K of all vertices R in G s.t. no marked $R \to R'$ is in E
5. **If** $K = \emptyset$ and G has vertices left,
   then output "no"
   else output K as stratum P_i;
   remove all vertices in K and corresponding arcs from G;
6. **If** G has vertices left
   then $i := i + 1$; **goto** step 4;
   else stop.

Runs in polynomial time!

unibz

Stable Models Semantics

- **Idea**: Try to construct a (minimal) fixpoint by iteration from input.
  If the construction succeeds, the result is the semantics.
- **Problem**: Application of rules might be compromised.

Example

$$P = \{p(a) \leftarrow \neg p(a), \qquad q(b) \leftarrow p(a), \qquad p(a) \leftarrow q(b)\}$$

(*edb*(P) is void, thus $\mathbf{I}$ is immaterial and omitted)

- $\mathbf{T}_P$ has the least fixpoint $\{p(a), q(b)\}$
- It is iteratively constructed $\mathbf{T}_P^\omega = \{p(a), q(b)\}$
- $p(a)$ is included into $\mathbf{T}_P^1$ by the first rule, since $p(a) \notin \mathbf{T}_P^0 = \emptyset$.
- This compromises the rule application, and $p(a)$ is not “foundedly” derived!

Fixed Evaluation of Negation

Observation: $\mathbf{T}_P$ is not monotonic.

Solution: Keep negation throughout fixpoint-iteration fixed.

- Evaluate negation w.r.t. a fixed candidate fixpoint model $\mathbf{J}$
- Introduce for datalog$^\neg$ program and $\mathbf{J} \in \mathit{inst}(\mathit{sch}(P))$ a new immediate consequence operator $\mathbf{T}_{P,\mathbf{J}}$:

unibz

Immediate Consequences under Fixed Negation

Definition

Given a datalog$^\neg$ program P and $\mathbf{J}, \mathbf{K} \in$ *inst*(*sch*(P)),
a fact $R(\vec{t})$ is an *immediate* consequence for $\mathbf{K}$ and P under negation $\mathbf{J}$,
if either

- $R \in$ *edb*(P) and $R(\vec{t}) \in \mathbf{K}$, or
- there exists some ground instance r of a rule in P such that
  - $H(r) = R(\vec{t})$,
  - $B^+(r) \subseteq \mathbf{K}$, and
  - $B^-(r) \cap \mathbf{J} = \emptyset$

(that is, evaluate "$\neg$" under $\mathbf{J}$ instead of $\mathbf{K}$).

Immediate Consequences under Fixed Negation/2

Definition

For any datalog$^\neg$ program P and $\mathbf{J}, \mathbf{K} \in \mathit{inst}(\mathit{sch}(P))$, let

$$\mathbf{T}_{P,\mathbf{J}}(\mathbf{K}) = \{A \mid A \text{ is an immediate consequence for } \mathbf{K} \text{ and } P \text{ under negation } \mathbf{J}\}$$

Notice:

- $\mathbf{T}_P(\mathbf{K})$ coincides with $\mathbf{T}_{P,\mathbf{K}}(\mathbf{K})$
- $\mathbf{T}_{P,\mathbf{J}}$ is a monotonic operator, hence has for each $\mathbf{K} \in \mathit{inst}(\mathit{sch}(P))$ a least fixpoint containing $\mathbf{K}$, denoted $\mathit{lfp}(\mathbf{T}_{P,\mathbf{J}}(\mathbf{K}))$
- $\mathit{lfp}(\mathbf{T}_{P,\mathbf{J}}(\mathbf{I}))$ coincides with $\mathbf{I}$ on $\mathit{edb}(P)$ and is the limit $\mathbf{T}^\omega_{P,\mathbf{J}}(\mathbf{I})$ of the sequence

$$\{\mathbf{T}^i_{P,\mathbf{J}}(\mathbf{I})\}_{i \geq 0},$$

where $\mathbf{T}^0_{P,\mathbf{J}}(\mathbf{I}) = \mathbf{I}$ and $\mathbf{T}^{i+1}_{P,\mathbf{J}}(\mathbf{I}) = \mathbf{T}_{P,\mathbf{J}}(\mathbf{T}^i_{P,\mathbf{J}}(\mathbf{I}))$.

unibz

Stable Models

Using $\mathbf{T}_{P,\mathbf{J}}$, stable models are defined by requiring that $\mathbf{J}$ is reproduced by the program:

Definition

Let P be a datalog$^\neg$ program P and $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$.
Then, a stable model for P and $\mathbf{I}$ is any $\mathbf{J} \in \mathit{inst}(\mathit{sch}(P))$ such that

1. $\mathbf{J}|\mathit{edb}(P) = \mathbf{I}$, and
2. $\mathbf{J} = \mathit{lfp}(\mathbf{T}_{P,\mathbf{J}}(\mathbf{I}))$.

Notice: Monotonicity of $\mathbf{T}_{P,\mathbf{J}}$ ensures that at no point in the construction of $\mathit{lfp}(\mathbf{T}_{P,\mathbf{J}})(\mathbf{I})$ using fixpoint iteration from $\mathbf{I}$, the application of a rule can be compromised later.

Example

Let

$$P = \{ \ \ p(a) \leftarrow \neg p(a), \qquad q(b) \leftarrow p(a), \qquad p(a) \leftarrow q(b) \ \ \}$$

(*edb*(P) is void, thus $\mathbf{I}$ is immaterial and omitted)

- Take $\mathbf{J} = \{p(a), q(b)\}$. Then
  - $\mathbf{T}^0_{P,\mathbf{J}} = \emptyset$
  - $\mathbf{T}^1_{P,\mathbf{J}} = \emptyset$
- Thus *lfp*$(\mathbf{T}_{P,\mathbf{J}}) = \emptyset \neq \mathbf{J}$.
- Hence, the fixpoint $\mathbf{J}$ of $\mathbf{T}_P$ is refuted.
- For P, no stable model exists; thus, it may be regarded as "inconsistent".

unibz

Nondeterminism

- **Problem**: A datalog program may have multiple stable models:

$$P = \{ \quad \begin{array}{r} \mathit{single}(X) \leftarrow \mathit{man}(X), \neg \mathit{husband}(X) \\ \mathit{husband}(X) \leftarrow \mathit{man}(X), \neg \mathit{single}(X) \end{array} \quad \}$$

$\mathbf{I} = \{\mathit{man}(\mathit{dilbert})\}$

- $\mathbf{J}_1 = \{\mathit{man}(\mathit{dilbert}),\ \mathit{single}(\mathit{dilbert})\}$ is a stable model:
  - $\mathbf{T}^0_{P,\mathbf{J}_1}(\mathbf{I}) = \{\mathit{man}(\mathit{dilbert})\}$
  - $\mathbf{T}^1_{P,\mathbf{J}_1}(\mathbf{I}) = \{\mathit{man}(\mathit{dilbert}), \mathit{single}(\mathit{dilbert})\}$ (apply 2nd rule)
  - $\mathbf{T}^2_{P,\mathbf{J}_1}(\mathbf{I}) = \{\mathit{man}(\mathit{dilbert}), \mathit{single}(\mathit{dilbert})\} = \mathbf{T}^\omega_{P,\mathbf{J}_1}(\mathbf{I})$

- Similarly, $\mathbf{J}_2 = \{\mathit{man}(\mathit{dilbert}),\ \mathit{husband}(\mathit{dilbert})\}$ is a stable model (symmetry)

Stable Model Semantics – Definition

Solution: Define stable model semantics of P as the intersection of all stable models (*certain semantics*):

Denote for a datalog$^\neg$ program P and $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$ by $\mathit{SM}(P, \mathbf{I})$ the set of all stable models for $\mathbf{I}$ and P.

Definition

The stable models semantics of a datalog$^\neg$ program P for $\mathbf{I} \in \mathit{inst}(\mathit{edb}(P))$, denoted $P_{\mathsf{sm}}(\mathbf{I})$, is given by

$$P_{sm}(\mathbf{I}) = \begin{cases} \bigcap \mathit{SM}(P, \mathbf{I}), & \text{if } \mathit{SM}(P, \mathbf{I}) \neq \emptyset, \\ \mathbf{B}(P, \mathbf{I}), & \text{otherwise.} \end{cases}$$

Examples

Example

$$P = \{ \quad \mathit{single}(X) \leftarrow \mathit{man}(X), \neg \mathit{husband}(X)$$
$$\mathit{husband}(X) \leftarrow \mathit{man}(X), \neg \mathit{single}(X) \quad \}$$

$P_{\mathsf{sm}}(\{\mathit{man}(\mathit{dilbert})\}) = \{\mathit{man}(\mathit{dilbert})\}$

Example

$$P = \{p(a) \leftarrow \neg p(a), \qquad q(b) \leftarrow p(a), \qquad p(a) \leftarrow q(b)\}$$

$P_{\mathsf{sm}}(\emptyset) = \{p(a), p(b), q(a), q(b)\} = \mathbf{B}(P, \mathbf{I})$.

unibz

Some Properties

Proposition

Each $\mathbf{K} \in \mathit{SM}(P, \mathbf{I})$ is a minimal model $\mathbf{K}$ of P such that $\mathbf{K}|\mathit{edb}(P) = \mathbf{I}$.

Proposition

Each $\mathbf{K} \in \mathit{SM}(P, \mathbf{I})$ is a minimal fixpoint $\mathbf{K}$ of $\mathbf{T}_P$ such that $\mathbf{K}|\mathit{edb}(P) = \mathbf{I}$.

Theorem

If P is a stratified program, than for every $\mathbf{I} \in \mathit{edb}(P)$, $P_{\mathsf{sm}}(\mathbf{I}) = P_{\mathsf{strat}}(\mathbf{I})$.
Thus, stable model semantics extends stratified semantics to a larger class of programs

Evaluation of stable model semantics is intractable: Deciding whether $R(\vec{c}) \in P_{sm}(\mathbf{I})$ for given $R(\vec{c})$ and $\mathbf{I}$ (while P is fixed) is coNP-complete.