Logic Programming

Formal Semantics

Michael Genesereth
Computer Science Department
Stanford University
Assume we have a dataset $D$ and a compatible ruleset $O$.

1. Facts in $D$ are $true$.

2. Negation is $true$ if and only if the negated expression is $not\ true$.

3. Conjunction is $true$ if and only if all conjuncts are $true$.

4. If an instance of the body of any rule is $true$, then the corresponding instance of the head is $true$.

*almost* right
The **Herbrand universe** for a basic logic program is the set of all *ground terms* that can be formed from the symbols and predicates in the vocabulary.

Symbols: a, b
Predicate: r/2
Herbrand Universe: \{a, b\}

The **Herbrand base** for a basic logic program is the set of all *ground atoms* that can be formed from the vocabulary.

Herbrand Base: \{r(a,a), r(a,b), r(b,a), r(b,b)\}
An **interpretation** of a basic logic program is an arbitrary *subset* of the program’s Herbrand base.

Herbrand Base: \{ r(a,a), r(a,b), r(b,a), r(b,b) \}
Interpretation: \{ r(a,b), r(b,a) \}
Interpretation: \{ r(a,a), r(a,b), r(b,a) \}
Interpretation: \{ r(a,a), r(a,b), r(b,a), r(b,b) \}

...  
Interpretation: \{\}

A **model** of a basic logic program is an interpretation that *satisfies* the program (definition to follow).
An **instance of a rule** is a rule in which all variables have been consistently replaced by ground terms.

**Rule**

\[ r(X, Y) :- p(X, Y) \land \neg q(Y) \]

**Herbrand Universe**

\{a, b\}

**Instances**

\[
\begin{align*}
    r(a, a) & :- p(a, a) \land \neg q(a) \\
    r(a, b) & :- p(a, b) \land \neg q(b) \\
    r(b, a) & :- p(b, a) \land \neg q(a) \\
    r(b, b) & :- p(b, b) \land \neg q(b)
\end{align*}
\]
An interpretation $D$ satisfies a rule instance $p : - p_1 \land \ldots \land p_n$ if and only if $p \in D$ whenever $D$ satisfies $p_1, \ldots, p_n$. $D$ satisfies a ground atom $p$ if and only if $p \in D$. $D$ satisfies a ground negation $\neg p$ if and only if $p \notin D$.

An interpretation $D$ satisfies a basic logic program if and only if (1) all of the elements of the constituent dataset are included in $D$ and (2) $D$ satisfies all instances of all of the rules in the constituent ruleset.
Example

Dataset
\[ p(a,b) \]
\[ p(b,c) \]
\[ p(c,d) \]
\[ p(d,c) \]

Ruleset
\[ r(X,Y) :- p(X,Y) \land \neg p(Y,X) \]

Model
\[ p(a,b) \]
\[ p(b,c) \]
\[ p(c,b) \}
\[ p(d,c) \]
\[ r(a,b) \}
\[ r(b,c) \}

p extension
r extension
Non-Examples

Dataset
- \( p(a,b) \)
- \( p(b,c) \)
- \( p(c,d) \)
- \( p(d,c) \)

Ruleset
- \( r(X,Y) : \sim p(X,Y) \)

Model
- \( p(a,b) \)
- \( p(b,c) \)
- \( p(c,d) \)
- \( p(d,c) \)
- \( r(a,b) \)

Not Models
- \( p(a,b) \)
- \( p(b,c) \)
- \( p(c,d) \)
- \( p(d,c) \)
- \( r(a,b) \)
- \( r(b,c) \)
- \( r(c,d) \)
Safety
Safety

A rule is *safe* if and only if every variable in the head appears in some positive subgoal in the body and every variable in a negative subgoal appears in a prior positive subgoal.

**Safe Rule:**
\[ r(X, Z) :- p(X, Y) \land q(Y, Z) \land \neg r(X, Y) \]

**Unsafe Rule:**
\[ r(X, Z) :- p(X, Y) \land q(Y, X) \]

**Unsafe Rule:**
\[ r(X, Y) :- p(X, Y) \land \neg q(Y, Z) \]
Unbound Variables in Head

Ruleset

\[ s(X,Y,Z) :- p(X,Y) \]

Dataset

\[ p(a,b) \]

Output for \( s \)

\[ s(a,b,a) \]
\[ s(a,b,b) \]
\[ s(a,b,c) \]
\[ s(a,b,d) \]
\[ s(a,b,e) \]
\[ \ldots \]
Unbound Variables in Negation

Ruleset

\[ t(X,Y) :- p(X,Y) \land \neg q(Y,Z) \]

Dataset

\[
\begin{align*}
p(a,b) \\
p(a,c) \\
p(a,c) \\
q(c,d)
\end{align*}
\]

Output
Unbound Variables in Negation

Ruleset

\[ t(X,Y) :- p(X,Y) \land \sim q(Y,Z) \]

Dataset

\begin{align*}
p(a,b) \\
p(a,c) \\
q(c,d)
\end{align*}

Output

\begin{align*}
t(a,b) \\
t(a,c)
\end{align*}
Unbound Variables in Negation

Ruleset

\[ t(X, Y) :- p(X, Y) \& \neg q(Y, Z) \]

Dataset

\[
\begin{align*}
p(a, b) \\
p(a, c) \\
q(c, d)
\end{align*}
\]

Output

\[
\begin{align*}
t(a, b) \\
t(a, c)
\end{align*}
\]

Alternative Rules

\[
\begin{align*}
t(X, Y) :- p(X, Y) \& \neg \neg q(Y) \\
\neg q(Y) :- q(Y, Z)
\end{align*}
\]
Stratified Negation
Multiple Models

Dataset

\[
\begin{align*}
& p(a,b) \\
& p(b,c) \\
& p(c,d) \\
& p(d,c)
\end{align*}
\]

Ruleset

\[
\begin{align*}
& r(X,Y) :- p(X,Y) & \& \neg p(Y,X)
\end{align*}
\]

Models

\[
\begin{align*}
& p(a,b) & p(a,b) & p(a,b) \\
& p(b,c) & p(b,c) & p(b,c) \\
& p(c,d) & p(c,d) & p(c,d) \\
& p(d,c) & p(d,c) & p(d,c) \\
& r(a,b) & r(a,b) & r(a,b) \\
& r(b,c) & r(b,c) & r(b,c) \\
& r(c,d) & r(c,d) & r(c,d) \\
& r(d,c) & r(d,c) & r(d,c)
\end{align*}
\]
So What?

We want our definitions to be *if and only if*. We want to include among our conclusions only those facts that *must* be true based on the definition.

All factoids in dataset *must* be true.
All factoids required by rules *must* be true.
*Other factoids should be excluded.*

What we want is logical entailment. A factoid is **logically entailed** by a basic logic program if and only if it is true in *every* model of the program, i.e. the set of conclusions is the intersection of all models of the program.
A model $D$ of a logic program $P$ is minimal if and only if no proper subset of $D$ is a model of $P$.

**Ruleset**

$$r(X,Y) :\neg p(X,Y) \& \neg p(Y,X)$$

**Models**

- $p(a,b)$
- $p(b,c)$
- $p(c,d)$
- $p(d,c)$
- $r(a,b)$
- $r(b,c)$
- $r(c,d)$
- $r(d,c)$
Minimal Models

If a program has just one minimal model, then every factoid true in that model is trivially true in every model of the program.

A logic program that does not contain any negations has a unique minimal model.

A logic program with negations can have more than one minimal model (in addition to multiple non-minimal models).

If a program is stratified (as defined below), then once again there is only one minimal model.
Multiple *Minimal* Models

**Dataset**

\[ p(a, b) \]
\[ p(b, a) \]

**Ruleset**

\[ r(X) :- p(X, Y) \& \neg r(Y) \]

**Interpretations**

<table>
<thead>
<tr>
<th></th>
<th>( p(a, b) )</th>
<th>( p(a, b) )</th>
<th>( p(a, b) )</th>
<th>( p(a, b) )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( p(b, a) )</td>
<td>( p(b, a) )</td>
<td>( p(b, a) )</td>
<td>( p(b, a) )</td>
</tr>
<tr>
<td></td>
<td>( r(a) )</td>
<td>( r(b) )</td>
<td>( r(a) )</td>
<td>( r(b) )</td>
</tr>
</tbody>
</table>

*Is \( r(a) \) true or not? What about \( r(b) \)?

*The intersection of all models is not necessarily a model!*
The *dependency graph* for a set of rules is a directed graph in which (1) the nodes are the relations mentioned in the head and bodies of the rules and (2) there is an arc from a node $p$ to a node $q$ whenever $p$ occurs with the body of a rule in which $q$ is in the head.

$$
\begin{align*}
    r(X,Y) &: r(X,Y) :- p(X,Y) \& q(X,Y) \\
    s(X,Y) &: s(X,Y) :- r(X,Y) \\
    s(X,Z) &: s(X,Z) :- r(X,Y) \& t(Y,Z) \\
    t(X,Z) &: t(X,Z) :- s(X,Y) \& s(Y,X)
\end{align*}
$$

A set of rules is *recursive* if it contains a cycle. Otherwise, it is *non-recursive*. 
A set of rules is said to be *stratified* if and only if there is no recursive cycle in the dependency graph involving a negation.

**Stratified Negation:**

\[
\begin{align*}
    r(X,Z) & : \neg p(X,Y) \\
    r(X,Z) & : \neg r(X,Y) \land r(Y,Z)
\end{align*}
\]

**Negation that is not stratified:**

\[
\begin{align*}
    r(X,Z) & : \neg p(X,Y) \\
    r(X,Z) & : \neg p(X,Y) \land \neg r(Y,Z)
\end{align*}
\]

*All negations must be stratified.*
We say that a set of view definitions is \textit{stratified} if and only if its rules can be partitioned into \textit{strata} in such a way that

(1) every stratum contains at least one rule

(2) the rules defining relations that appear in positive goals of a rule appear in the \textit{same stratum} as that rule \textit{or} in \textit{some lower stratum}

(3) the rules defining relations that appear in negative subgoals of a rule occur in \textit{some lower stratum} (not the same stratum)
Example

\[
\begin{align*}
  r(X,Y) & :- \ q(X,Y) \\
  r(X,Z) & :- \ q(X,Y) \ & r(Y,Z) \\
  s(X,Y) & :- \ p(X) \ & p(Y) \ & \neg r(X,Y)
\end{align*}
\]

<table>
<thead>
<tr>
<th>Stratum</th>
<th>Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(s(X,Y) :- p(X) \ &amp; p(Y) \ &amp; \neg r(X,Y))</td>
</tr>
</tbody>
</table>
| 1       | \(r(X,Y) :- q(X,Y)\)  \\
\   & \(r(X,Z) :- q(X,Y) \ & r(Y,Z)\) |

Non-example

\[
\begin{align*}
  r(X,Y) & :- \ p(X) \ & p(Y) \ & q(X,Y) \\
  s(X,Y) & :- \ r(X,Y) \ & \neg s(Y,X)
\end{align*}
\]
Good:
\[
\begin{align*}
r(X,Z) & : - p(X,Y) \\
r(X,Z) & : - r(X,Y) & & \& & r(Y,Z)
\end{align*}
\]

Bad:
\[
\begin{align*}
r(X,Z) & : - p(X,Y) \\
r(X,Z) & : - p(X,Y) & & \& & \sim r(Y,Z)
\end{align*}
\]
If a program has just one minimal model, then every factoid true in that model is trivially true in every model of the program.

A logic program that does not contain any negations has a unique minimal model.

A logic program with negations can have more than one minimal model (in addition to multiple non-minimal models).

If a program is stratified (as just defined), then there is only one minimal model.