Logic Programming

Datalog

Michael Genesereth
Computer Science Department
Stanford University
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
grandparent(art, cal)
grandparent(art, cam)
grandparent(art, cat)
grandparent(art, coe)
grandparent(X, Z) :- parent(X, Y) & parent(Y, Z)
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)


\[
\text{grandparent}(X,Z) \leftarrow \text{parent}(X,Y) \& \text{parent}(Y,Z)
\]

= 

grandparent(art,cal)
grandparent(art,cam)
grandparent(art,cat)
grandparent(art,coe)
Basic Logic Programming

Represent information as combination of facts and rules. Encode information about base relations as facts in a dataset and write rules to define view relations in terms of the base relations.

Benefits:
- Economy - fewer facts need to be stored
- Less chance of getting out of sync
- Rules work for any number of objects
Syntax
A **constant** is a string of lower case letters, digits, underscores, and periods *or* strings of arbitrary ascii characters within double quotes.

```
art  cs151  94301
    barack_obama  person.age
  “The quick brown fox & the lazy dog.”
```

A **variable** is a lone underscore or a string of letters, digits, underscores beginning with an upper case letter

```
_       x       x23       X_23       Somebody
```
Symbols
  art
  bob

Variables
  X
  Y23
Atoms

\[ p(a, b) \]
\[ p(a, x) \]
\[ p(Y, c) \]

Negations

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

Literals (atoms or negations of atoms)

\[ p(a, Y) \]
\[ \neg p(a, Y) \]

An atom is a *positive literal*.
A negation is a *negative literal*. 
Informal semantics: If an instance of the body is “true”, then the corresponding instance of the head is “true”. Formal semantics in next lecture.
r(X,Y) :- p(X,Y) & ~q(Y)

p(X,Y) :- f(X,Y)

p(X,Y) :- m(X,Y)
Basic Logic Program

Dataset
- p(a,b)
- p(a,c)
- q(b,c)
- r(a)
- r(c)

Ruleset
- s(X,Y) :- p(X,Y)
- s(X,Y) :- q(X,Y)
- t(X,Y) :- s(X,Y) & ~r(Y)
A ruleset is **compatible** with a dataset if and only if

1. all constants shared between the dataset and the ruleset are of the same type (symbol, constructor, predicate)
2. all shared constructors and predicates have the same arity
3. none of the predicates in the dataset appear in the heads of any rules in the logic program. Okay in bodies.

The **vocabulary** of a basic logic program is the union of the vocabularies of the dataset and the logic program.
Informal Semantics

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*.

Negations complicate matters.

Full formal semantics next time.
Kinship
parent(art, bob)
parent(art, bea)
parent(bob, cal)
parent(bob, cam)
parent(bud, coe)
parent(bud, cory)
Example

Personhood:

Data:
- male(art)
- male(bob)
- male(cal)
- male(cam)
- female(bea)
- female(cat)
- female(coe)

View:
- person(art)
- person(bob)
- person(cal)
- person(cam)
- person(bea)
- person(cat)
- person(coe)
Personhood:

\[
\text{person}(X) :- \text{male}(X) \\
\text{person}(X) :- \text{female}(X)
\]

Data:  

- male(art)  
- male(bob)  
- male(cal)  
- male(cam)  
- female(bea)  
- female(cat)  
- female(coe)

View:  

- person(art)  
- person(bob)  
- person(cal)  
- person(cam)  
- person(bea)  
- person(cat)  
- person(coe)
Ancestors:

Data:

parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,coe)
parent(bea,coe)
parent(bea,cory)

View:

ancestor(art,bob)
ancestor(art,bea)
ancestor(bob,cal)
ancestor(bob,coe)
ancestor(bea,coe)
ancestor(bea,cory)
ancestor(art,cal)
ancestor(art,coe)
ancestor(art,cory)
Ancestors:

\[
\text{ancestor}(X, Z) :- \text{parent}(X, Z) \\
\text{ancestor}(X, Z) :- \text{parent}(X, Y) \land \text{ancestor}(Y, Z)
\]

Data:                                     View:

\[
\text{parent}(\text{art}, \text{bob}) \\
\text{parent}(\text{art}, \text{bea}) \\
\text{parent}(\text{bob}, \text{cal}) \\
\text{parent}(\text{bob}, \text{coe}) \\
\text{parent}(\text{bea}, \text{coe}) \\
\text{parent}(\text{bea}, \text{cory})
\]

\[
\text{ancestor}(\text{art}, \text{bob}) \\
\text{ancestor}(\text{art}, \text{bea}) \\
\text{ancestor}(\text{bob}, \text{cal}) \\
\text{ancestor}(\text{bob}, \text{cam}) \\
\text{ancestor}(\text{bea}, \text{coe}) \\
\text{ancestor}(\text{bea}, \text{cory}) \\
\text{ancestor}(\text{art}, \text{cal}) \\
\text{ancestor}(\text{art}, \text{cam}) \\
\text{ancestor}(\text{art}, \text{coe}) \\
\text{ancestor}(\text{art}, \text{cory})
\]
Childlessness:

<table>
<thead>
<tr>
<th>Data</th>
<th>View</th>
</tr>
</thead>
<tbody>
<tr>
<td>parent(art,bob)</td>
<td>childless(cal)</td>
</tr>
<tr>
<td>parent(art,bea)</td>
<td>childless(cam)</td>
</tr>
<tr>
<td>parent(bob,cal)</td>
<td>childless(coe)</td>
</tr>
<tr>
<td>parent(bob,cam)</td>
<td>childless(cory)</td>
</tr>
<tr>
<td>parent(bea,coe)</td>
<td></td>
</tr>
<tr>
<td>parent(bea,cory)</td>
<td></td>
</tr>
</tbody>
</table>
Childlessness:

\[
\text{childless}(X) : \text{ person}(X) \& \, \neg \text{isparent}(X)
\]

\[
\text{isparent}(X) : \text{ parent}(X,Y)
\]

Data:  
parent(art,bob)  
parent(art,bea)  
parent(bob,cal)  
parent(bob,cam)  
parent(bea,coe)  
parent(bea,cory)  

View:  
\text{childless}(cal)  
\text{childless}(cam)  
\text{childless}(coe)  
\text{childless}(cory)
Blocks World
Symbols: a, b, c, d, e

Unary Predicates:
- clear - blocks with no blocks on top
- cluttered - blocks with something on top
- supported - blocks resting on other blocks
- table - blocks on the table

Binary Predicates:
- on - pairs of blocks in which first is on the second
- above - pairs in which first block is above the second

Ternary Predicates:
- stack - triples of blocks arranged in a stack
block(a)
block(b)   on(a,b)
block(c)   on(b,c)
block(d)   on(d,e)
block(e)

clear(a)   table(c)   above(a,b)
clear(d)   table(d)   above(b,c)
cluttered(b)   supported(a)   above(d,e)
cluttered(c)   supported(b)
cluttered(e)   supported(d)   stack(a,b,c)
Blocks World

block(a)
bloc(k(b)) on(a,b)
block(c) on(b,c)
block(d) on(d,e)
block(e)

cluttered(Y) :- on(X,Y)
clear(X) :- block(X) & ~cluttered(X)

cluttered(b) clear(a)
cluttered(c) clear(d)
cluttered(e)
Blocks World

\[\begin{align*}
\text{block}(a) &\quad \text{on}(a, b) \\
\text{block}(b) &\quad \text{on}(b, c) \\
\text{block}(c) &\quad \text{on}(d, e) \\
\text{block}(d) &\quad \text{on}(d, e) \\
\text{block}(e) &
\end{align*}\]

\[\begin{align*}
\text{supported}(X) &\quad \text{on}(X, Y) \\
\text{table}(X) &\quad \text{block}(X) \land \neg \text{supported}(X)
\end{align*}\]

\[\begin{align*}
\text{supported}(a) &\quad \text{table}(c) \\
\text{supported}(b) &\quad \text{table}(e) \\
\text{supported}(d) &
\end{align*}\]
```
blocks World

block(a)
block(b)    on(a,b)
block(c)    on(b,c)
block(d)    on(d,e)
block(e)

stack(X, Y, Z) :- on(X, Y) & on(Y, Z)
```
Blocks World

\[
\begin{align*}
\text{block}(a) & \quad \text{on}(a, b) \\
\text{block}(b) & \quad \text{on}(b, c) \\
\text{block}(c) & \quad \text{on}(d, e) \\
\text{block}(d) & \quad \text{block}(e) \\
\text{above}(X,Z) & \quad \text{on}(X,Z) \\
\text{above}(X,Z) & \quad \text{on}(X,Y) \land \text{above}(Y,Z) \\
\text{above}(a,b) & \\
\text{above}(b,c) & \\
\text{above}(d,e) & \\
\text{above}(a,c) & 
\end{align*}
\]
Modular Arithmetic
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0+0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0+1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>0+2</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0+3</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1+0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1+1</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1+2</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1+3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2+0</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2+1</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2+2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2+3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3+0</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3+1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3+2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>3+3</td>
<td>2</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td></td>
</tr>
<tr>
<td>plus(0,0,0)</td>
<td>plus(1,0,1)</td>
<td>plus(2,0,2)</td>
<td>plus(3,0,3)</td>
<td></td>
</tr>
<tr>
<td>plus(0,1,1)</td>
<td>plus(1,1,2)</td>
<td>plus(2,1,3)</td>
<td>plus(3,1,0)</td>
<td></td>
</tr>
<tr>
<td>plus(0,2,2)</td>
<td>plus(1,2,3)</td>
<td>plus(2,2,0)</td>
<td>plus(3,2,1)</td>
<td></td>
</tr>
<tr>
<td>plus(0,3,3)</td>
<td>plus(1,3,0)</td>
<td>plus(2,3,1)</td>
<td>plus(3,3,2)</td>
<td></td>
</tr>
</tbody>
</table>
Approach 2 (1)

number(0)
number(1)
number(2)
number(3)
Approach 2 (2)

number(0)
number(1)
number(2)
number(3)

next(0,1)
next(1,2)
next(2,3)
next(3,0)
number(0)
number(1)
number(2)
number(3)

next(0, 1)
next(1, 2)
next(2, 3)
next(3, 0)

plus(0, Y, Y) :- number(Y)
plus(1, Y, Z) :- next(Y, Z)
plus(2, Y, Z) :- next(Y, Z1) & next(Z1, Z)
plus(3, Y, Z) :- next(Y, Z1) & next(Z1, Z2) & next(Z2, Z)
number(0)
number(1)
number(2)
number(3)

less(0,1)
less(1,2)
less(2,3)
number(0)
number(1)
number(2)
number(3)

less(0,1)
less(1,2)
less(2,3)

plus(0,Y,Y) :- number(Y)
plus(X,3,Z) :- less(X1,X) & plus(X1,0,Z)
plus(X,Y,Z) :- less(X1,X) & next(Y,Y1) & plus(X1,Y1,Z)
Directed Graphs
Sample

```
a -> b -> c
```

```
+----+    
|    |    
|  v |    
| a  | b  | c 
```
Nodes and Arcs

node(a)
node(b)
node(c)
node(d)

edge(a,b)
edge(b,c)
edge(c,d)
edge(d,c)
The relation $p$ is true of all nodes with an outgoing arc.

$$ p(X) :- \text{edge}(X,Y) $$

The relation $q$ is true of all nodes with an incoming arc.

$$ q(Y) :- \text{edge}(X,Y) $$
The relation $r$ is true of two nodes if there is an edge from the first to the second or from the second to the first.

\[
\begin{align*}
r(X, Y) & : \text{edge}(X, Y) \\
r(X, Y) & : \text{edge}(Y, X)
\end{align*}
\]

The relation $s$ is the transitive closure of the edge relation.

\[
\begin{align*}
s(X, Y) & : \text{edge}(X, Y) \\
s(X, Z) & : \text{edge}(X, Y) \& s(Y, Z)
\end{align*}
\]
A node is reflexive if and only if it has a self-arc.

\[ \text{reflexive}(X) \iff \text{edge}(X,X) \]

A node is irreflexive if and only if it does \textit{not} have a self-arc.

\[ \text{irreflexive}(X) \iff \text{node}(X) \land \neg \text{edge}(X,X) \]

A graph is nonreflexive if and only if some node is irreflexive.

\[ \text{nonreflexivegraph}() \iff \text{irreflexive}(X) \]

A graph is reflexive if and only if it is \textit{not} irreflexive.

\[ \text{reflexivegraph}() \iff \neg \text{nonreflexivegraph}() \]