Logic Programming

Model Management

Michael Genesereth
Computer Science Department
Stanford University
Operations

View

Operation

t=1

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

\[ g(a,c) \]
\[ g(b,d) \]

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

\[ g(c,a) \]
\[ g(d,b) \]

View

t=2
Changing Worlds

\[ u(a,b) \]

\[ \text{on}(a,b) \]
\[ \text{on}(b,c) \]
\[ \text{on}(d,e) \]
Changing Models

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

\[
\begin{align*}
on(a,b) \\
on(b,c) \\
on(d,e) \\
clear(a) \\
clear(d) \\
table(c) \\
table(e)
\end{align*}
\]

Augment
Programme

Tables versus Triples

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Address</th>
<th>City</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mickey</td>
<td>Mouse</td>
<td>123 Fantasy Way</td>
<td>Anaheim</td>
<td>73</td>
</tr>
<tr>
<td>Bat</td>
<td>Man</td>
<td>321 Cavern Ave</td>
<td>Gotham</td>
<td>54</td>
</tr>
<tr>
<td>Wonder</td>
<td>Woman</td>
<td>987 Truth Way</td>
<td>Paradise</td>
<td>39</td>
</tr>
<tr>
<td>Donald</td>
<td>Duck</td>
<td>555 Quack Street</td>
<td>Hallard</td>
<td>65</td>
</tr>
<tr>
<td>Bugs</td>
<td>Bunny</td>
<td>567 Carrot Street</td>
<td>Rascal</td>
<td>58</td>
</tr>
<tr>
<td>Wiley</td>
<td>Coyote</td>
<td>999 Acme Way</td>
<td>Canyon</td>
<td>61</td>
</tr>
<tr>
<td>Cat</td>
<td>Woman</td>
<td>234 Purrfect Street</td>
<td>Hairball</td>
<td>32</td>
</tr>
<tr>
<td>Tweety</td>
<td>Bird</td>
<td>543</td>
<td>Iotityaw</td>
<td>20</td>
</tr>
</tbody>
</table>

Graph Editing

View Materialization

Constraint Propagation
Tables versus Triples
<table>
<thead>
<tr>
<th>Students:</th>
<th>Departments:</th>
<th>Faculty:</th>
<th>Years:</th>
</tr>
</thead>
<tbody>
<tr>
<td>aaron</td>
<td>architecture</td>
<td>alan</td>
<td>freshman</td>
</tr>
<tr>
<td>belinda</td>
<td>computers</td>
<td>cathy</td>
<td>sophomore</td>
</tr>
<tr>
<td>calvin</td>
<td>english</td>
<td>donna</td>
<td>junior</td>
</tr>
<tr>
<td>george</td>
<td>physics</td>
<td>frank</td>
<td>senior</td>
</tr>
<tr>
<td>Students:</td>
<td>Departments:</td>
<td>Faculty:</td>
<td>Years:</td>
</tr>
<tr>
<td>-----------</td>
<td>--------------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>aaron</td>
<td>architecture</td>
<td>alan</td>
<td>freshman</td>
</tr>
<tr>
<td>belinda</td>
<td>computers</td>
<td>cathy</td>
<td>sophomore</td>
</tr>
<tr>
<td>calvin</td>
<td>english</td>
<td>donna</td>
<td>junior</td>
</tr>
<tr>
<td>george</td>
<td>physics</td>
<td>frank</td>
<td>senior</td>
</tr>
</tbody>
</table>

**Predicate:**
```
student(Student,Department,Advisor,Year)
```

**Dataset:**
```
student(aaron,architecture,alan,freshman)
student(belinda,computers,cathy,sophomore)
student(calvin,english,donna,junior)
student(george,physics,frank,senior)
```
student.major(aaron,architecture)
student.advisor(aaron,alan)
student.year(aaron,freshman)

student.major(belinda,computers)
student.advisor(belinda,cathy)
student.year(belinda,sophomore)

student.major(calvin,physics)
student.advisor(calvin,donna)
student.year(calvin,senior)

student.major(george,physics)
student.advisor(george,frank)
student.year(george,senior)
convert_to_triples ::
  student(S,D,A,Y) ==>~student(S,D,A,Y) &
  student.major(S,D) &
  student.advisor(S,A) &
  student.year(S,Y)

convert_to_triples ::
  student(S,D,A,Y) ==> ~student(S,D,A,Y) &
  ~student(S,D,A,Y) &
  student.major(S,D) &
  student.advisor(S,A) &
  student.year(S,Y)

- or -

convert_to_triples ::
  student(S,D,A,Y) ==> ~student(S,D,A,Y) &
  student.major(S,D) &
  student.advisor(S,A) &
  student.year(S,Y)
convert_to_tables ::
    student.major(S,D) &
    student.advisor(S,A) &
    student.year(S,Y) ==> 
    student(S,D,A,Y)

convert_to_tables ::
    student.major(S,D) ==> ~student.major(S,D)

convert_to_tables ::
    student.advisor(S,D) ==> 
    ~student.advisor(S,D)

convert_to_tables ::
    student.year(S,D) ==> ~student.year(S,D)
student(aaron, architecture, alan, freshman)
student(belinda, computers, cathy, sophomore)
student(calvin, english, donna, junior)
student(george, physics, frank, senior)

convert_to_triples ::
  student(S,D,A,Y) \rightarrow
  student.major(S,D) &
  student.advisor(S,A) &
  student.year(S,Y)

convert_to_triples ::
  student(S,D,A,Y) \rightarrow \neg student(S,D,A,Y)

convert_to_tables ::
  student.major(S,D) &
  student.advisor(S,A) &
  student.year(S,Y) \rightarrow
  student(S,D,A,Y)

convert_to_tables ::
  student.major(S,D) \rightarrow \neg student.major(S,D)

convert_to_tables ::
  student.advisor(S,D) \rightarrow
  \neg student.advisor(S,D)

convert_to_tables ::
  student.year(S,D) \rightarrow \neg student.year(S,D)
student(aaron, architecture, alan, freshman)
student(belinda, computers, cathy, sophomore)
student(calvin, english, donna, junior)
student(george, physics, frank, senior)

convert_to_triples ::
  student(S,D,A,Y) =>
  student.major(S,D) &
  student.advisor(S,A) &
  student.year(S,Y)

convert_to_triples ::
  student(S,D,A,Y) => ~student(S,D,A,Y)

convert_to_tables ::
  student.major(S,D) &
  student.advisor(S,A) &
  student.year(S,Y) =>
  student(S,D,A,Y)

convert_to_tables ::
  student.major(S,D) => ~student.major(S,D)

convert_to_tables ::
  student.advisor(S,D) =>
  ~student.advisor(S,D)

convert_to_tables ::
  student.year(S,D) => ~student.year(S,D)

Transform

Condition: evaluate(remainder(timer(),2),0)
Conclusion: convert_to_triples

Transform

Condition: evaluate(remainder(timer(),2),1)
Conclusion: convert_to_tables
Directed Graphs
Dataset:
  \text{edge}(a,b)
  \text{edge}(b,d)
  \text{edge}(b,e)

Operations:
  \text{copy}(X,Y) - copy outgoing links from \text{X} to \text{Y}.
  \text{invert}(X) - reverse the outgoing arcs from \text{X}.
  \text{insert}(X,Y) - add arc from \text{X} to \text{Y} and all \text{Y} successors.
Dataset:
\{\text{edge}(a,b), \text{edge}(b,d), \text{edge}(b,e)\}

Operation Definition:
\text{copy}(X,Y) :: \text{edge}(X,Z) \implies \text{edge}(Y,Z)

Action: \text{copy}(b,c)
Positive Updates: \{\text{edge}(c,d), \text{edge}(c,e)\}
Negative Updates: {}
Result:
\{\text{edge}(a,b), \text{edge}(b,d), \text{edge}(b,e), \text{edge}(c,d), \text{edge}(c,e)\}
invert(X)

Dataset:
\{edge(a,b), edge(b,d), edge(b,e), edge(c,d), edge(c,e)\}

Operation Definition:
invert(X) - reverse the outgoing arcs from x.
Dataset:
{\text{edge}(a,b), \text{edge}(b,d), \text{edge}(b,e), \text{edge}(c,d), \text{edge}(c,e)}

Operation Definition:
\text{invert}(X) :: \text{edge}(X,Y) \implies \sim\text{edge}(X,Y) \land \text{edge}(Y,X)

Action: \text{invert}(c)
Positive Updates: \{\text{edge}(d,c), \text{edge}(e,c)\}
Negative Updates: \{\text{edge}(c,d), \text{edge}(c,e)\}
Result:
{\text{edge}(a,b), \text{edge}(b,d), \text{edge}(b,e), \text{edge}(d,c), \text{edge}(e,c)}
Dataset:
{edge(a,b), edge(b,d), edge(b,e), edge(d,c), edge(e,c)}

Operation Definition:
insert(X,Y) - add arc from X to Y and all successors of Y.
Dataset:
\{\text{edge}(a,b), \text{edge}(b,d), \text{edge}(b,e), \text{edge}(d,c), \text{edge}(e,c)\}

Operation Definition:
\begin{align*}
\text{insert}(X,Y) & : : \text{edge}(X,Y) \\
\text{insert}(X,Y) & : : \text{edge}(Y,Z) \implies \text{insert}(X,Z)
\end{align*}

Action: \text{insert}(w,b)

Expansion:
\{\text{insert}(w,b), \text{edge}(w,b), \text{insert}(w,d), \text{insert}(w,e), \\
\text{edge}(w,d), \text{edge}(w,e), \text{insert}(w,c), \text{edge}(w,c)\}

Negative Updates: \{\}

Positive Updates:
\{\text{edge}(w,b), \text{edge}(w,d), \text{edge}(w,e), \text{edge}(w,c)\}
Lambda

edge(a,b)
edge(b,d)
edge(b,e)

 Execute

 Library

 copy(X,Y) :: edge(X,Z) ==> edge(Y,Z)
 invert(X) :: edge(X,Y) ==> ¬edge(X,Y) & edge(Y,X)
 insert(X,Y) :: edge(X,Y)
 insert(X,Y) :: edge(Y,Z) ==> insert(X,Z)
Lambda

edge(a,b)
edge(b,d)
edge(b,e)

Library

copy(X,Y) :: edge(X,Z) ==> edge(Y,Z)
invert(X) :: edge(X,Y) ==> ~edge(X,Y) & edge(Y,X)
insert(X,Y) :: edge(X,Y)
insert(X,Y) :: edge(Y,Z) ==> insert(X,Z)
Lambda

edge(a,b)
edge(b,d)
edge(b,e)
edge(c,d)
edge(c,e)

Library

copy(X,Y) :: edge(X,Z) => edge(Y,Z)
invert(X) :: edge(X,Y) => ~edge(X,Y) & edge(Y,X)
insert(X,Y) :: edge(X,Y)
insert(X,Y) :: edge(Y,Z) => insert(X,Z)

Display a menu
edge(a,b)
edge(b,d)
edge(b,e)
edge(c,d)
edge(c,e)

~edge(c,d)
edge(d,c)
~edge(c,e)
edge(e,c)

copy(X,Y) :: edge(X,Z) => edge(Y,Z)
invert(X) :: edge(X,Y) => ~edge(X,Y) & edge(Y,X)
insert(X,Y) :: edge(X,Y)
insert(X,Y) :: edge(Y,Z) => insert(X,Z)
Lambda

Save Revert Sort

edge(a,b)
edge(b,d)
edge(b,e)
edge(d,c)
edge(e,c)

Execute

Action invert(c)
Expand Expand on update
Execute Run on clock tick

Library

Save Revert

copy(X,Y) :: edge(X,Z) => edge(Y,Z)

invert(X) :: edge(X,Y) => ~edge(X,Y) & edge(Y,X)

insert(X,Y) :: edge(X,Y)
insert(X,Y) :: edge(Y,Z) => insert(X,Z)
Lambda

edge(a, b)
edge(b, d)
edge(b, e)
edge(d, c)
edge(e, c)

Execute

Action insert(w, b)

Library

copy(X, Y) :: edge(X, Z) => edge(Y, Z)
invert(X) :: edge(X, Y) => ~edge(X, Y) & edge(Y, X)
insert(X, Y) :: edge(X, Y)
insert(X, Y) :: edge(Y, Z) => insert(X, Z)
Lambda

- edge(a, b)
- edge(b, d)
- edge(b, e)
- edge(d, c)
- edge(e, c)
- edge(w, b)
- edge(w, d)
- edge(w, c)
- edge(w, e)

Library

- copy(X, Y) :: edge(X, Z) => edge(Y, Z)
- invert(X) :: edge(X, Y) => ~edge(X, Y) & edge(Y, X)
- insert(X, Y) :: edge(X, Y) & edge(Y, Z) => insert(X, Z)
View Materialization
A **materialized view** is a view relation that is stored explicitly in the database.

**Ruleset:**
\[
\text{grandparent}(X,Z) \leftarrow \text{parent}(X,Y) \land \text{parent}(Y,Z)
\]

**Base Data:**
- parent(art,bob)
- parent(art,bea)
- parent(bob,cal)
- parent(bob,cam)
- parent(bea,cat)
- parent(bea,coe)

**Materialized View:**
- grandparent(art,cal)
- grandparent(art,cam)
- grandparent(art,cat)
- grandparent(art,coe)
**Ruleset**

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

**Computation Cost for s:**
- \(r\) computed multiple times
- for \(n=2\), unifications = 1242
- for \(n=3\), unifications = 41636
- where \(n\) is the number of objects

**Storage for p:**
- \(O(n^2)\) facts stored
Example with s Materialized

**Ruleset**

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

**Computation Cost for s**

for \( n=2 \), unifications = 8

for \( n=3 \), unifications = 27

**Storage for s**

\( O(n^3) \) in worst case
Ruleste

\[ s(X,Y,Z) \leftarrow r(X) \land r(Y) \land r(Z) \]
\[ r(X) \leftarrow p(X,Y) \land p(Y,Z) \]

Computation Cost for \( s \)

- for \( n=2 \), unifications = 15
- for \( n=3 \), unifications = 40

where \( n \) is the number of objects

Computation Cost for \( r \)

- for \( n=2 \), unifications = 17
- for \( n=3 \), unifications = 55

Storage for \( r \)

\( O(n) \) in worst case
Naive Approach:
On receipt of update, discard materialized data and rematerialize.

Differential Approach:
Write operation definitions for the materialized view.
*Remove rules defining the materialized view.*
On receipt of update, run newly defined operations.
Old View Definitions

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

Operation Definitions

\[ +p(X,Y) :: p(X,Y) \]
\[ +p(X,Y) :: p(Y,Z) \Rightarrow r(X) \]
\[ +p(Y,Z) :: p(X,Y) \Rightarrow r(X) \]

Exercise for viewer: What are the deletion rules?

New View Definition (remove definition of \( r \))

\[ s(X,Y,Z) :- r(X) \& r(Y) \& r(Z) \]
Old Dataset: \{p(a,b), p(b,a), r(a), r(b)\}

Rules:

\begin{align*}
\text{s}(X,Y,Z) & : r(X) \& r(Y) \& r(Z) \\
+p(X,Y) & : p(X,Y) \\
+p(X,Y) & : p(Y,Z) \Rightarrow r(X) \\
+p(Y,Z) & : p(X,Y) \Rightarrow r(X)
\end{align*}

Change Request: +p(d,c)

Update: \{p(d,c)\}

New Dataset:

\begin{align*}
\{p(a,b), p(b,a), p(d,c), r(a), r(b)\}
\end{align*}
Old Dataset: \{p(a,b), p(b,a), p(d,c), r(a), r(b)\}

Rules:
\[
\begin{align*}
s(X,Y,Z) & \leftarrow r(X) \& r(Y) \& r(Z) \\
+p(X,Y) & \leftarrow p(X,Y) \\
+p(X,Y) & \leftarrow p(Y,Z) \Rightarrow r(X) \\
+p(Y,Z) & \leftarrow p(X,Y) \Rightarrow r(X)
\end{align*}
\]

Change Request: \(+p(c,d)\)
Update: \{p(c,d), r(c), r(d)\}

New Dataset:
\{p(a,b), p(b,a), p(c,d), p(d,c), r(a), r(b), r(c), r(d)\}
View Definition

\[ r(X) :- p(X) \]
\[ r(X) :- q(X) \]

Request: \( +r(bob) \)

Positive Update:

\{p(bob)\}?
\{q(bob)\}?
\{p(bob), q(bob)\}? \text{ \textit{What if \( p \) is dead and \( q \) is alive?}}
\{\}\?
View Definition

\[ r(X) := p(X) \]
\[ r(X) := q(X) \]

Update Rule

\[ +r(X) :: \neg q(X) \implies p(X) \]
Constraint Propagation
Constraint Satisfaction Problems

SEND
+MORE
-----
MONEY

6 1 4 5
8 3 5 6
2 4 7 1
6 4 7 3
9 1 4
7 9 1
5 7 2 6 9
4 5 8 7
Ruby Red, Willa White, Betty Blue are having lunch. One is wearing a red skirt, one white, one blue.

Betty to woman in white skirt: *Have you noticed that our skirts have colors different from our names?*

*Which woman is wearing which skirt?*
Ruby Red, Willa White, Betty Blue are having lunch. One is wearing a red skirt, one white, one blue.

Betty to woman in white skirt: *Have you noticed that our skirts have colors different from our names?*

*Why is this problem so easy to solve?*
Symbols: red, white, blue, v, x

Unary predicates Predicates:
  \(\text{color}(C)\) is true if \(C\) is a color.

Ternary Predicates:
  \(c(P, C, v)\) is true if person \(P\) is wearing color \(C\).
  \(c(P, C, x)\) is true if person \(P\) is not wearing color \(C\).

Operations:
  \(c1\) - No person is wearing same color as name.
  \(c2\) - Betty Blue spoke to person in white.
  \(c3\) - Every person has one color and vice versa.
color(red)
color(white)
color(blue)
No person is wearing same color as name.

\[
\text{c1} :: \text{color}(C) \implies c(C,C,x)
\]
Betty Blue spoke to person in white.

\[ c_2 \colon c(\text{blue}, \text{white}, x) \]
Existence Constraints

Every person is wearing some color.

\[c3 \::
\begin{align*}
& c(P, C1, x) \land c(P, C2, x) \land \text{color}(C3) \land \text{mutex}(C1, C2, C3) \\
& \implies c(P, C3, v)
\end{align*}
\]

Every color is worn.

\[c3 \::
\begin{align*}
& c(P1, C, x) \land c(P2, C, x) \land \text{color}(P3) \land \text{mutex}(P1, P2, P3) \\
& \implies c(P3, C, v)
\end{align*}
\]
Nobody is wearing two colors.

\[ c_3 \::\: \]
\[ c(P, C_1, v) \land \text{color}(C_2) \land \text{distinct}(C_1, C_2) \]
\[ \implies c(P, C_2, x) \]

No color is worn by two people.

\[ c_3 \::\: \]
\[ c(P_1, C, v) \land \text{color}(P_2) \land \text{distinct}(P_1, P_2) \]
\[ \implies c(P_2, C, x) \]
<table>
<thead>
<tr>
<th>Person</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td></td>
</tr>
<tr>
<td>w</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>
No person is wearing same color as name.

\[ cl :: \text{color}(C) \implies c(C,C,x) \]
Betty Blue spoke to person in white.

\[ c_2 \defeq c(\text{blue, white, x}) \]
Existence Constraints

c3 ::
\[ c(P, C1, x) \land c(P, C2, x) \land \text{color}(C3) \land \text{mutex}(C1, C2, C3) \implies c(P, C3, v) \]

c3 ::
\[ c(P1, C, x) \land c(P2, C, x) \land \text{human}(P3) \land \text{mutex}(P1, P2, P3) \implies c(P3, C, v) \]
Uniqueness Constraints

c3 ::
c(P, C1, v) & color(C2) & distinct(C1, C2)  
==> c(P, C2, x)

c3 ::
c(P1, C, v) & person(P2) & distinct(P1, P2)  
==> c(P2, C, x)
Existence Constraints

\[ c^3 :: \]
\[ c(P, C_1, x) & c(P, C_2, x) & \text{color}(C_3) & \text{mutex}(C_1, C_2, C_3) \implies c(P, C_3, v) \]

\[ c^3 :: \]
\[ c(P_1, C, x) & c(P_2, C, x) & \text{human}(P_3) & \text{mutex}(P_1, P_2, P_3) \implies c(P_3, C, v) \]
Map Coloring