Logic Programming

Query Optimization

Michael Genesereth
Computer Science Department
Stanford University
Two queries are semantically equivalent if and only if they produce identical results for every dataset.

Query 1:
\[
\text{goal}(X,Y) :\neg p(X) \& r(X,Y) \& q(X)
\]

Query 2:
\[
\text{goal}(X,Y) :\neg p(X) \& q(X) \& r(X,Y)
\]
Different semantically equivalent queries may have dramatically different computational properties.

Query 1: $O(n^4)$

$$\text{goal}(X,Y) :\neg p(X) \& r(X,Y) \& q(X)$$

Query 2: $O(n^3)$

$$\text{goal}(X,Y) :\neg p(X) \& q(X) \& r(X,Y)$$
Types of Reformulation
- Logical - Deleting and/or rearranging subgoals and rules
- Conceptual - changing vocabulary

Examples of Logical Reformulation
- Dropping Rules
- Dropping Subgoals
- Subgoal Ordering

Reformulation
- Reification
- Relationalization
- Relational Reformulation
Rule Removal
Original Rules

\[
\text{goal}(X) := p(X,b) \land q(b) \land r(Z)
\]
\[
\text{goal}(X) := p(X,Y) \land q(Y)
\]

Reformulation

\[
\text{goal}(X) := p(X,Y) \land q(Y)
\]
A rule \( r_1 \) subsumes a rule \( r_2 \) if and only if it is possible to replace some or all of the variables of \( r_1 \) in such a way that the heads are the same and all of subgoals of \( r_1 \) are members of the body of \( r_2 \).

\[
\text{goal}(X) :- p(X,b) \land q(b) \land r(Z)
\]
\[
\text{goal}(X) :- p(X,Y) \land q(Y)
\]

Here, the second rule subsumes the first. We just replace \( X \) in the second rule by itself and replace \( Y \) by \( b \), with the following result.

\[
\text{goal}(X) :- p(X,b) \land q(b)
\]
Accept rule 1 and rule 2 as inputs where (a) the heads are identical and (b) neither rule contains any negations.

(1) Create a substitution in which each variable in rule 2 is bound to a distinct, new symbol.

(2) Create a dataset consisting of the subgoals of rule 2 where all variables are replaced by these bindings.

(3) Substitute the bindings for the head variables in rule 1.

(4) Evaluate this modified rule on the dataset created in (2). If there are answers, then rule 1 subsumes rule 2. If not, then rule 1 does not subsume rule 2.
Example

Inputs

\[
\text{goal}(X) := p(X,Y) \land q(Y) \\
\text{goal}(X) := p(X,b) \land q(b) \land r(Z)
\]

Substitution: \( \{x \leftarrow c1, z \leftarrow c3\} \)

Dataset: \( \{p(c1,b), q(b), r(c3)\} \)

Evaluate: \( p(c1,Y) \land q(Y) \)

Result: \( \{x \leftarrow c1, Y \leftarrow b\} \)

The first rule \textit{does} subsume the second.
Example

Inputs

\[ \text{goal}(X) :- p(X,b) \land q(b) \land r(Z) \]
\[ \text{goal}(X) :- p(X,Y) \land q(Y) \]

Substitution: \{X\leftarrow c1, Y\leftarrow c2\}

Dataset: \{p(c1,c2), q(c2)\}

Evaluate: \ p(c1,b) \land q(b) \land r(Z)
Result: failure

The first rule does not subsume the second.
Rule Removal Technique

Compare every rule to every other rule (quadratic). If one rule subsumes another, okay to dropped the subsumed rule.

The technique is *sound* in that it is guaranteed to produce equivalent query.

In *the absence of any constraints* on the datasets to which the rules are applied, it is also guaranteed to be *complete* in that all surviving rules are needed for some dataset.

In *the face of constraints*, it may be possible to drop rules that are not detected by this method, i.e. *not complete*. 
If heads are not identical, they can sometimes be made identical by consistently replacing variables while avoiding clashes.

Original rules:

\[
\begin{align*}
goal(X) & :\, p(X,b) \land q(b) \land r(Z) \\
goal(U) & :\, p(U,V) \land q(V)
\end{align*}
\]

Equivalent rules:

\[
\begin{align*}
goal(X) & :\, p(X,b) \land q(b) \land r(Z) \\
goal(X) & :\, p(X,V) \land q(V)
\end{align*}
\]

There are extensions for dealing with rules involving negations and for dealing with constraints.
Subgoal Removal
Original Rule:

\[
\text{goal}(X,Y) :- p(X,Y) \land q(Y) \land q(Z)
\]

Equivalent Reformulation:

\[
\text{goal}(X,Y) :- p(X,Y) \land q(Y)
\]
Accept query rule as input.

(1) Create new query with rule head and empty body.

(2) Create a substitution with an entry for each head variable and a distinct constant.

(3) For each subgoal, (a) augment the substitution with bindings for the remaining variables in the other subgoals. (b) Create a dataset by replacing variables in those subgoals with their bindings. (c) Replace variables in chosen subgoal with bindings, leaving unbound variables as is. (d) Evaluate query. If fail, add to body of new query. Otherwise, drop.

Output the new query.
Query: \( \text{goal}(X,Y) \leftarrow p(X,Y) \& q(Y) \& q(Z) \)

Initial binding list - \{X←c1, Y←c2\}

Subgoal: \( p(X,Y) \). Binding list: \{X←c1, Y←c2, Z←c3\}.
Create dataset \{q(c2), q(c3)\}.
Evaluate \( p(c1,c2) \). Fail. So add \( p(X,Y) \) to new query.

Subgoal: \( q(Y) \). Binding list: \{X←c1, Y←c2, Z←c3\}.
Create dataset \{p(c1,c2), q(c3)\}.
Evaluate \( q(c2) \). Fail. So add \( q(Y) \) to new query.

Subgoal: \( q(Z) \). Binding list: \{X←c1, Y←c2\}.
Create dataset \{p(c1,c2), q(c2)\}.
Evaluate \( q(Z) \). Succeed. So drop.

Output: \( \text{goal}(X,Y) \leftarrow p(X,Y) \& q(Y) \)
Original Rule

\[ \text{goal}(X,Y) \leftarrow \ p(X,Y) \ & \ q(Y) \ & \ q(Z) \]

Reformulation

\[ \text{goal}(X,Y) \leftarrow \ p(X,Y) \ & \ q(Y) \]
Original Rule

\[ \text{goal}(X,Y) :- \ p(X,Y) \ & \ q(X,Y) \ & \ p(X,Z) \ & \ q(X,Z) \]

Reformulation

\[ \text{goal}(X,Y) :- \ p(X,Y) \ & \ q(X,Y) \]
Subgoal Ordering
Original Rule

\[ \text{goal}(X,Y) :- \ p(X) \ & \ r(X,Y) \ & \ q(X) \]

Reformulation

\[ \text{goal}(X,Y) :- \ p(X) \ & \ q(X) \ & \ r(X,Y) \]
Analysis

Original Rule

\[
\text{goal}(X,Y) \leftarrow p(X) \land r(X,Y) \land q(X)
\]

\[
(n^2 + 2n) + n^*((n^2 + 2n) + n^*(n^2 + 2n)) = n^4 + 3n^3 + 3n^2 + 2n
\]

Reformulation

\[
\text{goal}(X,Y) \leftarrow p(X) \land q(X) \land r(X,Y)
\]

\[
(n^2 + 2n) + n^*((n^2 + 2n) + 1*(n^2 + 2n)) = 2n^3 + 5n^2 + 2n
\]
Subgoal Ordering Technique

Accept query rule as input.

(1) Create new query with head of input and empty body.

(2) Iterate through subgoals. On encountering one with all variables bound, add to new query and remove from original query. If none found, remove first subgoal, add to new query, and repeat.

Output the new query.

Example:

```prolog
goal(X,Y) :- p(X) & r(X,Y) & q(X)
goal(X,Y) :- p(X) & q(X) & r(X,Y)
```
Example
Example

SEND
+MORE
-----
MONEY
goal(S,E,N,D,M,O,R,Y) :-
digit(S) & digit(E) & digit(N) & digit(D) &
digit(M) & digit(O) & digit(R) & digit(Y) &
M!=0 & M!=S & M!=E & M!=N & M!=D &
O!=S & O!=E & O!=N & O!=D & O!=M &
evaluate(S*1000+E*100+N*10+D,X) &
evaluate(M*1000+O*100+R*10+E,Y) &
evaluate(M*10000+O*1000+N*100+E*10+Y,Z) &
evaluate(plus(X,Y),Z)
Computational Analysis

Data

digit(1)      digit(6)
digit(2)      digit(7)
digit(3)      digit(8)
digit(4)      digit(9)
digit(5)      digit(0)

Rule

goal(S,E,N,D,M,O,R,Y) :-
digit(S) & digit(E) & digit(N) & digit(D) &
digit(M) & digit(O) & digit(R) & digit(Y) & ...

Analysis

10x10x10x10x10x10x10x10 = 10^8 = 100,000,000 cases
111,111,110 unifications
Running time ~ minutes
Another Solution

goal(S,E,N,D,M,O,R,Y) :-
    digit(S) & S!=0 &
    digit(E) & E!=S &
    digit(N) & N!=S & N!=E &
    digit(D) & D!=S & D!=E & D!=N &
    digit(M) & M!=0 & M!=S & M!=E & M!=N & M!=D &
    digit(O) & O!=S & O!=E & O!=N & O!=D & O!=M &
    digit(R) & R!=S & R!=E & R!=N & R!=D &
        R!=M & R!=O &
    digit(Y) & Y!=S & Y!=E & Y!=N & Y!=D &
        Y!=M & Y!=O & Y!=R &
    evaluate(S*1000+E*100+N*10+D,X) &
    evaluate(M*1000+O*100+R*10+E,Y) &
    evaluate(M*10000+O*1000+N*100+E*10+Y,Z) &
    evaluate(plus(X,Y),Z)
Rule

goal(S,E,N,D,M,O,R,Y) :-

digit(S) & S!=0 &
digit(E) & E!=S &
digit(N) & N!=S & N!=E &
digit(D) & D!=S & D!=E & D!=N &
digit(M) & M!=0 & M!=S & M!=E & M!=N & M!=D &
digit(O) & O!=S & O!=E & O!=N & O!=D & O!=M &
digit(R) & R!=S & R!=E & R!=N & R!=D &
    R!=M & R!=O &
digit(Y) & Y!=S & Y!=E & Y!=N & Y!=D &
    Y!=M & Y!=O & Y!=R & ...

Analysis

10x9x8x7x6x5x4x3 = 1,814,400 cases
5,989,558 / 7,921,010 unifications
Interpreted ~ 40 seconds  Compiled ~ 4 seconds
Rule

\[
goal(S,E,N,D,M,O,R,Y) \leftarrow \\
\text{digit}(D) \ \& \\
\text{digit}(E) \ \& \ \text{mutex}(E,D) \ \& \\
\text{digit}(Y) \ \& \ \text{mutex}(Y,D,E) \ \& \\
\text{digit}(S) \ \& \ \text{distinct}(S,0) \ \& \ \text{mutex}(S,D,E,Y) \ \& \\
\text{digit}(N) \ \& \ \text{mutex}(N,D,E,Y,S) \ \& \\
\text{digit}(M) \ \& \ \text{distinct}(M,0) \ \& \ \text{mutex}(M,D,E,Y,S,N) \ \& \\
\text{digit}(O) \ \& \ \text{mutex}(O,D,E,Y,S,N,M) \ \& \\
\text{digit}(R) \ \& \ \text{mutex}(R,D,E,Y,S,N,M,O) \ \& \ldots
\]

Analysis

\[4,002,398 \ / \ 7,921,010\] unifications
Interpreted < 30 seconds
Computational Analysis

Rule

goal(S,E,N,D,M,O,R,Y) :-
digit(D) &
digit(E) & mutex(E,D) &
digit(Y) & mutex(Y,D,E) &
\text{evaluate(remainder(plus(D,E),10),Y)} &
digit(S) & distinct(S,0) & mutex(S,D,E,Y) &
digit(N) & mutex(N,D,E,Y,S) &
digit(M) & distinct(M,0) & mutex(M,D,E,Y,S,N) &
digit(O) & mutex(O,D,E,Y,S,N,M) &
digit(R) & mutex(R,D,E,Y,S,N,M,O) & ...  

Analysis

438,728 / 7,921,010 unifications
Interpreted ~ 4 seconds
Data

digit(9)
digit(5)
digit(6)
digit(7)
digit(7)
digit(1)
digit(0)
digit(0)
digit(8)
digit(8)
digit(2)
digit(3)
digit(4)

Analysis

1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36 / 7,921,010 unifications
Interpreted ~ 0 seconds
Computational Analysis

Data
digit(9)
digit(5)
digit(6)
digit(7)
digit(7)
digit(1)
digit(0)
digit(0)
digit(8)
digit(2)
digit(3)
digit(4)

Analysis

\[ 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = 36 \]
\[ / \ 7,921,010 \text{ unifications} \]
Interpreted \( \sim 0 \) seconds
Narrow and Wide Relations
Represent wide relations as collections of binary relations.

Wide Relation:
\[ \text{student}(\text{Student,Department,Advisor,Year}) \]

Binary Relations:
\[ \text{student.major}(\text{Student,Department}) \]
\[ \text{student.advisor}(\text{Student,Faculty}) \]
\[ \text{student.year}(\text{Student,Year}) \]

Always works when there is a field of the wide relation (called the key) that uniquely specifies the values of the other elements. If none exists, possible to create one.
Wide Relation:

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

Triples:

\[ p1(a, d) \quad p2(a, e) \]
\[ p1(b, d) \quad p2(b, e) \]
\[ p1(c, d) \quad p2(c, e) \]
Wide Relation:
\[
p(a,d,e) \\
p(b,d,e) \\
p(c,d,e)
\]

Query: \texttt{goal(X) :- p(X,d,e)}

Cost without indexing: 3  
Cost with indexing: 3

Triples:
\[
p1(a,d)    p2(a,e) \\
p1(b,d)    p2(b,e) \\
p1(c,d)    p2(c,e)
\]

Query: \texttt{goal(X) :- p1(X,d) \& p2(X,e)}

Cost without indexing: 24  
Cost with indexing: 9
Minimal Spanning Trees
Social Isolation Cells
**Vocabulary:**
- People - a, b, c, d, e, f, g, h, i, j, ...
- Interaction - r/2

**Example:**
- \( r(a, b) \)
- \( r(a, e) \)
- \( r(b, a) \)
- \( r(b, c) \)
- \( r(c, b) \)
- \( r(d, e) \)
- \( r(e, a) \)
- \( r(e, d) \)

*NB: Possible to represent undirected with only one factoid per arc rather than two, but we will ignore that for now.*
Are two people are in the same cell?

\[
goal(a,e) \leftarrow r(a,e)
\]

\[
goal(a,e) \leftarrow r(a,Y) \land r(Y,e)
\]

\[
goal(a,e) \leftarrow r(a,Y1) \land r(Y1,Y2) \land r(Y2,e)
\]

\[
goal(a,e) \leftarrow r(a,Y1) \land r(Y1,Y2) \land r(Y2,Y3) \land r(Y3,e)
\]

Number of unifications for \( goal(a,e) \) (with indexing):

\[
8
\]

\[
8 + 4 \times 8 = 40
\]

\[
8 + 4 \times (8 + 4 \times 8) = 168
\]

\[
8 + 4 \times (8 + 4 \times (8 + 4 \times 8)) = 680
\]

Total: 896
Precompute and store the transitive closure of $r$

$r(a,b) \quad r(b,a) \quad r(c,a) \quad r(d,a) \quad r(e,a)$
$r(a,c) \quad r(b,c) \quad r(c,b) \quad r(d,b) \quad r(e,b)$
$r(a,d) \quad r(b,d) \quad r(c,d) \quad r(d,c) \quad r(e,c)$
$r(a,e) \quad r(b,e) \quad r(c,e) \quad r(d,e) \quad r(e,d)$

Are two people are in the same cell?

$\text{goal}(a,e) \ :- \ r(a,e)$

Number of unifications for $\text{goal}(a,e)$ (with indexing):

8

Number of factoids for $n$ objects:

$8*n$
Assign a number for each group and store with people:

\[
\begin{align*}
  & r(a,1) \quad r(f,2) \quad \ldots \\
  & r(b,1) \quad r(g,2) \quad \ldots \\
  & r(c,1) \quad r(h,2) \quad \ldots \\
  & r(d,1) \quad r(i,2) \quad \ldots \\
  & r(e,1) \quad r(j,2) \quad \ldots 
\end{align*}
\]

Are two people are in the same cell?
\[
\text{goal}(a,e) :\neg \ r(a,N) \ & \ r(e,N)
\]

Number of unifications for \( \text{goal}(a,e) \) (with indexing):

\[2\]

Number of factoids for \( n \) objects:

\[n\]
\begin{align*}
p(a, 1) \\
p(b, 1) \\
p(c, 1) \\
p(d, 1) \\
p(e, 1)
\end{align*}

Pattern: \texttt{goal(a,e)}

Query: \texttt{p(a,N) \& p(e,N)}

\text{Show} \quad \text{Next} \quad \text{100 result(s)} \quad \text{Autorefresh}

2 unification(s)

\texttt{goal(a,e)}