Logic Programming
Functional Queries

Michael Genesereth
Computer Science Department
Stanford University

Relational Queries

True or False questions:

e.g. *Is Art the parent of Bob?*

Fill-in-the-blanks questions:

e.g. *Art is the parent of ____?*

e.g. ____ *is the parent of Bob?*

e.g. ____ *is the parent of ____?*

Compound questions:

e.g. *Is Art the parent of Bob* ***or*** *the parent of Bud?*

e.g. ____ *has sons* ***and*** *no daughters?*

Functional Queries

Numbers and Strings:

e.g. 2 + 2?

e.g. *Concatenation of "abc" and "def"?*

Aggregates:

e.g. *Number of children of Art?*

e.g. *Number of people with daughters?*

Combinations:

e.g. *maximum of 2+3 and 2*3?*

e.g. *number Art's sons + number of Art's daughters?*

Transitions

Additions

e.g. *Add the fact that Art is the parent of Bill*.

Deletions

e.g. *Drop the fact that Art is the parent of Bill*.

Conditional Changes:

e.g. *Add grandparent facts implied by parent facts*.

e.g. *Replace parent facts with corresponding child facts*.

Functional Queries

Predefined Concepts

Evaluable Functions

Arithmetic Functions (e.g. plus, times, min, max, etc.)
String functions (e.g. concatenate, string matching, etc.)
Set functions (e.g. sum, average, standard deviation, etc.)
Other (e.g. converting between formulas and strings, etc.)

Aggregates

Sets (e.g. set of objects with given properties)
Counts (e.g. count of objects with given properties)

http://epilog.stanford.edu/documentation/epilog/vocabulary.php

Evaluable Terms

Evaluable terms - constants, variables, $f(t_1,\ldots,t_n)$

f is a predefined function or user-defined function (*later*)

i.e. ***not*** *a constructor / function constant*

$t_1, \ldots, t_n$ are evaluable terms

Examples

$\texttt{plus(2,3)} \longrightarrow \texttt{5}$

$\texttt{stringappend(“abc”,”def”)} \longrightarrow \texttt{"abcdef"}$

$\texttt{stringify(vinay)} \longrightarrow \texttt{"vinay"}$

$\texttt{symbolize(“vinay”)} \longrightarrow \texttt{vinay}$

$\texttt{min(plus(2,3),times(2,3))} \longrightarrow \texttt{5}$

Many predefined functions are variadic, e.g. `plus(2,3,4)`.

Logical Terms

Logical operators are used to create sets of answers as terms and then count, add, average those sets.

```
countofall
setofall
choose
if
```

Dataset `{p(a,b),p(a,c),p(b,d)}`

Examples

```
countofall(Z,p(a,Z))      ⟶  2
setofall(Z,p(a,Z))        ⟶  [b,c]
setofall([X,Y],p(X,Y))    ⟶ [[a,b],[a,c],[b,d]]
```

Sentences Inside Evaluable Terms

Dataset `{p(a,b),p(a,c),p(b,d)}`

Example

```
if(exist(Y,p(a,Y)&p(Y,Z)),yes,true,no)
```

Result `yes`

Example

```
if(exist(Y,p(b,Y)&p(Y,Z)),yes,true,no)
```

Result `no`

Evaluable Terms Inside Sentences

Dataset `{h(a,2),w(a,3),h(b,4),w(b,2)}`

Possible Query

```
goal(X,times(H,W)) :- h(X,H) & w(X,W)
```

Results

```
goal(a,times(2,3))
goal(b,times(4,2))
```

Evaluate Predicate

```
evaluate(x,v)
```

x is a term
v is the value of x

Examples

```
goal :- evaluate(times(2,3),6)

goal :- evaluate(plus(times(2,3),4),10)

goal(X,A) :-
  h(X,H) & w(X,W) & evaluate(times(H,W),A)
```

Safety: unbound variables in ***second*** *argument only.*

Nesting Okay

Example

```
goal(Z) :-
  evaluate(min(plus(2,3),times(2,3)),Z)
```

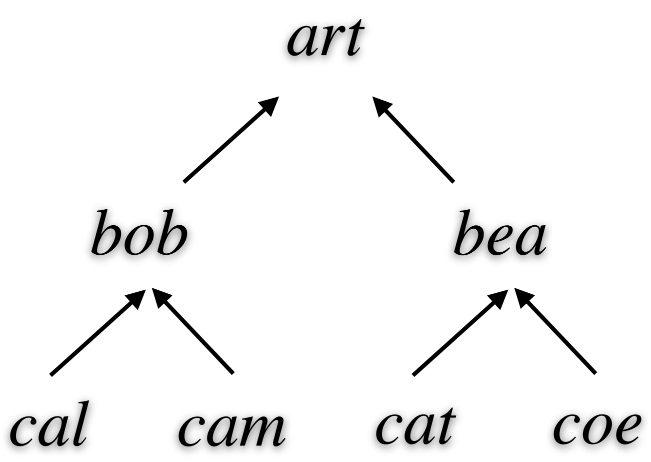
Result

```
goal(5)
```

Kinship

Dataset

```
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
```


$$\begin{tikzcd}
 & & art & \\
 & bob \arrow[ur] & & bea \arrow[ul] \\
cal \arrow[ur] & cam \arrow[u] & cat \arrow[ur] & coe \arrow[u]
\end{tikzcd}$$

Grandchildren

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
```

Query: find grandparents and grandchildren

```
goal(X,Z) :- parent(X,Y) & parent(Y,Z)
```

Result:

```
goal(art,cal)
goal(art,cam)
goal(art,cat)
goal(art,coe)
```

Grandchildren

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(bob,cal)
parent(bob,cam)
parent(bea,cat)
parent(bea,coe)
```

Evaluable Term:

```
setofall(Z,exists(Y,parent(art,Y)&parent(Y,Z))
```

Value: `[cal,cam,cat,coe]`

Evaluable Term:

```
countofall(Z,exists(Y,parent(art,Y)&parent(Y,Z))
```

Value: `4`

Children

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,cal)
```

Query: find every person with *exactly* two children

```
goal(X) :-
  parent(X,Y) &
  parent(X,Z) &
  distinct(Y,Z) &
  ~exists(W,parent(X,W) & mutex(W,Y,Z))
```

Children

Dataset:

```
parent(art,bob)
parent(art,bea)
parent(art,ben)
parent(bob,cal)
```

Query: find every person with *exactly* two children

```
goal(X) :-
  evaluate(countofall(W,parent(X,W)),2)
```

Cryptarithmetic

Example

```
  SEND
 +MORE
 -----
 MONEY
```

One Solution

Data

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

Query

```
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) &
  mutex(S,E,N,D,M,O,R,Y) &
  distinct(S,0) & distinct(M,0) &
  evaluate(S*1000+E*100+N*10+D,U) &
  evaluate(M*1000+O*100+R*10+E,V) &
  evaluate(M*10000+O*1000+N*100+E*10+Y,W) &
  evaluate(plus(U,V),W)
```

Computational Analysis

Data

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

Query

```
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 instances

Running time ~ minutes

See next week to see how to do in less than a second.

One Solution

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

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) &
  S!=0 & E!=S & N!=S & N!=E & D!=S & D!=E & D!=N &
  M!=0 & M!=S & M!=E & M!=N & M!=D &
  O!=S & O!=E & O!=N & O!=D & O!=M &
  R!=S & R!=E & R!=N & R!=D & R!=M & R!=O &
  Y!=S & Y!=E & Y!=N & Y!=D & Y!=M & Y!=O & Y!=R
  evaluate(S*1000+E*100+N*10+D,U) &
  evaluate(M*1000+O*100+R*10+E,V) &
  evaluate(M*10000+O*1000+N*100+E*10+Y,W) &
  evaluate(plus(U,V),W)
```

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,U) &
  evaluate(M*1000+O*100+R*10+E,V) &
  evaluate(M*10000+O*1000+N*100+E*10+Y,W) &
  evaluate(plus(U,V),W)
```

Another Solution

```
goal(S,E,N,D,M,O,R,Y) :-
  digit(S) & mutex(S,0) &
  digit(E) & mutex(S,E) &
  digit(N) & mutex(S,E,N) &
  digit(D) & mutex(S,E,N,D) &
  digit(M) & distinct(M,0) & mutex(S,E,N,D,M) &
  digit(O) & mutex(S,E,N,D,M,O) &
  digit(R) & mutex(S,E,N,D,M,O,R) &
  digit(Y) & mutex(S,E,N,D,M,O,R,Y) &
  evaluate(S*1000+E*100+N*10+D,XX) &
  evaluate(M*1000+O*100+R*10+E,YY) &
  evaluate(M*10000+O*1000+N*100+E*10+Y),ZZ) &
  evaluate(XX+YY,ZZ)
```

Computational Analysis

Goal

```
goal(S,E,N,D,M,O,R,Y) :-
  digit(S) & mutex(S,0) &
  digit(E) & mutex(S,E) &
  digit(N) & mutex(S,E,N) &
  digit(D) & mutex(S,E,N,D) &
  digit(M) & distinct(M,0) & mutex(S,E,N,D,M) &
  digit(O) & mutex(S,E,N,D,M,O) &
  digit(R) & mutex(S,E,N,D,M,O,R) &
  digit(Y) & mutex(S,E,N,D,M,O,R,Y) & ...
```

Analysis

10x9x8x7x6x5x4x3 = 1,814,400 cases

5,989,558 unifications

Running time ~20 seconds

Computational Analysis

Goal

```
goal(S,E,N,D,M,O,R,Y) :-
  digit(S) & mutex(S,0) &
  digit(E) & mutex(S,E) &
  digit(N) & mutex(S,E,N) &
  digit(D) & mutex(S,E,N,D) &
  digit(M) & same(M,1) & mutex(S,E,N,D,M) &
  digit(O) & mutex(S,E,N,D,M,O) &
  digit(R) & mutex(S,E,N,D,M,O,R) &
  digit(Y) & mutex(S,E,N,D,M,O,R,Y) & ...
```

Analysis

10x9x8x7x6x5x4x3 = 320,400 cases

699,858 unifications

Running time < 2 seconds

Computational Analysis

Data

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

Analysis

$1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 =$ **36** cases

Interpreted ~ 0 seconds

Computational Analysis

Data

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

Analysis

1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 = **36** unifications

Interpreted ~ 0 seconds