Logical Query Languages

Motivation:

1. Logical rules extend more naturally to recursive queries than does relational algebra.
   ♦ Used in SQL3 recursion.

2. Logical rules form the basis for many information-integration systems and applications.
Datalog Example

\[
\text{Likes}(\text{drinker}, \text{beer}) \\
\text{Sells}(\text{bar}, \text{beer}, \text{price}) \\
\text{Frequents}(\text{drinker}, \text{bar})
\]

\[
\text{Happy}(d) \leftarrow \\
\quad \text{Frequents}(d, \text{bar}) \land \\
\quad \text{Likes}(d, \text{beer}) \land \\
\quad \text{Sells}(\text{bar}, \text{beer}, p)
\]

- Above is a rule.
- Left side = head.
- Right side = body = AND of subgoals.
- Head and subgoals are atoms.
  - Atom = predicate and arguments.
  - Predicate = relation name or arithmetic predicate, e.g. \(<\).
  - Arguments are variables or constants.
- Subgoals (not head) may optionally be negated by NOT.
Meaning of Rules

Head is true of its arguments if there exist values for *local* variables (those in body, not in head) that make all of the subgoals true.

- If no negation or arithmetic comparisons, just natural join the subgoals and project onto the head variables.

Example

Above rule equivalent to $\text{Happy}(d) = \pi_{\text{drinker}}(\text{Frequents} \bowtie \text{Likes} \bowtie \text{Sells})$
Evaluation of Rules

Two, dual, approaches:

1. **Variable-based**: Consider all possible assignments of values to variables. If all subgoals are true, add the head to the result relation.

2. **Tuple-based**: Consider all assignments of tuples to subgoals that make each subgoal true. If the variables are assigned consistent values, add the head to the result.

Example: Variable-Based Assignment

\[
S(x, y) \leftarrow R(x, z) \text{ AND } R(z, y) \\
\text{AND NOT } R(x, y)
\]

\[
R =
\]

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
• Only assignments that make first subgoal true:
  1. \( x \rightarrow 1, \; z \rightarrow 2. \)
  2. \( x \rightarrow 2, \; z \rightarrow 3. \)

• In case (1), \( y \rightarrow 3 \) makes second subgoal true. Since \( (1, 3) \) is not in \( R \), the third subgoal is also true.
  ✦ Thus, add \((x, y) = (1, 3)\) to relation \( S \).

• In case (2), no value of \( y \) makes the second subgoal true. Thus, \( S = \)

\[
\begin{array}{c|c}
A & B \\
1 & 3 \\
\end{array}
\]
Example: Tuple-Based Assignment

Trick: start with the positive (not negated), relational (not arithmetic) subgoals only.

\[ S(x, y) \leftarrow R(x, z) \land R(z, y) \land \text{NOT } R(x, y) \]

\[ R = \]

\[
\begin{array}{cc}
A & B \\
1 & 2 \\
2 & 3 \\
\end{array}
\]

- Four assignments of tuples to subgoals:

\[
\begin{array}{cc}
R(x, z) & R(z, y) \\
(1, 2) & (1, 2) \\
(1, 2) & (2, 3) \\
(2, 3) & (1, 2) \\
(2, 3) & (2, 3) \\
\end{array}
\]

- Only the second gives a consistent value to \( z \).
- That assignment also makes \( \text{NOT } R(x, y) \) true.
- Thus, \( (1, 3) \) is the only tuple for the head.
Safety

A rule can make no sense if variables appear in funny ways.

Examples

- \( S(x) \leftarrow R(y) \)
- \( S(x) \leftarrow \text{NOT} R(x) \)
- \( S(x) \leftarrow R(y) \text{ AND } x < y \)

In each of these cases, the result is infinite, even if the relation \( R \) is finite.

- To make sense as a database operation, we need to require three things of a variable \( x \) (= definition of safety). If \( x \) appears in either
  1. The head,
  2. A negated subgoal, or
  3. An arithmetic comparison,

then \( x \) must also appear in a nonnegated, “ordinary” (relational) subgoal of the body.

- We insist that rules be safe, henceforth.
Datalog Programs

- A collection of rules is a *Datalog program*.
- Predicates/relations divide into two classes:
  - EDB = *extensional database* = relation stored in DB.
  - IDB = *intensional database* = relation defined by one or more rules.
- A predicate must be IDB or EDB, not both.
  - Thus, an IDB predicate can appear in the body or head of a rule; EDB only in the body.
Example

Convert the following SQL (Find the manufacturers of the beers Joe sells):

\[
\text{Beers(name, manf)} \\
\text{Sells(bar, beer, price)}
\]

SELECT manf \\
FROM Beers \\
WHERE name IN( \\
    SELECT beer \\
    FROM Sells \\
    WHERE bar = 'Joe’s Bar'
)

; to a Datalog program.

JoeSells(b) <- \\
    Sells('Joe’s Bar', b, p) \\
Answer(m) <- \\
    JoeSells(b) AND Beers(b,m)

- Note: Beers, Sells = EDB; JoeSells, Answer = IDB.
Expressive Power of Datalog

- Nonrecursive Datalog = relational algebra.
- Datalog simulates SQL select-from-where without aggregation and grouping.
- Recursive Datalog expresses queries that cannot be expressed in SQL.
- But none of these languages have full expressive power (Turing completeness).
Relational Algebra to Datalog

- Text has constructions for each of the operators of R.A.
  - Only hard part: selections with OR’s and NOT’s.

- Simulate a R.A. expression in Datalog by creating an IDB predicate for each interior node and using the construction for the operator at that node.
Example: Find the bars that sell two different beers at the same price.

\[
\begin{array}{c}
\pi_{\text{bar}} \\
\sigma_{\text{beer} \neq \text{beer1}} \\
\bowtie \\
\rho_{S(\text{bar,beer1,price})} \\
\text{Sells} & \text{Sells} \\
\end{array}
\]

\begin{align*}
R1(\text{bar,beer1,beer,price}) & \leftarrow \\
& \text{Sells(\text{bar,beer1,price}) AND} \\
& \text{Sells(\text{bar,beer,price})}; \\
R2(\text{bar,beer1,beer,price}) & \leftarrow \\
& \text{R1(\text{bar,beer1,beer,price}) AND} \\
& \text{beer1} \leftrightarrow \text{beer}; \\
\text{Answer(\text{bar})} & \leftarrow \\
& \text{R2(\text{bar,beer1,beer,price})};
\end{align*}
Datalog to Relational Algebra

- General rule is complex; the following often works for single rules:
  - Problems not handled: constant arguments and variables appearing twice in the same atom.
  - Can you provide the necessary fixes?
  1. Use $\rho$ to create for each relational subgoal a relation whose schema is the variables of that subgoal.
  2. Handle negated subgoals by finding an expression for the finite set of all possible values for each of its variables ($\pi$ a suitable column) and take their product. Then subtract.
  3. Natural join the relations from (1), (2).
  4. Get the effect of arithmetic comparisons with $\sigma$.
  5. Project onto head with $\pi$.
- Several rules for same predicate: use $\cup$. 
Example

\[ S(x,y) \leftarrow R(x,z) \text{ AND } R(z,y) \]
\[ \text{AND NOT } R(x,y) \]

\[ S1(x,y,z) := \rho_{R1(x,z)}(R) \bowtie \rho_{R2(z,y)}(R); \]
\[ S2(x,y) := \pi_x(S1) \times \pi_y(S1); \]
\[ S3(x,y) := S2 - \rho_{R3(x,y)}(R); \]
\[ S(x,y) := \pi_{x,y}(S1(x,y,z) \bowtie S3(x,y)); \]
Recursion

- IDB predicate $P$ depends on predicate $Q$ if there is a rule with $P$ in the head and $Q$ in a subgoal.

- Draw a graph: nodes = IDB predicates, arc $P \rightarrow Q$ means $P$ depends on $Q$.

- Cycles iff recursive.

Recursive Example

\[
\begin{align*}
\text{Sib}(x,y) & \leftarrow \text{Par}(x,p) \text{ AND } \text{Par}(y,p) \\
& \quad \text{AND } x \neq y \\
\text{Cousin}(x,y) & \leftarrow \text{Sib}(x,y) \\
\text{Cousin}(x,y) & \leftarrow \text{Par}(x,xp) \\
& \quad \text{AND } \text{Par}(y,yp) \\
& \quad \text{AND } \text{Cousin}(xp,ypp)
\end{align*}
\]
Iterative Fixed-Point Evaluates Recursive Rules

Start
IDB = ∅

Apply rules to IDB, EDB

Change to IDB?

yes no

done
Example

EDB Par =

```
    a       d
     / \     / \
    b   c   e
   /   /   /
  f   g   h
 / \  /  /
j  k i
```

- Note, because of symmetry, Sib and Cousin facts appear in pairs, so we shall mention only \((x, y)\) when both \((x, y)\) and \((y, x)\) are meant.
<table>
<thead>
<tr>
<th>Round</th>
<th>Sib</th>
<th>Cousin</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>Round 1</td>
<td>$(b, c), (c, e)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td></td>
<td>$(g, h), (j, k)$</td>
<td></td>
</tr>
<tr>
<td>Round 2</td>
<td>$(b, c), (c, e)$</td>
<td>$(b, c), (c, e)$</td>
</tr>
<tr>
<td></td>
<td>$(g, h), (j, k)$</td>
<td>$(g, h), (j, k)$</td>
</tr>
<tr>
<td>Round 3</td>
<td>$(f, g), (f, h)$</td>
<td>$(f, g), (f, h)$</td>
</tr>
<tr>
<td></td>
<td>$(g, i), (h, i)$</td>
<td>$(g, i), (h, i)$</td>
</tr>
<tr>
<td></td>
<td>$(i, k)$</td>
<td>$(i, k)$</td>
</tr>
<tr>
<td>Round 4</td>
<td>$(k, k)$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td></td>
<td>$(i, j)$</td>
<td></td>
</tr>
</tbody>
</table>