Logical Query Languages

Datalog
- Logical query language for the relational model
- Consists of “if-then” rules made up of atoms:
  - relational: predicates corresponding to relations
    > EDB extensional database (stored relations)
    > IDB intensional database (relations defined by rules)
  - arithmetic

Datalog example
Example:
database schema:
Movie(title, year, length, inColor, studionName, producerC#)
Contracts(starName, studioName, title, year, salary)

relational atom: Movie (t, y, l, c, s, p)
arithmetic atom: l > 100

Datalog Rules
Rule: head ← body
Head: a relational atom (no EDB predicates!)
Body: one or more atoms called subgoals

Example:
datalog rule: LongMovie(t, y) ← Movie(t, y, l, c, s, p) AND l >= 10
Relational Algebra...
Relational Calculus...

Interpreting Datalog Rules
Variables: - distinguished – appear in the head
- nondistinguished – appear in the body

Interpreting rules
the head is true of the distinguished variables if there exist values of the non-distinguished variables that make all subgoals of the body true.

Safe Datalog Rules
A rule is safe if each distinguished and nondistinguished variable appears in at least one nonnegated relational atom.

Note: only safe rules are allowed
Unsafe Datalog Rules

Example:

E(w) ← NOT Movies(t, y, l, c, s, p)

Years(w) ← Movies(t, y, l, c, s, p) AND w < y

Note: in each case an infinity of w’s can satisfy the rule, even though Movies is a finite relation.

Algorithms for Evaluating Datalog Rules

Variable-based: Consider all possible assignments to the variable of the body. If the assignment makes the body true, add the tuple for the head to the result.

Tuple-based: Consider all assignments of tuples from the nonnegated relational subgoals. If the assignment makes the body true, add the tuple for the head to the result.

Variable-based Evaluation

Example:

Database:     Edge(from, to)

<table>
<thead>
<tr>
<th>from</th>
<th>to</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

Datalog rule: NoTranzitive(x, z) ← Edge(x, y) AND Edge(y, z) AND NOT Edge(x, z)

Assignment       x = 1, y = 2 , z = 3
Edge(1, 2) AND Edge(2, 3) AND NOT Edge(1, 3) is true, make (1, 3) a tuple of the answer
Assignment       x = 1, y = 2 , z = 2
Edge(1, 2) AND Edge(2, 2) AND NOT Edge(1, 2) makes the body true
Assignment       x = 2, y = 3, z = 2
Edge(2, 3) AND Edge(3, 2) AND NOT Edge(2, 2) no z makes the body true
Assignment       x = 3, y = 3, z = 2
Edge(3, 3) AND Edge(3, 3) AND NOT Edge(3, 3) no z makes the body true

Note: No other assignment for x and y makes Edge(x, y) true. Stop searching.

Tuple-based Evaluation

Example:

Database:     Edge(from, to)

<table>
<thead>
<tr>
<th>from</th>
<th>to</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Datalog rule: NoTranzitive(x, z) ← Edge(x, y) AND Edge(y, z) AND NOT Edge(x, z)

Assignment       (x, y) = (1, 2), (y, z) = (2, 3), consistent assignment
Edge(1, 2) AND Edge(2, 3) AND NOT Edge(1, 3) is true, make (1, 3) a tuple of the answer
Assignment       (x, y) = (1, 2), (y, z) = (1, 2), inconsistent assignment
Assignment       (x, y) = (2, 3), (y, z) = (1, 2), inconsistent assignment
Assignment       (x, y) = (2, 3), (y, z) = (2, 3), inconsistent assignment

Note: No other assignment for (x, y) makes Edge(x, y) true. Stop searching.

Datalog Programs

A Datalog Program is a collection of rules

Example:

"Find actors who starred in the color movies made in the 1950"

MoviesColor50 (t,y)← Movie(t,y,l,c,s,p) AND y = "1950" AND c = "y"
Answer(star) ← Movies90 (t,y) AND Contracts(star, studio, t, y, salary)

Datalog Programs Evaluation

Non-recursive programs:
- pick an order to evaluate the rules (the IDB predicates) so that all the predicates in the body have already been evaluated.
- if an IDB predicate has more than one rule, each contributes tuples to its relation (union).
From Relational Algebra to Datalog -1

Intersection: \( R(x, y) \cap T(x, y) \)
\( I(x, y) \leftarrow R(x, y) \text{ AND } T(x, y) \)

Union: \( R(x, y) \cup T(x, y) \)
\( U(x, y) \leftarrow R(x, y) \)
\( U(x, y) \leftarrow T(x, y) \)

Difference: \( R(x, y) - T(x, y) \)
\( D(x, y) \leftarrow R(x, y) \text{ AND NOT } T(x, y) \)

From Relational Algebra to Datalog -2

Projection: \( \pi_x(R) \)
\( P(x) \leftarrow R(x, y) \)

Selection: \( \sigma_{x>10}(R) \)
\( S(x, y) \leftarrow R(x, y) \text{ AND } x>10 \)

Product: \( R \times T \)
\( P(x, y, z, w) \leftarrow R(x, y) \text{ AND } T(z, w) \)

From Relational Algebra to Datalog -3

Natural Join \( R \bowtie T \)
\( J(x, y, z) \leftarrow R(x, y) \text{ AND } T(y, z) \)

Theta Join \( R \bowtie_{R.x > T.y} T \)
\( J(x, y, z, w) \leftarrow R(x, y) \text{ AND } T(z, w) \text{ AND } x > y \)

Datalog Queries

Datalog Query: a datalog program.

Expressive Power:
- without recursion, Datalog has the same power as Core Relational Algebra and Relational Calculus
- with recursion: much more, but not Turing-complete

Recursivity

Example:
Database: SequelOf(movie, sequel)

Query: "What are the sequels of sequels of movies in the database?"
\( \pi_{\text{first,second}}(\rho_{\text{first,second}}(\text{SequelOf})) \leftarrow \rho_{\text{first,second}}(\text{SequelOf}) \)

"What are the sequels of the sequels of the sequels?"

Infinite unions?

Recursive Rules

Example:
FollowOn(x, y) ← SequelOf(x, y)
FollowOn(x, y) ← SequelOf(x, z) AND FollowOn(z, y)

Dependency Graph (of a program)
- nodes: the IDB predicates
- edges: from node\(1(p\text{redicate}1)\) to node\(p\text{redicate}2\) if and only if there is a rule with predicate\(1\) in the head and predicate\(2\) in the body.

A datalog program is recursive iff its dependency graph has a cycle.
**Dependency graph**

Example:
- FirstSequelOf(x, y) ← SequelOf(x, y)
- FollowOn(x, y) ← FirstSequelOf(x, y)
- FollowOn(x, y) ← FirstSequelOf(x, z) AND FollowOn(z, y)

Cyclic graph → recursive datalog program

---

**Evaluating Recursive Rules without Negation**

**Naive algorithm**

1. Begin by assuming all IDB relations are empty
2. Repeatedly evaluate the rules using the EDB and the previous IDB to get a new IDB
3. End when there is no change to IDB

---

**Negation in recursive rules**

- Naive evaluation does not work when there are negated subgoals.
- Arguably negation wrapped in a recursion makes little or no sense in general
- Even when negation and recursion are separate there is ambiguity about the “correct” IDB relations

---

**Example:**

Database: SequelOf = \{[(1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1)]\}

<table>
<thead>
<tr>
<th>Round</th>
<th>FirstSequelOf</th>
<th>FollowOn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>{}</td>
<td>{}</td>
</tr>
<tr>
<td>Round1</td>
<td>FirstSequelOf := {[(1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1)]}, FollowOn := {}</td>
<td></td>
</tr>
<tr>
<td>Round2</td>
<td>FollowOn := {[(1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1)]}</td>
<td></td>
</tr>
<tr>
<td>Round3</td>
<td>FollowOn := {[(1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1)] U [(1,2), (2,2), (3,2), (4,2), (5,2), (6,2), (7,2)]}</td>
<td></td>
</tr>
<tr>
<td>Round4</td>
<td>FollowOn := {[(1,1), (2,1), (3,1), (4,1), (5,1), (6,1), (7,1)] U [(1,2), (2,2), (3,2), (4,2), (5,2), (6,2), (7,2)] U [(1,3), (2,3), (3,3), (4,3), (5,3), (6,3), (7,3)]}</td>
<td></td>
</tr>
<tr>
<td>Round 5</td>
<td>no change in FollowOn. STOP</td>
<td></td>
</tr>
</tbody>
</table>

---

**Example:**

EDB predicate R = \{\{(0)\}\}

P(x) ← R(x) AND NOT Q(x)
Q(x) ← R(x) AND NOT P(x)

2 solutions
P = \{0\}, Q = \{0\}

Which one to choose?

---

**Example:**

EDB predicate S = \{\{(1)\}\}

R(x) ← S(x) AND NOT R(x)

Initial R = \{\}\nRound 1 R = \{\{(1)\}\}
Round 2 R = \{\}\nRound 3 R = \{\{(1)\}\}, etc.
Stratified Negation

- Constraint imposed on recursive Datalog programs
- Rules out negation wrapped in recursion
- The maximum number of negations that can be applied to an IDB predicate used in evaluating an IDB predicate must be finite.

Stratum Graph

Labeled dependency graph

- nodes: the IDB predicates
- edges: from node1(predicate1) to node(predicate2) if and only if there is a rule with predicate1 in the head and predicate2 in the body. If predicate2 appears negated, label the edge with "-".

Strata

- The stratum of a node (predicate) is the maximum number of "-" labeled edges on a path leading from that node.
- A Datalog program is stratified if all its IDB predicates have finite strata.

Example:

```
R(x) ← S(x) AND NOT R(x)
- - -
P(x) ← R(x) AND NOT Q(x)
Q(x) ← R(x) AND NOT P(x)
```

Stratified Datalog Evaluation

Algorithm:

1. Evaluate IDB predicates lowest-stratum-first
2. Once evaluated, treat them as "EDB" for the IDB predicates with higher strata.

SQL Recursion

- Datalog recursion has inspired the introduction of recursion in the SQL-99 standard.
- More difficult: SQL allows grouping and aggregation requires a more complex notion of stratification
SQL Recursive Queries

Syntax

WITH

<Datalog-like rules>

<a core SQL query using the predicates in the rules >

• The keyword WITH

• One or more definitions, separated by commas, of the form:
  - the optional keyword RECURSIVE
  - the name of the relation being defined
  - the keyword AS
  - the query that defines the relation

• A query which may refer to any of the prior definitions, and forms the result of the WITH statement.

Example: "Find all Rocky's sequels"

WITH

FirstSequelOf(x, y) AS SELECT * FROM SequelOf;
RECURSIVE FollowOn(x, y) AS
(SELECT * FROM FirstSequelOf)
UNION
(SELECT FirstSequelOf.x, FollowOn.y
FROM FirstSequelOf, FollowOn
WHERE FirstSequelOf.y = FollowOn.x)

SELECT y FROM FollowOn WHERE x="Rocky"

Monotonicity

• If a relation P is a function of a relation Q, we say P is monotone in Q if inserting tuples into Q cannot cause any tuples to be deleted from P.

Example:
P = Q  UNION  R
P = SELECT * FROM Q

Nonmonotonicity

Example:

Let P be the result relation of the query SELECT AVG(x) FROM Q

P is not monotone in Q: inserting a new tuple in Q may change the average and thus delete the old average.

SQL Stratum Graph

Nodes - IDB relations declared in WITH clause
- Subqueries in the body of the rules (at any level of nesting)

Edges P→Q if:
- P is a rule head and Q is a relation in the FROM clause or an immediate subquery
- P is a subquery and Q is a relation in its FROM clause or an immediate subquery.

Label with "-" an edge if P is not monotone in Q
Stratified SQL

Stratified SQL = finite number of "."s on the paths of the stratum graph

Example:
FirstSequelOf = ...
FollowOn = (... FROM FirstSequelOf) Subquery S1
UNION Subquery S2
(... FROM FollowOn)

Nonmonotone Example

WITH
FirstSequelOf(x,y) AS SELECT * FROM SequelOf;
RECURSIVE FollowOn(x, y) AS Subquery S1
(SELECT * FROM FirstSequelOf)
EXCEPT Subquery S2
(SELECT FirstSequelOf.x, FollowOn.y FROM FirstSequelOf, FollowOn WHERE FirstSequelOf.y = FollowOn.x )
SELECT * FROM FollowOn

• Note: inserting a tuple into S2 can delete a tuple from Follow on

Not and Nonmonotonicity

• Not every NOT means that the query is not monotone.

Example:
SELECT * FROM Q is monotone in Q
SELECT * FROM Q WHERE NOT(Q.x >10) is also monotone in Q

Note: All selections are monotone

Example

WITH
FirstSequelOf(x,y) AS SELECT * FROM SequelOf;
RECURSIVE FollowOn(x, y) AS
(SELECT * FROM FirstSequelOf)
UNION
(SELECT FirstSequelOf.x, FollowOn.y FROM FirstSequelOf, FollowOn WHERE FirstSequelOf.y = FollowOn.x AND
NOT (FirstSequelOf.x = FollowOn.y) )
SELECT * FROM FollowOn