Datalog

- Inspired by the impedance mismatch in relational databases.
- Main expressive advantage: recursive queries.
- More convenient for analysis: papers look better.
- Without recursion but with negation it is equivalent in power to relational algebra
- ***Has*** affected real practice: (e.g., recursion in SQL3, magic sets transformations).

Datalog Concepts

- Atoms
- Datalog rules, datalog programs
- EDB predicates, IDB predicates
- Conjunctive queries
- Recursion
- Built-in predicates
- Negated atoms, stratified programs.
- Semantics: least fixpoint.

Predicates and Atoms

- Relations are represented by predicates
- Tuples are represented by atoms.

Purchase(“joe”, “bob”, “Nike Town”, “Nike Air”, 2/2/98)

- arithmetic, *built-in,* atoms:

X < 100, X+Y+5 > Z/2

- negated atoms:

NOT Product(“Linux OS”, $100, “Microsoft”)

Datalog Rules and Queries

A datalog rule has the following form:

head :- atom1, atom2, …., atom,…

Examples:

PerformingComp(name) :- Company(name,sp,c), sp > $50

AmericanProduct(prod) :-
Product(prod,pr,cat,mak), Company(mak, sp,“USA”)

All the variables in the head must appear in the body.

A single rule can express exactly select-from-where queries.

Datalog Terminology

- A datalog program is a set of datalog rules.
- A program with a single rule is a conjunctive query.

We distinguish EDB predicates and IDB predicates:

- EDB’s are stored in the database, appear only in the bodies
- IDB’s are intensionally defined, appear in both bodies and heads

The Meaning of Datalog Rules

AmericanProduct(prod) :-
Product(prod,pr,cat,mak), Company(mak, sp,“USA”)

Consider every assignment from the variables in the body to the constants in the database.

If each of the atoms in the body is in the database, then
the tuple for the head is in the relation of the head.

More Examples

CREATE VIEW Seattle-view AS
SELECT buyer, seller, product, store
FROM Person, Purchase
WHERE Person.city = "Seattle" AND
Person.per-name = Purchase.buyer

SeattleView(buyer,seller,product,store) :-
Person(buyer, "Seattle", phone),
Purchase(buyer, seller, product, store).

More Examples (negation, union)

SeattleView(buyer,seller,product,store) :-
Person(buyer, "Seattle", phone),
Purchase(buyer, seller, product, store)
not Purchase(buyer, seller, product, "The Bon")

Q5(buyer) :- Purchase(buyer, "Joe", prod, store)
Q5(buyer) :- Purchase(buyer, seller, store, prod),
Product(prod, price, cat, maker)
Company(maker, sp, country),
sp > 50.

Defining Views

SeattleView(buyer,seller,product,store) :-
Person(buyer, "Seattle", phone),
Purchase(buyer, seller, product, store)
not Purchase(buyer, seller, product, "The Bon")

Q6(buyer) :- SeattleView(buyer, "Joe", prod, store)
Q6(buyer) :- SeattleView(buyer, seller, store, prod),
Product(prod, price, cat, maker)
Company(maker, sp, country),
sp > 50.

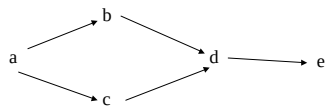
Meaning of Datalog Programs

- *Start with the facts in the EDB and iteratively derive facts for IDBs.*
  - Repeat the following until you cannot derive any new facts:
- Consider every assignment from the variables in the body to the constants in the database.
- If each of the atoms in the body is made true by the assignment, then,
- add the tuple for the head into the relation of the head.

Transitive Closure

Suppose we are representing a graph by a relation *Edge(X,Y):*

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



I want to express the query:

Find all nodes reachable from a.

Recursion in Datalog

Path(X, Y) :- Edge(X, Y)
Path(X, Y) :- Path(X, Z), Path(Z, Y).

Semantics: evaluate the rules until a fixedpoint:

Iteration #0: Edge: {(a,b), (a,c), (b,d), (c,d), (d,e)}
Path: {}
Iteration #1: Path: {(a,b), (a,c), (b,d), (c,d), (d,e)}

Iteration #2: Path gets the new tuples:
(a,d), (b,e), (c,e)
Iteration #3: Path gets the new tuple:
(a,e)
Iteration #4: Nothing changes -> We stop.
Note: number of iterations depends on the data. Cannot be anticipated by only looking at the query!

Model-Theoretic Semantics

- An interpretation is an assignment of extensions to the EDB and IDB rules.
- An interpretation of a DB is a model for it whenever it satisfies the rules:
  - I.e., if you apply the rules, you get nothing new.
- The least fixpoint model of a datalog program is also the intersection of all of its models (for a given EDB extension).

Built in Predicates

Rules may include atoms with built-in predicates:

ExpensiveProduct(X) :- Product(X,Y,P) & P > $100

But: we need to restrict the use of built-in atoms in rules.

P(X) :- R(X) & X<Y

What does this mean?

We could use active domain semantics, but that's problematic.

Hence, we require that every variable that appears in a built-in atom also appears in a relational atom.

Negated Subgoals

Rules may include negated subgoals, but in restricted forms:

P(X,Y) :- Between(X,Y,Z) & NOT Direct(X,Z)

Bad:

P(X, Y) :- R(X) & NOT S(Y)

Bad but ok:

P(X) :- R(X) & NOT S(X,Y)

We'll rewrite as:
S'(X) :- S(X,Y)
P(X) :- R(X) & NOT S'(X)

Stratified Rules

A predicate P depends – (+) on a predicate Q if:
Q appears negated (positive) in a rule defining P.

If there is a cycle in the dependency graph that involves a - edge, the datalog program is not stratified.

Example:

p(X) :- r(X) & NOT q(X)
q(X) :- r(X) & NOT p(X)

Suppose r has the tuple {1}.

Subtleties with Stratified Rules

Example:

p(X) :- r(X)
q(X) :- s(X) & NOT p(X).

Suppose: R = {1}, and S = {1,2}

One solution: P = {1} and Q = {2}

Another solution: P={1,2} and Q={}.

Perfect model semantics: apply the rules stratum after stratum.

Deductive Databases

- General idea: some relations are stored (extensional), others are defined by datalog queries (intensional).
- Many research projects (MCC, Stanford, Wisconsin) [Great Ph.D theses!]
- SQL3 realized that recursion is useful, and added linear recursion.
- Hard problem: optimizing datalog performance.
- Ideas from deductive databases made it into the mainstream.