Constraint Databases

Problem with logical models

How would you solve problems such as:

- Is there a point in the lake from which the whole shoreline can be seen?
- Is there a placement of cell stations that covers all areas?

What is the problem?

Finite/Infinite

Why can't we just write

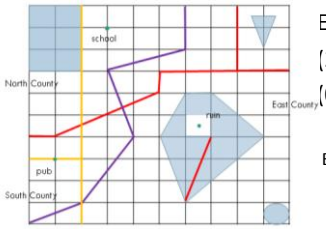
```
select R.x, R.y
from PlaneObject PO, Rect R
where PO.x = R.x
 and  PO.y = R.y;
```

What does this query do?

Allowing Infinite Relations

- not restricted to particular geometric shape, geometry is implicit
- geometric transformations/operations expressed logically
- structure of relational language remains the same
- how can such geometries be represented?

Implicit Representation



East County:

$(2 \le x \wedge x \le 10) \wedge$

$(0 \le y \wedge y \le 10)$

Examples
- pub
- ferry
- circle lake
- buoys
- purple road

Implicit Representation

All standard geometric objects can be written
- as logical formulas using Λ (and), v (or) and
- atomic conditions that are polynomial equalities/inequalities
- logical formula can be assumed to be in DNF
  (Disjunctive Normal Form: an Or of Ands)
  convex objects: written as conjunctions (ands)
  non-convex objects: written as unions (ors) of convex objects

Example: purple road/polygon lake

Evaluation

```
select S.x, S.y
from MyCounty C, MyStreet S
where C.x = S.x
 and  C.y = S.y;
```

What does this look like for North County and Red Street?

Levels

- abstract relational level

```
select x,y
from MyCounty
where count_name = 'East County';
```

- symbolic level

  $(2 \leq x \wedge x \leq 10) \wedge (0 \leq y \wedge y \leq 10)$

- physical level

Constraint Data Models

- **linear** constraints
  - atomic constraints involve + and multiplication by constants only
  - i.e. points, lines, halfplanes and combinations of these
  - Semilinear sets
- **polynomial** constraints
  - atomic constraints use + and *
  - points, lines, curves, circles, …
  - Semialgebraic sets

Linear Constraints

- atomic conditions $\sum_i s_i x_i \theta s$
  - x_i is variable
  - Θ is =, ≤, < (for > etc. multiply by -1)
- formula in DNF:
  And of Ors of atomic conditions
- linear constraint relation
  - if it can be described by such a formula
  - always is the union of convex objects

Two Issues to keep in mind

- Expressiveness
  - What is expressible, what isn't?
  - Closure: is our model closed under the basic operations, e.g. selection, projection, union, intersection, difference, join?
- Computability and Complexity
  - Can we evaluate the queries at all?
  - Can we evaluate the queries effectively?

Query Language based on Relational Calculus

- First-order queries
  - Add quantifiers: there is $\exists$
    for all $\forall$
  - Express:
    - purple street and red street have a crossing
    - the ferry lies in poly lake
    - there is a point in east county which is west of polygon lake
    - Polygon lake lies entirely north of circle lake
    - Boundary of triangle lake, interior of triangle lake
    - Give examples of queries we cannot express
  - How do we deal with quantifiers?

Quantifier Elimination

- For every quantifier formula, there is a quantifier-free formula expressing the same set.
  - For polynomial constraints exponential blow-up in formula-size
  - Non-trivial algorithm

Query Language based on Relational Algebra

- Set operations:
  - union, intersection, difference
  - Cartesian product, join
  - Selection, projection
  - Express:
    - Parts of purple street belonging to South County
    - Parts of purple street south of Rectangle Lake
- What's the difficult operation here?

Projection

Projection is difficult

corresponds to existential quantifier

Examples:

Project triangle lake to its x coordinate.

Project poly lake lake to its x coordinate.

Prototype Systems

- DEDALE
  - http://gemo.futurs.inria.fr/dedale/
- MLPQ
  - http://www.cse.unl.edu/~revesz/MLPQ/mlpq.htm

MLPQ

Developed by Peter Revesz (University of Nebraska-Lincoln)

Overview (includes executable system)

- http://www.cse.unl.edu/~revesz/MLPQ/mlpq.htm

Detailed instructions and sample databases

- http://cse.unl.edu/~revesz/MLPQ/Instruction.pdf
- http://cse.unl.edu/~revesz/MLPQ/database.zip

Datalog

- deductive database language
- subset of Prolog
- based on rules

  A :- B, C, …

  where A, B, C are relations (involving variables).

Example (MLPQ using Datalog): class prereqs

```
class(id, name) :- id = 1, name = "CSC 421".
class(id, name) :- id = 2, name = "CSC 383".
class(id, name) :- id = 3, name = "CSC 212".
class(id, name) :- id = 4, name = "CSC 211".
class(id, name) :- id = 5, name = "CSC 202".

requires(rqid,rdid) :- rqid = 1, rdid = 2.
requires(rqid,rdid) :- rqid = 1, rdid = 5.
requires(rqid,rdid) :- rqid = 2, rdid = 3.
requires(rqid,rdid) :- rqid = 3, rdid = 4.
```

Exercise:

- Write rule that lists class and prereq by name.
- Write rule that lists all classes required (implicitly or explicitly) by a class.

Examples (MLPQ)

```
begin %gallery%
painter(id, name, phone) :- id=123, name="Rose", phone="888-4567".
painter(id, name, phone) :- id=126, name="Itero", phone="345-1122".
painter(id, name, phone) :- id=234, name="Picasso", phone="456-3345".
painter(id, name, phone) :- id=335, name="OKeefe", phone="567-8999".
painter(id, name, phone) :- id=456, name="Miro", phone="777-7777".

painting(pnum, title, price, id) :- pnum=2345, title="Wild Waters", price=245, id=126.
painting(pnum, title, price, id) :- pnum=4536, title="Sea Storm", price=8359, id=335.
painting(pnum, title, price, id) :- pnum=6666, title="Wild Waters", price=6799, id=234.
painting(pnum, title, price, id) :- pnum=7878, title="High Tide", price=98000, id=456.
painting(pnum, title, price, id) :- pnum=6789, title="Paradise", price=590000, id=234.
painting(pnum, title, price, id) :- pnum=7896, title="Faded Rose", price=145, id=123.
painting(pnum, title, price, id) :- pnum=9889, title="Sunset", price=975000, id=234.

gallery(pnum, owner) :- pnum=2345, owner="Johnson".
gallery(pnum, owner) :- pnum=6666, owner="Johnson".
gallery(pnum, owner) :- pnum=4536, owner="McCloud".
gallery(pnum, owner) :- pnum=7878, owner="McCloud".
gallery(pnum, owner) :- pnum=6789, owner="Palmer".
gallery(pnum, owner) :- pnum=7896, owner="Palmer".
gallery(pnum, owner) :- pnum=9889, owner="Palmer".
end %gallery%
```

Sample queries (gallery)

- List titles of paintings for less than $10,000.
- Find average price of paintings by Picasso.
- List paintings by Miro and Picasso.
- Find title and price of most expensive painting.

Spatial Examples (My World)

Warning: Spatial data tuples cannot contain non-numeric info (solution: store in separate table)

```
begin%my_world%
poi_g(id,x,y):- id = 1, x = 6.5, y = 4.5.
poi_g(id,x,y):- id = 2, x = 1, y = 3.

poi_i(id,name):- id = 1, name = "ruin".
poi_i(id,name):- id = 2, name = "pub".
....
```

Exercises:

- list pois and counties they belong to
- what counties does purple street pass through?
- list pois and lakes they are "close" to
- find eastern-most points of purple street
- which part of purple street lies in south county

Spatial/Temporal Examples (regions)

```
begin%Test%
country(id,x,y,t):- id = 1, x >= 0, x <= 4, y >=5 , y <= 15, t >=1800 , t <=1950.
country(id,x,y,t):- id = 1, x >= 0, x <= 8, y >=5, y <=15, t >=1950 , t <= 2000.
country(id,x,y,t):- id = 2, x >= 4, x <= 12, y >=5 , y <=15 , t >= 1800, t <=1950 .
country(id,x,y,t):- id = 2, x >= 8, x <= 12, y >=5 , y <=15 , t >= 1950, t <= 2000.
country(id,x,y,t):- id = 3, x >= 0, x <= 12, y >=0 , y <=5 , t >= 1800, t <= 2000.
location(c,x,y):- x = 3, y = 2, c = 101.
location(c,x,y):- x = 7, y = 3, c = 102.
location(c,x,y):- x = 5, y = 6, c = 103.
location(c,x,y):- x = 7, y = 10, c = 104.
location(c,x,y):- x = 10, y = 8, c = 105.
location(c,x,y):- x = 1, y = 7, c = 106.
location(c,x,y):- x = -8, y = 6, c = 107.
growth(t,c,p):- c = 101, p = 10000 , t >=1800 , t <= 2000.
growth(t,c,p):- c = 102, p = 20000 , t >=1800 , t <= 2000.
growth(t,c,p):- c = 103, p = 10000 , t >=1800 , t <= 2000.
growth(t,c,p):- c = 104, p = 30000 , t >=1800 , t <= 2000.
growth(t,c,p):- c = 105, p = 40000 , t >=1800 , t <= 2000.
growth(t,c,p):- c = 106, p = 35000 , t >=1800 , t <= 2000.
end%Test%
```

Exercises:

- create a relation for country with id = 1,
- can animate
- find cities that belonged to country 2 before they belonged to country 1
- find cities that in 1998 had a population of more than 20,000 and did not belong to country 2.

Network/Travel (go)

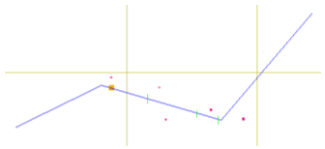
```
go(city1, t1, city2, t2):- city1=1, city2=2, t2-t1>=60.
go(city1, t1, city2, t2):- city1=1, city2=3, t2-t1>=650.
go(city1, t1, city2, t2):- city1=1, city2=4, t2-t1>=400.
go(city1, t1, city2, t2):- city1=3, city2=5, t2-t1>=370.
...

city(id, name):- id=1, name="Omaha".
city(id, name):- id=2, name="Lincoln".
city(id, name):- id=3, name="Minneapolis".
city(id, name):- id=4, name="Des_Moines".
city(id, name):- id=5, name="Chicago".
city(id, name):- id=6, name="Kansas_City".
```

Exercise: write a relation that calculates minimal travel time between cities.

Using buffer

Find hotel within 50 miles of nearest exit to the east.



Current_Pos(Name, x,y) :- Name=1, x>= 172, x<=180, y>=172, y <= 180.
Road(80, x, y):- -65x + 132y = 13295, x >= 29, x <= 161. *(…)*
Exit(409, x, y):- x = 231, y>= 152.344, y<=165.344.
Hotel(1, x, y):- x >= 174, x<=177, y >= 191.5, y <= 192.5.

Other operations

- Set operations (complement)
- Functions (max, min)
- Drawing (line, polygon, rectangle)

Exercise:
create poly lake in my_world