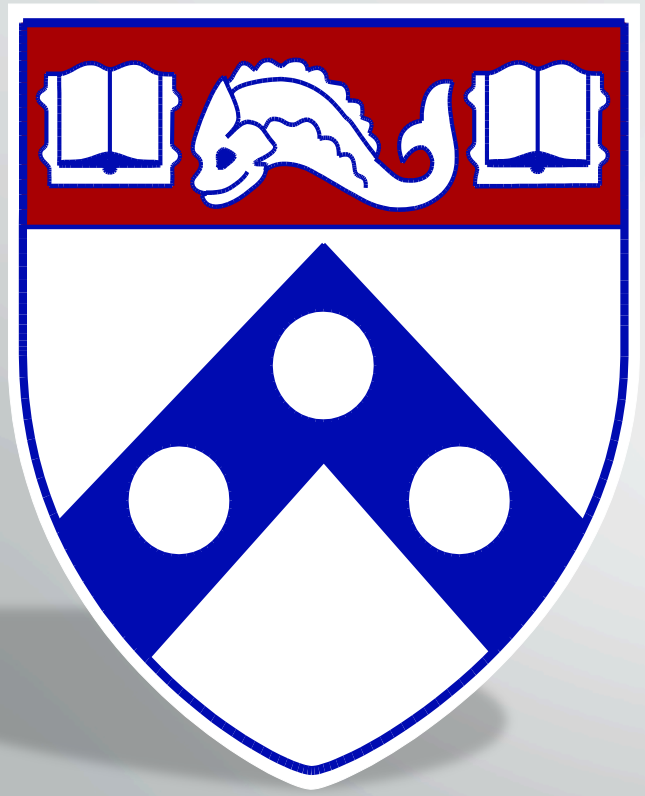


Datalog and Negation

Susan B. Davidson

CIS 700: Advanced Topics in Databases

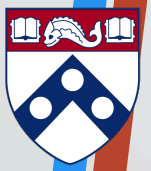
MW 1:30-3

Towne 309

http://www.cis.upenn.edu/~susan/cis700/homepage.html

Homework for this week

- Sign up to present a paper (the Google doc link was sent on Friday)
- Class schedule is being updated based on this.
- Think about a course project, and come speak with me about it. You should send me a written proposal by Feb. 9.



Last time: Datalog$^+$ with recursion

- Evaluating non-recursive IDB predicates: ordering
- Naïve and semi-naïve evaluation of recursive predicates
- Negation can be tricky…

Example: Recursion and Negation

EDB:

red(X,Y)= There is a red bus from X to Y

green(X,Y)= There is a green bus from X to Y

IDB:

greenPath(X,Y)= You can get from *X* to *Y* using only green buses.

monopoly(X,Y)= Red has a bus from *X* to *Y*, but you can't get there on Green, even changing buses.

Example: Recursion and Negation

EDB:

red(X,Y)= There is a red bus from X to Y

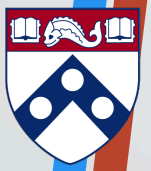
green(X,Y)= There is a green bus from X to Y

IDB:

greenPath(X,Y) :- green(X,Y)

greenPath(X,Y) :- greenPath(X,Z) , greenPath(Z,Y)

monopoly(X,Y) :- red(X,Y) , NOT greenPath(X,Y)



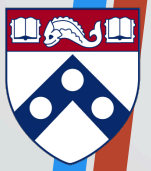
Two Minimal Models

1. EDB + greenPath(1,2) + monopoly(2,3)
2. EDB + greenPath(1,2) + greenPath(2,3) + greenPath(1,3)

greenPath(X,Y) :- green(X,Y)

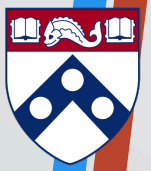
greenPath(X,Y) :- greenPath(X,Z), greenPath(Z,Y)

monopoly(X,Y) :- red(X,Y), NOT greenPath(X,Y)



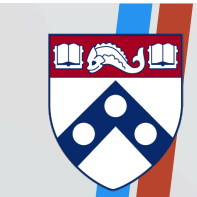
Stratified Models

1. *Dependency graph* describes how IDB predicates depend negatively on each other.
2. *Stratified Datalog* = no recursion involving negation.
3. *Stratified model* is a particular model that "makes sense" for stratified Datalog programs.

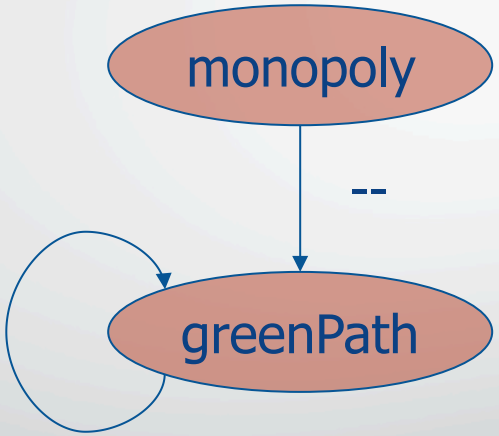


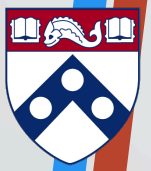
Dependency Graph

- Nodes = IDB predicates.
- Arc $p \rightarrow q$ iff there is a rule for p that has a subgoal with predicate q.
- Arc $p \rightarrow q$ labeled "–" iff there is a subgoal with predicate q that is negated.



Monopoly Example

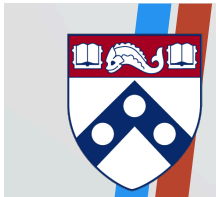




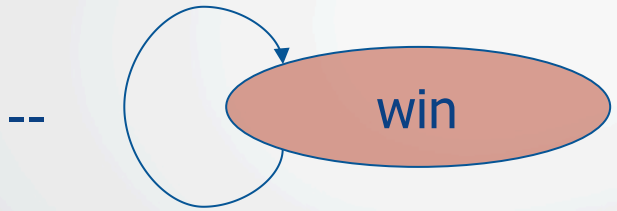
Another Example: “Win”

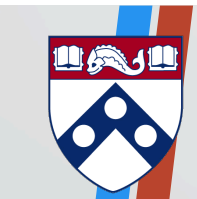
win(X) :- move(X,Y), NOT win(Y)

- Represents games where you win by forcing your opponent to a position where they have no move.



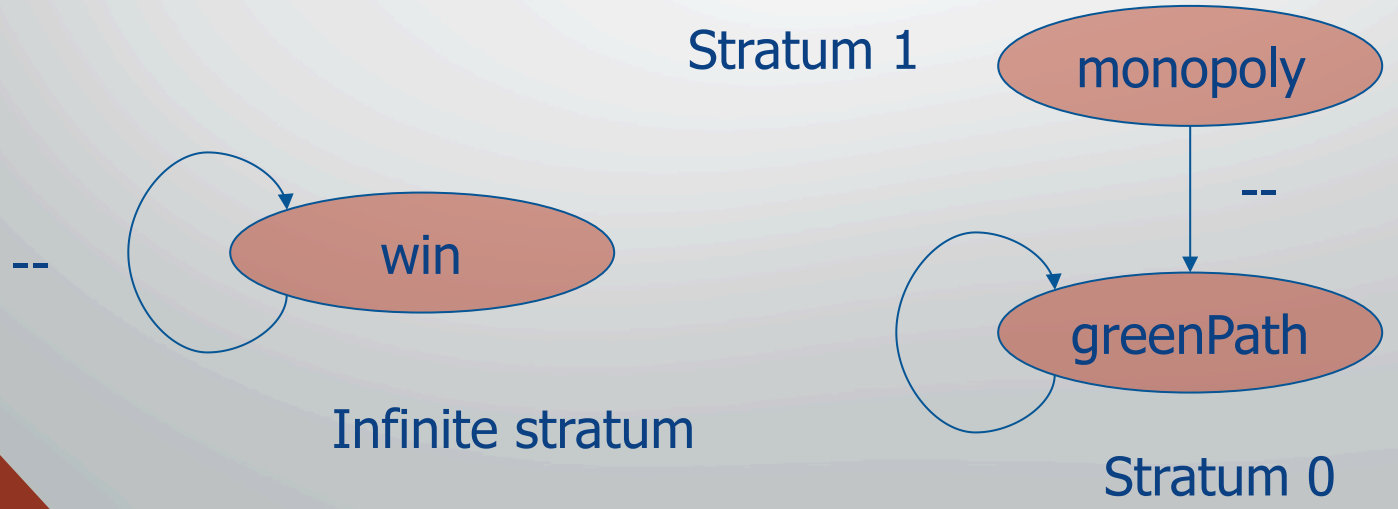
Dependency Graph for “Win”

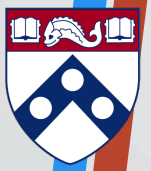




Strata

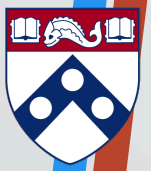
- The stratum of an IDB predicate is the largest number of –'s on a path from that predicate, in the dependency graph.
- Examples:





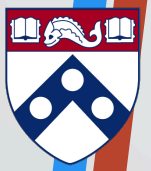
Stratified Programs

- If all IDB predicates have finite strata, then the Datalog program is s*tratified*.
- If any IDB predicate has an infinite stratum, then the program is *unstratified*, and no stratified model exists.



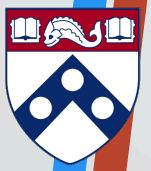
Stratified Model

- Evaluate strata 0, 1,… in order.
- If the program is stratified, then any negated IDB subgoal has already had its relation evaluated.
  - Safety assures that we can “subtract it from something.”
  - Treat it as EDB.
- Result is the stratified model.



Examples

- For "Monopoly," greenPath is in stratum 0: compute it (the transitive closure of green).
- Then, monopoly is in stratum 1: compute it by taking the difference of red and greenPath.
- Result is first model proposed.
- "Win" is not stratified, thus no stratified model.



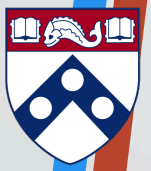
Putting it all together…

p(x,y):- q(x,y), NOT r(x,y)
r(x):- s(x,y), not t(y)
r(x):- s(x,y), r(y)

EDB predicates are q, t, s.

Q={ab, bc, cd, de}
S={ab, bc, ac}
T={a, b, c}

- What are the strata of the IDB predicates p and r?
- What is the translation of the rules for p and r into relational algebra?



Summary

- Datalog captures the core of many query languages:
  - Unions of conjunctive queries
  - Recursion
  - Negation
- The syntax for conjunctive queries is used for a number of different practical problems
  - Query optimization
  - Data integration
  - Data citation
  - …