Extensions of Datalog

Wednesday, February 13, 2001

1

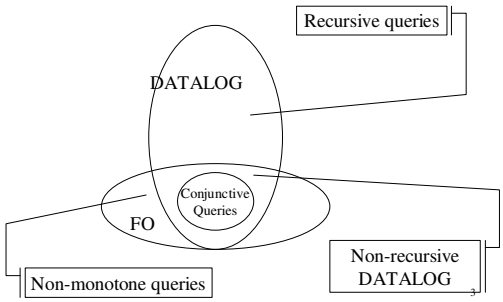
Outline

- Non-recursive Datalog with negation
- Datalog with negation
  - Stratified Datalog¬
  - Inflationary Datalog¬
  - Partial Datalog¬
- Query languages and complexity classes

[AHV] Chapters 14, 15, 17

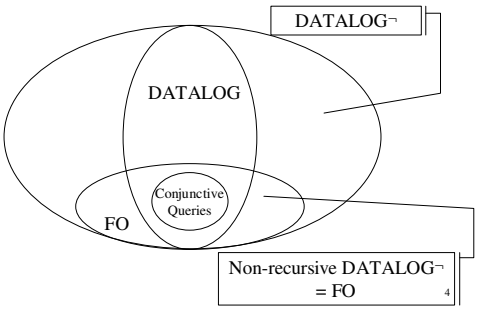
2

Picture So Far



3

Goal Today



4

Datalog¬

- A datalog¬ **rule** is:

$$R_0(\bar{x}_0) :- R_1(\bar{x}_1), ..., R_k(\bar{x}_k)$$

head body

- Where:
  - R_0 is an IDB relation
  - R_1, ..., R_k are EDB and/or IDB relations, possibly negated !

5

Example

Employee(x), ManagedBy(x,y), Manager(y)

- Find all employees that report to John or to Dave:

  Answer(x) :- ManagedBy(x,"John")

  Answer(x) :- ManagedBy(x,"Dave")
- FO:

$$Aux(x) = ManagedBy(x,"John") \vee ManagedBy(x,"Dave")$$

6

Example

Employee(x), ManagedBy(x,y), Manager(y)

- Find all employees that are not managers:

  Answer(x) :- Employee(x), ¬Manager(x)

7

Example

Employee(x), ManagedBy(x,y), Manager(y)

- Find all employees that are not managed by Smith:

  Answer(x) :- Employee(x), ¬ManagedBy(x, "Smith")

8

Example

Employee(x), ManagedBy(x,y), Manager(y)

- Find all employees without a manager:

  Answer(x) :- Employee(x), ¬ManagedBy(x,y)

- WRONG ! How is y quantified ?

9

Example

Employee(x), ManagedBy(x,y), Manager(y)

- Find all employees without a manager:

  Aux(x) :- ManagedBy(x,y)
  Answer(x) :- Employee(x), ¬Aux(x)

- FO:

$$Answer(x) = Employee(x) \wedge \neg(\exists y.ManagedBy(x, y))$$

10

Example

Employee(x), ManagedBy(x,y), Manager(y)

- Find the manager of all employees

  Aux(y) :- Employee(x), Manager(y), ¬ManagedBy(x,y)
  Answer(y) :- Manager(y), ¬Aux(y)

- FO:

$$Answer(y) = Manager(y) \wedge \forall x.(Employee(x) \Rightarrow ManagedBy(x, y))$$

11

Datalog¬

Safe Datalog¬ rules:

- Every variable in the head occurs in the body
- Every variable in the body occurs in a positive literal

E.g. of unsafe rules:

A(x,y) :- R(x,z), ¬R(z,y)
A(x) :- R(x,y), ¬R(z,y)

12

Problems with Recursion and Negation

A1(x) :- R(x), ¬A2(x)

A2(x) :- R(x), ¬A1(x)

- This program has no minimal model. E.g. assuming R(10):
  - Model 1: A1={10}, A2=∅
  - Model 2: A1=∅, A2={10}

13

Fixes to Datalog¬

- Non-recursive Datalog¬:
  - Simple semantics
- Recursive Datalog¬:
  - Several fixes are possible, none is elegant

14

Non-recursive Datalog¬

- Semantics: "compute" the IDB relations in the order in which they are defined
- ***Theorem***. Non-recursive Datalog¬ can express precisely the same queries as FO
- Datalog¬ has nicer syntax (no quantifiers) than FO
- Important difference: Datalog¬ is much more concise than FO ! (next)

15

Non-recursive Datalog(¬)

- A concise non-recursive Datalog program:
  P2(x,y) :- R(x,y)
  P2(x,y) :- R(x,z), R(z,y)
  P4(x,y) :- P2(x,z), P2(z,y)
  P8(x,y) :- P4(x,z), P4(z,y)
  Answer(x,y) :- P8(x,z), P8(z,y)
- Looks for paths of length ≤ 16
- Equivalent FO formula (after simplifications !) has 16 disjuncts, each with 1, 2, ..., 16 conjuncts respectively

16

Non-recursive Datalog(¬)

Fact. Unfolding non-recursive Datalog or Datalog¬ programs may result in exponentially larger FO formulas

17

Containment of non-recursive Datalog Queries

Theorem Containment of unions of conjunctive queries is NP-complete

Idea:

$$q = c_1 \vee c_2 \vee ... \vee c_m,\ q' = c'_1 \vee c'_2 \vee ... \vee c'_n$$

$$q \subseteq q' \quad \Leftrightarrow \quad \forall i = 1,...,m\ \exists j = 1,...,n\ \ c_i \subseteq c'_j$$

Corollary Containment of non-recursive datalog queries is decidable

BUT in exponential time !

18

Recursion and Negation

- It's OK to negate the EDB predicates; problems occur when we negate IDB predicates
- Are there any useful instances ?
- Example: graph V(x), R(x,y), find all nodes that are not accessible from "a":

  T(x) :- R("a",x)

  T(x) :- T(y), R(y,x)

  Answer(x) :- V(x), ¬T(x)
- How do we define its meaning ?

19

Solution 1: Stratified Datalog¬

- Require that the rules of a program be grouped in *strata*
- Each stratum may use negation only over the IDB predicates defined in previous strata
- Semantics: compute strata successively
- This is the same idea as in non-recursive Datalog¬

20

Solution 1: Stratified Datalog¬

- Example:

  T(x) :- R("a",x)

  T(x) :- T(y), R(y,x)

  Answer(x) :- V(x), ¬T(x)
- Example:

  A1(x) :- R(x), ¬A2(x)

  A2(x) :- R(x), ¬A1(x)

  no stratification is possible

21

Solution 1: Stratified Datalog¬

Advantage:

- Natural definition
- Semantics can be defined in terms of a *stable* model (generalizes minimal model).

Disadvantage:

- Some "real" queries are not expressible as stratified programs

22

Solution 2: Inflationary Datalog¬

- Always add new facts to the IDB's, stop when no more facts can be added
- Example:

  A1(x) :- R(x), ¬A2(x)

  A2(x) :- R(x), ¬A1(x)

  Assuming R(10), the answers are:

  A1(10), A2(10)

23

Solution 2: Inflationary Datalog¬

- Example:

  T(x) :- R("a",x)

  T(x) :- T(y), R(y,x)

  Answer(x) :- V(x), ¬T(x)
- During first step, all nodes V(x) are inserted into Answer: this is not what we want
- We rewrite this query to have our intended meaning under inflationary semantics

24

Solution 2: Inflationary Datalog¬

T(x) :- R("a",x)
T(x) :- T(y), R(y,x)
oldT(x) :- T(x)
oldTbutLast(x) :- T(x), T(y), R(y,x'), ¬T(x')
Answer(x) :- V(x), ¬T(x), oldT(x'), ¬oldTbutLast(x')

- Need a PhD in databases to understand it

Theorem. Every stratified Datalog¬ program can be translated into an inflationary Datalog¬ program.

25

Solution 2: Inflationary Datalog¬

Advantage:

- More expressive

Disadvantage:

- Ad-hoc, procedural semantics
- Some queries are hard to read

26

Solution 3: Partial Datalog¬

- Compute the fixpoint until it converges
- Example:

  T(x) :- R("a",x)
  T(x) :- T(y), R(y,x)
  Answer(x) :- V(x), ¬T(x)

  Answer will have wrong answer initially, then they are deleted
- Example:

  A1(x) :- R(x), ¬A2(x)
  A2(x) :- R(x), ¬A1(x)

  doesn't converge

27

Solution 3: Partial Datalog¬

Theorem Every inflationary Datalog¬ program can be translated into a partial Datalog¬ program

Idea: just add the rule T(x) :- T(x) for every IDB relation T

28

Data Complexity

Theorem The data complexity of:

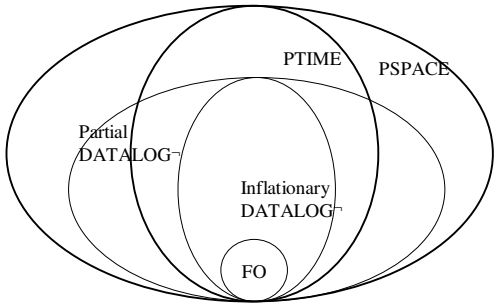
- Datalog
- Stratified Datalog¬
- Inflationary Datalog¬

is PTIME.

Theorem The data complexity of partial Datalog¬ is PSPACE.

29

Global Picture



30

Query Languages and Complexity Classes

- Datalog¬ $\subset$ PTIME
- Q: What is in PTIME but not in Datalog¬ ?

- A: Parity. Given R(x),
  - Answer = {x | R(x)} if |R| is even
  - Answer = {} if |R| is odd

Theorem Parity is not expressible in partial Datalog¬ (hence not in inflationary Datalog¬ either)

31

Ordered Databases

- An *ordered* database is **D** = (D, R_1, ..., R_k, <) where < is a total order on D

Theorem [Immerman, Vardi]
  - on ordered databases, inflationary Datalog¬ = PTIME
  - on ordered databases, partial Datalog¬ = PSPACE

- Beautiful and celebrated results.
  - Characterize complexity classes without referring to computation cost

32