Lecture 11: Datalog

Tuesday, February 6, 2001

Outline

- Datalog syntax
- Examples
- Semantics:
  - Minimal model
  - Least fixpoint
  - They are equivalent ☺
- Naive evaluation algorithm
- Data complexity

[AHV] chapters 12, 13

Motivation

- **Theorem**. The **transitive closure** query is not expressible in FO:
  - q(G) = {(x,y) | there exists a path from x to y in G}
- TC is called a *recursive query*.
- Datalog extends FO with *fixpoints* (or *recursion*) enabling us to express recursive queries
- Datalog also offers a more user-friendly syntax than FO

Datalog

- Let $R_1, R_2, ..., R_k$ be a database schema
  - They define the *extensional database,* EDB
  - *EDB relations*
- Let $R_{k+1}, ..., R_{k+p}$ be additional relational names
  - They define the *intensional database,* IDB
  - *IDB relations*

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

Datalog

- A **datalog program** is a collection of rules
- Example: transitive closure.

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

- R = EDB relation, T = IDB relation

Examples in Datalog

- Transitive closure version 2:

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

Examples in Datalog

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

- Find all employees reporting directly to "Smith"

```
Answer(x) :- ManagedBy(x, "Smith")
```

Examples in Datalog

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

- Find all employees reporting directly or indirectly to "Smith"

```
Answer(x) :- ManagedBy(x, "Smith")
Answer(x) :- ManagedBy(x,y), Answer(y)
```

- This is the *reachability problem:* closely related to TC

Examples in Datalog

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

- We say that (x, y) are on the *same level* if x, y have the same manager, or if their managers are on the same level.

Examples in Datalog

- Find all employees on the same level as Smith:

```
T(x,y) :- ManagedBy(x,z), ManagedBy(y,z)
T(x,y) :- ManagedBy(x,u), ManagedBy(y,v),T(u,v)
Answer(x) :- T(x, "Smith")
```

- Called the *same generation problem*
- Also related to TC

Examples in Datalog

- Representing boolean expression trees:
  - Leaf1(x), AND(x, y1, y2), OR(x, y1, y2), Root(x)
- Find out if the tree value is 0 or 1

```
One(x) :- Leaf1(x)
One(x) :- AND(x, y1, y2), One(y1), One(y2)
One(x) :- OR(x, y1, y2), One(y1)
One(x) :- OR(x, y1, y2), One(y2)
Answer() :- Root(x), One(x)
```

Examples in Datalog

- **Exercise**: extend boolean expresions with NOT(x,y) and Leaf0(x); write a datalog program to compute the value of the expression tree.
- **Note**: you need Leaf0 here. Prove that without Leaf0 no datalog program can compute the value of the expresssion tree.

Discussion of Datalog So Far

- Any connections to Prolog ?
  - It is *exactly* prolog, with two changes:
    - There are no functions
    - The standard evaluation is bottom up, not top down
- Any connections to First Order Logic ?
  - Can express some queries that are not in FO
    - Transitive closure, accessibility, same generation, etc
    - But can only express *monotone* queries, e.g. we cannot say "find all employees that are not managers" (will fix this later).

Meaning of a Datalog Rule

- The rule T(x,z) :- R(x,y), T(y,z) means:
  - "when (x,y) is in R and (y,z) is in T then insert (x,z) in T"
- Formally, we associate to each rule r a formula φ_r:

$$\varphi_r = \forall x.\forall y\forall z.T(x,z) \Leftarrow (R(x,y) \wedge T(y,z))$$

- Rules of thumb:
  - Comma means AND
  - All variables are universally quantified
  - The :- sign means $\Leftarrow$

Meaning of Datalog Rule

- What about this:
  T(x,y) :- Manager(x) infinitely many y's !
- A rule is **safe** if all variables in the head occur in the body
- A safe rule can be rewritten:

$$\varphi_r = T(x,z) \Leftarrow \exists y.(R(x,y) \wedge T(y,z))$$

- Rule of thumb:
  - extra variables in the body are, in fact, existentially quantified

Meaning of Datalog Program

- Given a datalog program **P**

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

  T(x,z) :- R(x,y), T(y,z)
- We associate a FO formula Φ_P

$$\Phi_P = \forall x\forall y.(T(x,y) \Leftarrow R(x,y)) \wedge$$
$$\forall x\forall z.(T(x,z) \Leftarrow \exists y.(R(x,y) \wedge T(y,z)))$$

Minimal Model Semantics

- Given: a database **D** = (D, R_1, ..., R_k)
- Given: a datalog program **P**
- The answer **P(D)** consists of relations R_{k+1}, ..., $R_{k+p.}$
- Equivalently: **P(D)** is **D'** = (D, R_1, ..., R_k, R_{k+1}, ..., R_{k+p}) which is an extension of **D** (i.e. R_1, ..., R_k are the same as in **D**).
- In the sequel, **D'**, **D''**, denote extensions of **D**.

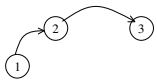
Minimal Model Semantics

- We say that **D'** is a **model of P**, if **D'** $\models \Phi_P$
- We say that **D'** is the **minimal model** of **P** if for any other model **D''**, **D'** $\subseteq$ **D''**
- ***Proposition*** The minimal model always exists and is unique.
- ***Definition***. **P(D)** is defined to be the minimal model of **P** extending **D**.

Example of Models

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

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



1	2
1	3
2	3

Minimal model T

Some other model T

1	2
1	3
2	3
3	2
2	2

Least Fixpoint

- For each rule r, φ_r defines a query
  - φ_r is a simple select-project-join query
- For each IDB predicate R, consider all rules with R in the head: they define a query, q_R
  - q_R is the union of all φ_r 's
- Given **D'** = (D, R_1, ..., R_k, R_{k+1}, ..., R_n), let

$$T_P(\mathbf{D'}) = (D, R_1, ..., R_k, q_{R_{k+1}}(D'), ..., q_{R_{k+p}}(D'))$$

Least Fixpoint

- In English: $\mathbf{T_P(D')}$ applies the program P once, affecting the IDB relations.
- ***Fact***. T_P is monotone: **D'** $\subseteq$ **D''** implies $\mathbf{T_P(D')} \subseteq \mathbf{T_P(D'')}$
- ***Definition*** **P(D)** is defined to be the least fixpoint of $\mathbf{T_P}$.

Least Fixpoint

- OOPS. Now we have two meanings for **P**(**D**) ?? Formally:

Definition **D'** is a fixpoint of $\mathbf{T_P}$ if $\mathbf{D' = T_P(D')}$

Definition **D'** is a prefixpoint of $\mathbf{T_P}$ if $\mathbf{D' \supseteq T_P(D')}$

Theorem [Tarski] A monotone operator on a lattice has a least fixpoint and it coincides with the least prefixpoint.

Proposition **D'** is a prefixpoint of $\mathbf{T_P}$ iff it is a model of **P**

Consequence: least fixpoint = minimal model

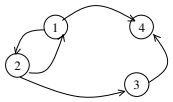
Naive Datalog Evaluation Algorithm

Standard way to compute a least fixpoint:

- $\mathbf{D'_0}$ = (D, R_1, ..., R_k, ϕ, ..., ϕ),
- $\mathbf{D'_1 = T_P(D'_0)}$
- $\mathbf{D'_2 = T_P(D'_1)}$
- ...
- $\mathbf{D'_{m+1} = T_P(D'_m)}$
- Stop when $\mathbf{D'_{m+1} = D'_m}$, define $\mathbf{T_P(D) = D'_m}$

Example

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



- $\mathbf{D'_0}$: T is empty
- $\mathbf{D'_1}$: T contains paths of length 1
- $\mathbf{D'_2}$: T contains paths of length 2
- $\mathbf{D'_3}$: T contains paths of length 3
- $\mathbf{D'_4 = D'_3}$ stop.

Data Complexity of Datalog

- $\mathbf{D'_0 \subset D'_1 \subset ... \subset D'_m = D'_{m+1}}$
- Let $n = |D|$, and let the IDB relations in **P** have arities $a_1, ..., a_p$.
- Then:

$$m \leq n^{a_1} + n^{a_2} + ... + n^{a_p}$$

- ***Theorem*** The data complexity of datalog is PTIME.

Datalog and Prolog

Datalog:

- naive evaluation algorithm is bottom-up

Prolog:

- evaluation is top-down

Datalog and First Order Logic

- Datalog is more expressive:
  - Can express recursive queries, such as transitive closure
- Datalog is less expressive:
  - Can only express monotone queries