Lecture 15: A Crash Course in Datalog

Yu Feng

Winter 2020

Logic programming

- Logic programming: the use of mathematical logic for computer programming
- Prolog (1972)
  - Use logical rules to specify how mathematical relations are computed
  - Turing complete
  - Dynamically typed

Logic programming overview

- Logic Programming based on logical rules
- A prolog program is a database of logical rules
- Example:
  - goleta is sunny
  - ucsb is sunny
  - ucsb is hot
  - If a location is both sunny and hot, then it is dry
- Search for solutions based on rules
  - Query: which place is dry?

Datalog

- Every Datalog program is a Prolog program
- Enforce several restrictions
- As a result, Datalog is pure declarative programming
  - All Datalog programs terminate
  - Ordering of rules do not matter
  - Not Turing complete
  - Efficient implementations typically based on databases

A brief history of Datalog

- E.F. Codd invented relational algebra and relational calculus in 1970. And then there was SQL.
- In 1979, Aho and Ullman pointed out that SQL cannot express recursive queries.
- In 1982, Chandra and Harel embarked on the study of the expressive power of Datalog.
- Between 1982 and 1995, Datalog "took the field by storm".
- After 1995, interest in Datalog decreased
- However, Datalog continued to find uses and applications in other areas, such as constraint satisfaction. And in recent years, Datalog has made a striking comeback!

Facts and rules

Facts: tuples in the database

Actor(344759,'Douglas','Fowley').
Casts(344759, 29851).
Casts(355713, 29000).
Movie(7909,'A Night in Armour', 1910).
Movie(29000,'Arizona', 1940).
Movie(29445,'Ave Maria', 1940).

Rules: queries

Q1(y) :- Movie(x,y,z), z='1940'

Q2(f,l):- Actor(z,f,l),Casts(z,x), Movie(x,y,'1940')

Q2(f,l):- Actor(z,f,l),Casts(z,x1),
Movie(x1,y1,'1910'),
Casts(z,x2), Movie(x2,y2,'1940')

Find movies made in 1940

Find actors who acted in movies made in 1940

Find actors who acted in a movie in 1940 and in one in 1910.

Datalog basics

Head Subgoal

Q2(f,l):- Actor(z,f,l),Casts(z,x), Movie(x,y,'1940')

Body

Atom= Actor(z,f,l)

Literal= Atom or Not Atom

Rule= Atom :- Literal & … & Literal

A program is a collection of logical rules

- h :− l_1, l_2, …, l_n.
- li is either a predicate or the negation of a predicate
- Semantics: h is true when l_1, l_2, …, and l_n are **simultaneously** true

Datalog examples

x is the parent of y

x has child y

```
parent(x,y) :- child(y,x).
grandparent(x,z) :- parent(x,y), parent(y,z).

ancestor(x,y) :- parent(x,y).
ancestor(x,z) :- parent(x,y), ancestor(y,z).
```

_ means “Dont-care”

! means “Not”

```
hasChild(x) :- child(_,x).
hasNoChild(x) :- !child(_,x).
hasSibling(x) :- child(x,y), child(z,y), z!=x.
```

- subgoals in the body are combined by “and”
- Multiple rules for a predicate (head) are combined by “or.”

Datalog framework

- We will use Souffle: https://souffle−lang.github.io/.
- Demo for the dry program

.decl sunny(c:symbol)

.decl hot(c:symbol)

.decl dry(c:symbol)

.output **dry**

sunny("goleta").

sunny("ucsb").

hot("ucsb").

dry(c) :- sunny(c), hot(c).

Datalog programming model

- A program is a set of rules (i.e., Horn clauses)
- The dry program has 3 facts and 1 rule (or 4 rules)
- Notes:
  - The rule holds for any instantiation of its variables (c="goleta" or "ucsb")
  - Closed−world assumption: anything not declared is not true
  - Ordering of rules does not matter for results (differ from Prolog)

Transitive Closure Example

```
.decl edge(x:number, y:number)
.input edge

.decl path(x:number, y:number)
.output path

path(x, y) :- edge(x, y).
path(x, y) :- path(x, z), edge(z, y).
```

Datalog in Security: pointer analysis

- Assume program consists of statements of form
  - p = &a (address of, includes allocation statements)
  - p = q
  - *p = q
  - p = *q

- Pointer analysis: compute the locations that each variable may point to

Anderson pointer analysis

- View pointer assignments as **subset** constraints
- Use constraints to propagate points-to information
- Worst case complexity: $O(n^3)$, where n = program size

Constraint type	Assignment	Constraint	Meaning
Base	$a = \&b$	$a \supseteq \{b\}$	$loc(b) \in pts(a)$
Simple	$a = b$	$a \supseteq b$	$pts(a) \supseteq pts(b)$
Complex	$a = *b$	$a \supseteq *b$	$\forall v \in pts(b).\ pts(a) \supseteq pts(v)$
Complex	$*a = b$	$*a \supseteq b$	$\forall v \in pts(a).\ pts(v) \supseteq pts(b)$

pts(a,b) :- loc(b), addressTaken(a,b).

pts(a,c) :- pts(b,c), assign(a,b).

??

??