Lecture 17: Relational Queries

1 Introduction

In last lecture we discussed how important is abstraction for computer science. Also we spent some time talking about How modern day relational databases has been modelled on relations and formulas .

1. Relations are Tables.
2. Formulas are queries .
3. If φ is unsat, then $\varphi(A) = \emptyset$.

2 Query Evaluation

We saw early in the course that there was a distinction between syntax and the world, and that $\models$ is the connection between the two. It is interesting that this distinction was actually made late in the study of logic. The idea is attributed to Tarski who was a famous 20^{th} century logician. We have seen the semantics of $\models$ and how it is evaluated. Now we turn to the task of defining what it means to evaluate a query.

$A = (R, <, = 0, +, *)$ is infinite so it cannot be a database. In fact, a database is a *finite relational structure*. Now that we know how we can write queries, we should focus on finding efficient ways to 'execute' the queries. This is referred to as the query evaluation problem.

Definition 1 (Query Evaluation Problem) *Given A and φ, compute $\varphi(A)$.*

Recall that $\varphi(A) = \{\langle a_1, ..., a_k\rangle \mid A, < a_1, \ldots, a_k > \models \varphi\})$. Instead, we can ask: given A, α, and φ, does $A, \alpha \models \varphi$? After we have a way to answer this, then we can enumerate over all α to evaluate our query.

If we want to enumerate all α's there are $D^{Fvars(\varphi)}$ different assignments to check. So, here is an algorithm:

```
Truth($A, \alpha, \varphi$)
Case
  $\varphi$ is $p(x_1, \ldots, x_k)$ : return $(\alpha(x_1), \ldots, \alpha(x_k)) \in p^A$
  $\varphi$ is $\neg\psi$ : return $\neg$Truth($A$,$\alpha$,$\psi$)
  $\varphi$ is $\theta \circ \psi$ : return $\circ$(Truth($A$,$\alpha$,$\theta$),Truth($A$,$\alpha$,$\psi$))
```

```
$\varphi$ is $(\exists x)\psi$ :
  B := false
  do for $a \in D$
    B := B$\vee$Truth($A$,$\alpha[x \mapsto a]$,$\psi$)
  od
  return B
$\varphi$ is $(\forall x)\psi$ :
  B := true
  do for $a \in D$
    B := B$\wedge$Truth($A$,$\alpha[x \mapsto a]$,$\psi$)
  od
  return B
```

Notice that the last case for $Truth()$ is not really necessary, since we can actually rewrite the formula by $(\exists x)\varphi \models\!\!\dashv \neg(\forall x)(\neg\varphi)$ and $(\forall x)\varphi \models\!\!\dashv \neg(\exists x)(\neg\varphi)$

Theorem 1 *$A, \alpha \models \varphi$ iff $Truth(A, \alpha, \varphi) = 1$.*

This can be easily proved by the semantic definition of the formula.

Now, what is the complexity of $Truth()$ in terms of its inputs, i.e. in terms of $|A| + |\alpha| + |\varphi|$?

The definition is by induction on the syntax of the formula. None of the cases generate many more recursions, except for the $\varphi = (\forall x)\psi$ and $\varphi = (\exists x)\psi$ cases, which actually include nested loops. So, the worst-case possible formula is $\forall x_1 \exists x_2 \forall x_3 \exists x_4 ... \forall x_{n-1} \exists x_n p(x_1, x_2, x_3, ..., x_n)$. This is $O(|A|^{\#vars})$. Therefore, the time complexity is $(|A| + |\alpha| + |\varphi|)^{|\varphi|}$, which is exponential in $|\varphi|$. The space complexity is $|\varphi| * (|\varphi| * log(|A|))$, which is polynomial in φ, and logarithmic in A. So, this is still in $PSPACE$.

Now is this good news or bad news? The answer is that it depends on your perspective. Notice that the complexity depends differently on φ and on A. The idea is to analyze the complexity while one of the values constant. We can think of the query as a fixed size and the database variable. This would correspond to running the same query on data that is constantly updating. We can also hold the database to a fixed size and vary the size of the query. This would correspond to keeping the data the same and varying the query. We examine these possibilities below.

Recall that $P \subseteq NP \subseteq PSPACE$. This problem is $PSPACE - complete$. However, if we consider this problem in terms of different parts of the input, we get different complexities:

- Complexity of $\{\langle A, \alpha \rangle | A, \alpha \models \varphi\}$. So, this is the complexity if we keep φ fixed. This is called *data complexity*, and it is PTIME, and LOGSPACE.

- Complexity of $\{\langle \alpha, \varphi \rangle | A, \alpha \models \varphi\}$. This is the complexity of we keep the structure fixed. This is the case with databases, since the database uses a specific structure, so this is called *query complexity.* This is EXPTIME, and PSPACE. Actually, this is also $PSPACE - complete$.

- Complexity of $\{\langle A, \alpha, \varphi\rangle | A, \alpha \models \varphi\}$. The *combined complexity* makes this $PSPACE - complete$.

Usage Scenario:

1. Fixed query: varying data like share cost/stock market. In this case complexity is polynomial.

2. Varying query:fixed data like census , Complexity may be exponential.

1: Data complexity $\in logspace$

2: Fixed data : query complexity $\in PSPACE - COMPLETE$
$P \subseteq NP \subseteq PSPACE$

1. Query complexity is exponentially higher then data complexity.

2. Query answering can be hard .

3 Classifying formulas and queries

1. φ - satisfiable: for some A and α we have $A, \alpha \models \varphi$

2. φ - unsatisfiable: for all A and α we have $A, \alpha \not\models \varphi$

3. φ - valid: for all A and α we have $A, \alpha \models \varphi$

1. $\varphi(A) \neq \emptyset$ for some $A - \varphi$ satisfiable

2. $\varphi(A) = \emptyset$ for all $A - \varphi$ unsatisfiable

3. $\varphi(A) = (D^A)^k - \varphi$ valid

4. We call a query *interesting* if it is satisfiable but not valid.

1. $\varphi \equiv \psi$: $A, \alpha \models \varphi$ iff $A, \alpha \models \psi$ for all A,α

2. $\varphi(A) = \psi(A)$ for all A iff $\varphi \equiv \psi$

Note: Satisfiability, validity, and equivalence for first-order logic are *undecidable.*