CSE 544: Principles of Database Systems

Query Complexity
Announcements

• Project milestone due tomorrow night

• HW3 due on Monday
Query Complexity

Fix some query language $L$

• What is the complexity of evaluating queries in $L$ on input databases?

• The answer gives us several insights:
  – What physical operators we need for $L$
  – What queries can and cannot we write in $L$
  – Whether we need to extend $L$
We are interested in these classes

All computable problems

\( PSPACE \)

\( NP \)

\( \text{PTIME} \)

\( \text{NC}^k \) …

\( \text{NL} \)

\( L \)

\( \text{AC}^0 \)

Please review them for next Tuesday; we will briefly discuss them in class.
Measuring Size

• The **size** of a database D is the **number of tuples**

• Alternatively, the **size** of D is the **number of constants in its active domain**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>f</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
</tr>
<tr>
<td>a</td>
<td>f</td>
</tr>
</tbody>
</table>

Number of tuples: 5+7 = 12

ADom(D)={a,b,c,d,e,f,g}
Size of the active domain = 7

These two are related (how?)
Example

Q(x) = ∃y.R(x,y) ∧ (∀z.S(y,z) → ∃u.R(z,u))

Database D:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>b</td>
<td>f</td>
</tr>
<tr>
<td>b</td>
<td>g</td>
</tr>
<tr>
<td>a</td>
<td>f</td>
</tr>
</tbody>
</table>

R =

<table>
<thead>
<tr>
<th>c</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td>b</td>
</tr>
<tr>
<td>g</td>
<td>b</td>
</tr>
<tr>
<td>f</td>
<td>a</td>
</tr>
</tbody>
</table>

S =

Give an algorithm for computing Q on any input D. Express its complexity as a function of* n = |ADom(D)|

* Active Domain = all constants in D. ADom(D) = {a,b,c,...}
Example

\[ Q(x) = \exists y. R(x,y) \land (\forall z. S(y,z) \implies \exists u. R(z,u)) \]

What is the running time \( f(n) \)?

In what complexity class is this query?
Example

\[
Q(x) = \exists y. R(x,y) \land (\forall z. S(y,z) \implies \exists u. R(z,u))
\]

What is the running time \( f(n) \)?

**Complexity = \( O(n^4) \)**

In what complexity class is this query?

4 = number of variables
Example

\[ Q(x) = \exists y. R(x,y) \land (\forall z. S(y,z) \rightarrow \exists u. R(z,u)) \]

What is the running time \(f(n)\)?

Complexity = \(O(n^4)\)

In what complexity class is this query?

PTIME
Example

\[ Q(x) = \exists y. R(x,y) \land (\forall z. S(y,z) \rightarrow \exists u. R(z,u)) \]

What is the running time \( f(n) \)?

Complexity = \( O(n^4) \)

In what complexity class is this query?

\[ \text{LOGSPACE} \ (\text{why?}) \]
Example

\[ Q(x) = \exists y. R(x, y) \land (\forall z. S(y, z) \rightarrow \exists u. R(z, u)) \]

for \( x \) in ADom do
  good\(_y\) = false
  for \( y \) in ADom do
    if \( (x, y) \in R \) then
      good\(_z\) = true
      for \( z \) in ADom do
        if \( (y, z) \in S \) then
          good\(_u\) = false
          for \( u \) in ADom do
            if \( (z, u) \in R \) then good\(_u\) = true
          endfor
        endfor
        if not good\(_u\) then good\(_z\) = false
      endfor
      if good\(_z\) then good\(_y\) = true
    endfor
  endfor
  if good\(_y\) then output \( x \)
endfor

What is the running time \( f(n) \)?

Complexity = \( O(n^4) \)

In what complexity class is this query?

FL, \( AC^0 \) (why?)
Discussion

• The query complexity of $O(n^4)$ if we assumed that the query is fixed and only the database is variable.

• If we assume that both query and database are variable, then the complexity is different (and much higher).

• Data complexity v.s. Query complexity.
Query Q, database D

• **Data complexity:**
  fix Q, complexity = f(D)

• **Query complexity:**
  fix D, complexity = f(Q)

• **Combined complexity:**
  complexity = f(D,Q)
Data Complexity

Given a query Q in a query language L, what is the complexity of the following problem? “Given D, compute Q(D)”

• Language design tradeoff
  – High complexity $\rightarrow$ L can express rich queries
  – Low complexity $\rightarrow$ L can be implemented efficiently
Conventions

• The complexity is usually defined for a decision problem
  – Hence, we will study only the complexity of Boolean queries

• The complexity usually assumes some encoding of the input
  – Hence we will encode the database instances using a binary representation
Boolean Queries

**Definition** A *Boolean Query* is a query that returns either true or false

**Factoid**: the complexity of $L$ is fully determined by its Boolean queries
**Boolean Queries**

**Definition** A *Boolean Query* is a query that returns either true or false.

**Factoid**: the complexity of $L$ is fully determined by its Boolean queries.

Non-boolean queries:

$$Q(x,y) = \exists z. R(x,z) \land S(z,y)$$

Boolean queries:

$$Q = \exists z. R('a',z) \land S(z,'b')$$

$a, b \in \text{ADom}(D)$
**Definition** A *Boolean Query* is a query that returns either true or false.

Q(x,y) = ∃z.R(x,z) ∧ S(z,y)

**Factoid**: the complexity of L is fully determined by its Boolean queries.

### Non-boolean queries

Q(x,y) = ∃z.R(x,z) ∧ S(z,y)

SELECT DISTINCT R.x, S.y FROM R, S WHERE R.z = S.z

### Boolean queries:

Q = ∃z.R('a',z) ∧ S(z,'b') \( a, b \in \text{ADom}(D) \)

SELECT DISTINCT 'yes' FROM R, S WHERE R.x='a' and R.y = S.y and S.y='b'
**Definition** A *Boolean Query* is a query that returns either true or false

**Factoid**: the complexity of $L$ is fully determined by its Boolean queries

**Non-boolean queries**

- $Q(x, y) = \exists z. R(x, z) \land S(z, y)$
- $\text{SELECT DISTINCT R.x, S.y FROM R, S WHERE R.z = S.z}$
- $Q(x) = \exists y. R(x, y) \land (\forall z. S(y, z) \rightarrow \exists u. R(z, u))$

**Boolean queries**:

- $Q = \exists z. R('a', z) \land S(z, 'b')$
- $\text{SELECT DISTINCT 'yes' FROM R, S WHERE R.x='a' and R.y = S.y and S.y='b'}$
- $Q = \exists y. R('a', y) \land (\forall z. S(y, z) \rightarrow \exists u. R(z, u))$
- $\text{SELECT DISTINCT R.x, S.y FROM R, S WHERE R.z = S.z}$

- $Q = \exists z. R('a', z) \land S(z, 'b')$
- $\text{SELECT DISTINCT 'yes' FROM R, S WHERE R.x='a' and R.y = S.y and S.y='b'}$
- $Q = \exists y. R('a', y) \land (\forall z. S(y, z) \rightarrow \exists u. R(z, u))$
**Definition** A *Boolean Query* is a query that returns either true or false

**Factoid**: the complexity of $L$ is fully determined by its Boolean queries

### Non-boolean queries

- $Q(x,y) = \exists z. R(x,z) \land S(z,y)$
- SELECT DISTINCT R.x, S.y
  FROM R, S
  WHERE R.z = S.z
- $Q(x) = \exists y. R(x,y) \land (\forall z. S(y,z) \rightarrow \exists u. R(z,u))$
- $T(x,y) :- R(x,y)$
  $T(x,y) :- T(x,z), R(z,y)$

### Boolean queries: Non-boolean queries

- $Q = \exists z. R('a',z) \land S(z,'b')$
  $a,b \in ADom(D)$
- SELECT DISTINCT ‘yes’
  FROM R, S
  WHERE R.x='a' and R.y = S.y and S.y='b'
- $Q = \exists y. R('a',y) \land (\forall z. S(y,z) \rightarrow \exists u. R(z,u))$
- $T(x,y) :- R(x,y)$
  $T(x,y) :- T(x,z), R(z,y)$
- Answer() :- T('a','b')

**Transitive closure**

**Reachability**
Database Encoding

• We could encode a database D as a list of encodings of tuples, similar to how database systems store data on disk:
  – “[R(a,b),R(cac,bbfg),R(cac,b)],[S(b),S(cac)]”

• Instead, we use a simpler representation, generalizing adjacency matrix for graphs
Database Encoding

Encode $\mathbf{D} = (D, R_1^D, \ldots, R_k^D)$ as follows:

- Let $n = |\text{ADom}(D)|$
- If $R_i$ has arity $k$, then encode it as a string of $n^k$ bits:
  - 0 means element $(a_1,\ldots, a_k) \notin R_i^D$
  - 1 means element $(a_1,\ldots, a_k) \in R_i^D$

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
The Data Complexity

Fix any Boolean query $Q$ in the query language. Determine the complexity of the following problem:

- Given an input database instance $D = (D, R_1^D, \ldots, R_k^D)$, check if $Q(D) = \text{true}$.

- This is also known as the **Model Checking Problem**: check if $D$ is a model for $Q$. 
What we will discuss next time

• Relational queries
• Datalog and stratified datalog
• Datalog with inflationary fixpoint
• Datalog with partial fixpoint

• $AC^0$
• $L = \text{a.k.a. LOGSPACE}$
• $NL = \text{a.k.a. NLOGSPACE}$
• $NC$
• $P = \text{a.k.a. PTIME}$
• $NP$
• $PSPACE$