DATABASE THEORY

Lecture 4: Complexity of FO Query Answering

Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 23rd Apr 2019
How to Measure Query Answering Complexity

Query answering as decision problem
\[ \sim \] consider Boolean queries

Various notions of complexity:
- Combined complexity (complexity w.r.t. size of query and database instance)
- Data complexity (worst case complexity for any fixed query)
- Query complexity (worst case complexity for any fixed database instance)

Various common complexity classes:

\[ L \subseteq NL \subseteq P \subseteq NP \subseteq PSpace \subseteq \text{ExpTime} \]
An Algorithm for Evaluating FO Queries

\[
\text{function } \text{Eval}(\varphi, I) \text{ }
\]

01 \hspace{1em} \text{switch } (\varphi) \{  
02 \hspace{1em} \text{case } p(c_1, \ldots, c_n): \text{ return } \langle c_1, \ldots, c_n \rangle \in p^I  
03 \hspace{1em} \text{case } \neg \psi: \text{ return } \neg \text{Eval}(\psi, I)  
04 \hspace{1em} \text{case } \psi_1 \land \psi_2: \text{ return } \text{Eval}(\psi_1, I) \land \text{Eval}(\psi_2, I)  
05 \hspace{1em} \text{case } \exists x. \psi:  
06 \hspace{1em} \hspace{1em} \text{for } c \in \Delta^I \{  
07 \hspace{1em} \hspace{1em} \text{if } \text{Eval}(\psi[x \mapsto c], I) \text{ then return true}  
08 \hspace{1em} \hspace{1em} \}  
09 \hspace{1em} \text{return false}  
10 \hspace{1em} \}
Let $m$ be the size of $\varphi$, and let $n = |I|$ (total table sizes)
Let $m$ be the size of $\varphi$, and let $n = |\mathcal{I}|$ (total table sizes)

- How many recursive calls of Eval are there?
  $\leadsto$ one per subexpression: at most $m$

- Maximum depth of recursion?
  $\leadsto$ bounded by total number of calls: at most $m$

- Maximum number of iterations of for loop?
  $\leadsto$ $|\Delta I| \leq n$ per recursion level
  $\leadsto$ at most $n^m$ iterations

- Checking $\langle c_1, \ldots, c_n \rangle \in p^I$ can be done in linear time w.r.t. $n$

Runtime in $m \cdot n^m \cdot n = m \cdot n^{m+1}$
Let \( m \) be the size of \( \varphi \), and let \( n = |\mathcal{I}| \) (total table sizes)

Runtime in \( m \cdot n^{m+1} \)

**Time complexity of FO query evaluation**

- Combined complexity: in \( \text{ExpTime} \)
- Data complexity (\( m \) is constant): in \( \text{P} \)
- Query complexity (\( n \) is constant): in \( \text{ExpTime} \)
We can get better complexity bounds by looking at memory.

Let $m$ be the size of $\varphi$, and let $n = |\mathcal{I}|$ (total table sizes).

- For each (recursive) call, store pointer to current subexpression of $\varphi$: $\log m$
- For each variable in $\varphi$ (at most $m$), store current constant assignment (as a pointer): $m \cdot \log n$
- Checking $\langle c_1, \ldots, c_n \rangle \in p^I$ can be done in logarithmic space w.r.t. $n$

Memory in $m \log m + m \log n + \log n = m \log m + (m + 1) \log n$
Space Complexity of FO Algorithm

Let $m$ be the size of $\varphi$, and let $n = |I|$ (total table sizes)

Memory in $m \log m + (m + 1) \log n$

**Space complexity of FO query evaluation**
- Combined complexity: in PSpace
- Data complexity ($m$ is constant): in L
- Query complexity ($n$ is constant): in PSpace
The algorithm shows that FO query evaluation is in PSpace.
Is this the best we can get?

**Hardness proof:** reduce a known PSpace-hard problem to FO query evaluation
The algorithm shows that FO query evaluation is in PSpace. Is this the best we can get?

**Hardness proof:** reduce a known PSpace-hard problem to FO query evaluation

\[ \leadsto \text{QBF satisfiability} \]

Let \( \mathcal{Q}_1 X_1. \mathcal{Q}_2 X_2. \cdots \mathcal{Q}_n X_n \cdot \varphi[X_1, \ldots, X_n] \) be a QBF (with \( \mathcal{Q}_i \in \{ \forall, \exists \} \))

- Database instance \( \mathcal{I} \) with \( \Delta^I = \{0, 1\} \)
- One table with one row: true(1)
- Transform input QBF into Boolean FO query

\[
\mathcal{Q}_1 x_1. \mathcal{Q}_2 x_2. \cdots \mathcal{Q}_n x_n \cdot \varphi[X_1 \mapsto \text{true}(x_1), \ldots, X_n \mapsto \text{true}(x_n)]
\]

It is easy to check that this yields the required reduction. \( \square \)
PSpace-hardness for DI Queries

The previous reduction from QBF may lead to a query that is not domain independent.

Example: QBF $\exists p. \neg p$ leads to FO query $\exists x. \neg \text{true}(x)$
PSpace-hardness for DI Queries

The previous reduction from QBF may lead to a query that is not domain independent

**Example**: QBF $\exists p. \neg p$ leads to FO query $\exists x. \neg \text{true}(x)$

**Better approach**:

- Consider QBF $\varphi_1 X_1 \varphi_2 X_2 \cdots \varphi_n X_n \varphi[X_1, \ldots, X_n]$ with $\varphi$ in negation normal form: negations only occur directly before variables $X_i$ (still PSpace-complete: exercise)
- Database instance $\mathcal{I}$ with $\Delta^\mathcal{I} = \{0, 1\}$
- Two tables with one row each: true(1) and false(0)
- Transform input QBF into Boolean FO query

$$\varphi_1 x_1 \varphi_2 x_2 \cdots \varphi_n x_n \varphi'$$

where $\varphi'$ is obtained by replacing each negated variable $\neg X_i$ with $\text{false}(x_i)$ and each non-negated variable $X_i$ with $\text{true}(x_i)$. 
Summing up, we obtain:

**Theorem 4.1:** The evaluation of FO queries is PSpace-complete with respect to combined complexity.
Summing up, we obtain:

**Theorem 4.1:** The evaluation of FO queries is PSpace-complete with respect to combined complexity.

We have actually shown something stronger:

**Theorem 4.2:** The evaluation of FO queries is PSpace-complete with respect to query complexity.
Summary and Outlook

The evaluation of FO queries is

- PSpace-complete for combined complexity
- PSpace-complete for query complexity

**Open questions:**

- What is the data complexity of FO queries?
- Are there query languages with lower complexities? (next lecture)
- Which other computing problems are interesting?