TECHNISCHE UNIVERSITÄT DRESDEN

DATABASE THEORY

Lecture 4: Complexity of FO Query Answering

Sebastian Rudolph

Computational Logic Group

Slides based on Material of Markus Krötzsch and David Carral

TU Dresden, 26th Apr 2021

How to Measure Query Answering Complexity

Query answering as decision problem
$\rightsquigarrow$ 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:

$$\text{L} \subseteq \text{NL} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSpace} \subseteq \text{ExpTime}$$

An Algorithm for Evaluating FO Queries

function Eval($\varphi, \mathcal{I}$)

```
switch ($\varphi$) {
    case $p(c_1, \ldots, c_n)$ : return $\langle c_1, \ldots, c_n \rangle \in p^{\mathcal{I}}$
    case $\neg\psi$ : return $\neg$Eval($\psi, \mathcal{I}$)
    case $\psi_1 \wedge \psi_2$ : return Eval($\psi_1, \mathcal{I}$) $\wedge$ Eval($\psi_2, \mathcal{I}$)
    case $\exists x.\psi$ :
        for $c \in \Delta^{\mathcal{I}}$ {
            if Eval($\psi[x \mapsto c], \mathcal{I}$) then return true
        }
        return false
}
```

FO Algorithm Worst-Case Runtime

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

- Maximum depth of recursion (=max tree depth)?
  $\rightsquigarrow$ in an Eval call (on a formula), Eval is called recursively only on **shorter** formulas
  $\rightsquigarrow$ recursion depth bounded by length of φ: at most m
- Maximum number of direct calls from within one Eval call (=max branching degree)?
  $\rightsquigarrow$ $|\Delta^{\mathcal{I}}| \leq n$ (lines 06–08) – unless $|\Delta^{\mathcal{I}}| < 2$, then it may still be 2 (line 04)
  $\rightsquigarrow$ to be on the safe side, pick $n + 2$
- Maximum number of total Eval calls?

$$\sum_{\text{depth}=0}^{\text{max tree depth}} (\text{max branching degree})^{\text{depth}} = \sum_{i=0}^{m} (n+2)^i \leq (n+2)^{m+1}$$

- Maximum time needed for one Eval call (without subcalls)?
  $\rightsquigarrow$ Checking $\langle c_1, \ldots, c_n \rangle \in p^{\mathcal{I}}$ can be done in linear time w.r.t. n (line 02)
  $\rightsquigarrow$ so can the **`for`** loop (lines 06-08), all other cases are less costly

Runtime in $(n+2)^{m+1} \cdot O(n) \leq O((n+2)^{m+2})$

Time Complexity of FO Algorithm

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

Runtime in $O((n+2)^{m+2})$

Time complexity of FO query evaluation

- Combined complexity: in ExpTime
- Data complexity (m is constant): in P
- Query complexity (n is constant): in ExpTime

FO Algorithm Worst-Case Memory Usage

We can get better complexity bounds by looking at memory

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

- For each (recursive) call, store pointer to current subexpression of φ: $\log m$
- For each variable in φ (at most m), store current constant assignment (as a pointer): $m \cdot \log n$
- Checking $\langle c_1, \ldots, c_n \rangle \in p^{\mathcal{I}}$ 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 φ, and let $n = |\mathcal{I}|$ (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

FO Combined Complexity

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
$\rightsquigarrow$ QBF satisfiability

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

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

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

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 $\exists x.\neg\text{true}(x)$

Better approach:

- Consider QBF $\mathsf{Q}_1 X_1.\mathsf{Q}_2 X_2.\cdots\mathsf{Q}_n X_n.\varphi[X_1,\ldots,X_n]$ with φ 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

$$\mathsf{Q}_1 x_1.\mathsf{Q}_2 x_2.\cdots\mathsf{Q}_n x_n.\varphi'$$

  where φ' is obtained by replacing each negated variable $\neg X_i$ with false(x_i) and each non-negated variable X_i with true(x_i).

Combined Complexity of FO Query Answering

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?