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

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:

\[
L \subseteq NL \subseteq P \subseteq NP \subseteq PSpace \subseteq ExpTime
\]
function Eval(\(\varphi, I\))

01 switch (\(\varphi\)) {
02 case \(p(c_1, \ldots, c_n)\) : return \(\langle c_1, \ldots, c_n \rangle \in p^I\)
03 case \(\neg \psi\) : return \(\neg \text{Eval}(\psi, I)\)
04 case \(\psi_1 \land \psi_2\) : return \(\text{Eval}(\psi_1, I) \land \text{Eval}(\psi_2, I)\)
05 case \(\exists x. \psi\) :
06 for \(c \in \Delta^I\) {
07 if \(\text{Eval}(\psi[x \mapsto c], I)\) then return true
08 }
09 return false
10 }

Markus Krötzsch, 17th Apr 2018
Let $m$ be the size of $\varphi$, and let $n = |I|$ (total table sizes)
Let $m$ be the size of $\varphi$, and let $n = |I|$ (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 \texttt{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 ExpTime
- Data complexity ($m$ is constant): in P
- Query complexity ($n$ is constant): in ExpTime
We can get better complexity bounds by looking at memory

Let $m$ be the size of $\varphi$, and let $n = |I|$ (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 $m \log m + m \log n + \log n = m \log m + (m + 1) \log n$
Let $m$ be the size of $\varphi$, 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
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

~ QBF satisfiability

Let $Q_1 x_1. Q_2 x_2. \cdots Q_n x_n. \varphi[X_1, \ldots, X_n]$ be a QBF (with $Q_i \in \{\forall, \exists\}$)

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

$$Q_1 x_1. Q_2 x_2. \cdots Q_n x_n. \varphi[X_1 \mapsto \text{true}(x_1), \ldots, X_n \mapsto \text{true}(x_n)]$$

It is easy to check that this yields the required reduction. □
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 $\bigwedge_{1 \leq i \leq n} Q_i X_i. \varphi[ X_1, \ldots, X_n ]$ 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

$$\bigwedge_{1 \leq i \leq n} Q_i x_i. \bigwedge_{1 \leq i \leq n} \varphi'$$

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