8. PSPACE

Reinhard Pichler

Institut für Informationssysteme
Arbeitsbereich DBAI
Technische Universität Wien

16 June, 2015

Motivation

PSPACE captures unrestricted alternation. Therefore, ... 
- it generalizes the polynomial hierarchy,
- it is the class of many strategy games, decision making, etc.,
- it has QSAT (QBF) as natural complete problem.

QSAT (QBF)

INSTANCE: Boolean expression $\varphi$ in CNF with variables $x_1, \ldots, x_n$.

QUESTION: Is there a truth value for the variable $x_1$ such that for both 
truth values of $x_2$ there is a truth value for $x_3$ and so on up to $x_n$, such 
that $\varphi$ is satisfied by the overall truth assignment?

Notation

An instance of QSAT is written as $\exists x_1 \forall x_2 \exists x_3 \cdots Q x_n \varphi$, where $Q$ is $\forall$ if $n$ 
is even and $\exists$ if $n$ is odd.

Theorem

QSAT is PSPACE-complete.
Proof of the PSPACE-Membership of QSAT

Remark. We only prove the PSPACE-membership here. The hardness will be proved below via the complexity of First-Order Logic.

Let an arbitrary QBF be given as $\psi \equiv \exists x_1 \forall x_2 \exists x_3 \cdots Qx_n \varphi$. All possible truth assignments of the variables can be represented by the leaves in a full binary tree of depth $n$ (="semantic tree"): The left subtree of the root contains all truth assignments $T$ with $T(x_1) = \text{false}$, while the right subtree of the root contains all truth assignments $T$ with $T(x_1) = \text{true}$.

Analogously, for every $i \geq 1$, the subtrees at depth $i + 1$, whose root is the first child of its parent, contains all truth assignments $T$ with $T(x_{i+1}) = \text{false}$, while the subtrees at depth $i + 1$, whose root is the second child of its parent, contains all truth assignments $T$ with $T(x_{i+1}) = \text{true}$.

Proof of the PSPACE-Membership of QSAT (continued)

The linear space bound on the evaluation of the Boolean circuit follows immediately from the following observation: At any time, the algorithm only needs to store (the label of) exactly 1 gate of the tree, namely the current gate $g$ of the evaluation.

Implicitly, we thus have the entire path from $g$ to the root. If the path contains a gate which is the first child of its parent $h$, then it is clear that the second child of $h$ has not been visited yet. If the path contains a gate which is the second child of $h$, then it is clear that the value of the first child of $h$ is true for an AND-gate $h$ and false for an OR-gate $h$.

The only difficulty remaining is that the circuit $C$ has exponential size. Observe that both, the construction of $C$ and the evaluation of $C$ work in polynomial space. Hence, the combination of these two algorithms is feasible in PSPACE – by the same idea as in the proof that the composition of two log-space computations is feasible in log-space.
**Complexity Theory**

**8. PSPACE**

8.1. QSAT (QBF)

**PSPACE vs. PH**

**Proposition**

QSAT is a generalization of the $\Sigma_i P$-complete problem QSAT$_i$ for any value of $i$.

**Corollary**

$PH \subseteq PSPACE$

**Remark**

It is not known if PH is properly included in PSPACE. Most probably, PH $\subset$ PSPACE holds, because PH = PSPACE would imply that the polynomial hierarchy collapses (since there exist PSPACE-complete problems).

**Games**

**Observation**

PSPACE is the class of many strategy games, decision making, etc. QSAT can be considered as a two-person game:

- two players: $\exists$ and $\forall$
- players move alternatingly ($\exists$ first)
- a move: determining the truth value of a variable
- $\exists$ tries to make the formula $\varphi$ true while $\forall$ tries to make it false.
- after $n$ moves either $\exists$ or $\forall$ wins.

Decision making can sometimes be considered as a game against nature.

**Complexity of Query Evaluation**

**Decision Problems**

For (Boolean) queries of a certain query language (e.g., SQL, datalog, XPath, XQuery, etc.), there are three main kinds of decision problems:

**Data complexity** refers to the following decision problem:

Let $Q$ be some fixed query.

INSTANCE: An input database $D$.

QUESTION: Does query $Q$ yield a non-empty result over the DB $D$?

**Query complexity** refers to the following decision problem:

Let $D$ be some fixed input database.

INSTANCE: A query $Q$.

QUESTION: Does query $Q$ yield a non-empty result over the DB $D$?

**Combined complexity** refers to the following decision problem:

INSTANCE: An input database $D$ and a query $Q$.

QUESTION: Does query $Q$ yield a non-empty result over the DB $D$?

**Definition**

A term is a constant or a variable.

For a given input schema $R = \{R_1, \ldots, R_n\}$, the base formulae are either equality atoms $s = t$ or atoms of the form $R(t_1, \ldots, t_n)$, where the $t_i$ are terms and $\alpha$ is the arity of $R$. A first-order query over $R$ is either a base formula or a formula of the following form:

1. $(\varphi \land \psi)$, where $\varphi$ and $\psi$ are formulae over $R$;
2. $(\varphi \lor \psi)$, where $\varphi$ and $\psi$ are formulae over $R$;
3. $\neg \varphi$, where $\varphi$ is a formula over $R$;
4. $\exists x \varphi$, where $x$ is a variable and $\varphi$ is a formula over $R$;
5. $\forall x \varphi$, where $x$ is a variable and $\varphi$ is a formula over $R$.

**Remark**

First-order queries essentially correspond to SQL without GROUP BY, (aggregate) functions and arithmetic.
PSPACE-Hardness of First-Order Queries

Proof of the PSPACE-Hardness

We prove the hardness by reduction from an arbitrary language \( L \) in PSPACE. To this end, we define a fixed database \( D \). Moreover, we describe a reduction \( R \) which, for every string \( w \), constructs a First-Order sentence \( R(w) \) such that \( w \in L \iff R(w) \) evaluates to true over \( D \).

Let \( T = (K, \Sigma, \delta, s) \) be a single-string Turing machine that decides \( L \) in polynomial space. W.l.o.g., we assume that on any positive instance \( w \), the TM \( T \) has exactly one accepting configuration, say \( (\text{"yes"}, \delta, \bigcup \subseteq \cdots) \). Assume that the computation on input \( w \) requires at most \( d \cdot n^k \) space with \( n = |w| \) and constants \( d, k \). Then the computation takes at most \( N = c^d \cdot n^k \) steps for some constant \( c \).

We first define the (fixed) input database \( D \): it just contains two unary relations \( K \) and \( \Sigma \) with the states and symbols, respectively, of \( T \).

Now let \( w \) be an arbitrary instance of \( L \). We have to construct an FO formula \( R(w) \). This construction is based on well-known ideas.

**Proof of the PSPACE-Hardness (continued)**

**Configurations.** Every configuration can be represented by a vector of length \( M = d \cdot n^k + 1 \): we represent \((q, u, v)\) with \( u = u_1, \ldots, u_\alpha \) and \( v = v_1, \ldots, v_\beta \) as \((u_1, \ldots, u_\alpha, q, v_1, \ldots, v_\beta, \bigcup \subseteq \cdots)\).

**Encoding of \( \text{PATH}(a, b, i) \).** For every \( i \in \{0, \ldots, \log N\} \) we define a formula \( \psi_i(x_1, \ldots, x_M, y_1, \ldots, y_M) \) with free variables \( x_1, \ldots, x_M, y_1, \ldots, y_M \), s.t. \( \psi_i \) is true in \( D \iff (x_1, \ldots, x_M) \) is instantiated to (the representation of) some configuration \( C_1, (y_1, \ldots, y_M) \) is instantiated to (the representation of) some configuration \( C_2 \), and there is a path of length at most \( 2^i \) from \( C_1 \) to \( C_2 \) in the configuration graph \( G(T,w) \).

**Reduction from \( L \) to FO evaluation.** Suppose that we have defined the predicates \( \psi_i(x_1, \ldots, x_M, y_1, \ldots, y_M) \). Let \( j = \log N \). Moreover, let \((a_1, \ldots, a_M)\) be the (representation of the) initial configuration \( C_0 \) on input \( w \) and let \((b_1, \ldots, b_M)\) be the accepting configuration \( C_{\text{\"yes"}} \).

We define \( \psi^* = \psi_j(a_1, \ldots, a_M, b_1, \ldots, b_M) \).

Then we have \( x \in L \iff \psi^* \) is true over \( D \).
PSPACE-Membership of First-Order Queries

Proof of the PSPACE-Membership (continued)

Let \( D \) be an arbitrary input database and let \( \varphi \) be an arbitrary first-order sentence. Moreover, let all constants in \( \varphi \) and all elements in \( D \) be from the domain \( \text{dom} \). We prove the PSPACE-membership by reducing the problem of evaluating \( \varphi \) over \( D \) to the QSAT problem.

1. Restricting the domain to \( \{0, 1\} \). Let \( \text{dom} = \{a_1, \ldots, a_n\} \). Then these elements can be encoded by bit-vectors of size \( m \approx \log(n) \). Let \( \vec{b}_i \) denote the encoding of \( a_i \). Then we transform \( D \) into \( D' \) by replacing any \( \alpha \)-ary relation \( r \) by an \((\alpha \cdot m)\)-ary relation \( r' \).

PSPACE-Hardness of First-Order Queries

Proof of the PSPACE-Hardness (continued)

Base Case. \( \psi_0(x_1, \ldots, x_M, y_1, \ldots, y_M) \) is defined as a big quantifier-free formula in DNF where each disjunct represents a valid combination of values for \((x_1, \ldots, x_M)\) and \((y_1, \ldots, y_M)\), i.e., either they represent the same configuration or they correspond to the transition of \( T \) in one step. For every \( \ell \in \{1, \ldots, M - 1\} \), \( \psi_0 \) thus contains disjuncts:

\[
D = \Sigma(x_1) \land \cdots \land \Sigma(x_{\ell}) \land K(x_{\ell+1}) \land \Sigma(x_{\ell+2}) \land \cdots \land \Sigma(x_M) \land
x_1 = y_1 \land \cdots \land x_M = y_M.
\]

For each transition \((a, a', b, \rightarrow)\) in \( \delta \), \( \psi_0 \) contains the following disjuncts (cursor movements and \( \leftarrow \) are treated analogously):

\[
D = \Sigma(x_1) \land x_1 = y_1 \land \cdots \land \Sigma(x_{\ell-1}) \land x_{\ell-1} = y_{\ell-1} \land
x_{\ell} = a \land x_{\ell+1} = q \land \Sigma(x_{\ell+2}) \land
y_{\ell} = b \land y_{\ell+1} = x_{\ell+2} \land y_{\ell+2} = q' \land
\Sigma(x_{\ell+3}) \land x_{\ell+3} = y_{\ell+3} \land \cdots \land \Sigma(x_M) \land x_M = y_M.
\]
PSPACE-Membership of First-Order Queries

Proof of the PSPACE-Membership (continued)

3. Replacing first-order variables by propositional variables. The only atoms occurring in ϕ′′ are equality atoms s = t, where the terms s, t are either variables (which can take the value 0 or 1) or the constants 0, 1. We identify 0 with the truth value false and 1 with the truth value true. Then we can transform ϕ′′ into the QSAT formula ψ by replacing the equality atoms by “equivalent” propositional formulae in the obvious way:

\[ x = y \iff x \iff y \]
\[ x = 0, 0 = x \iff \neg x \]
\[ 0 = 1, 1 = 0 \iff \text{false} \]
\[ x = 1, 1 = x \iff x \]
\[ 0 = 0, 1 = 1 \iff \text{true} \]

Clearly, ϕ evaluates to true over D ⇔ ϕ′ evaluates to true over D′ ⇔ ϕ′′ evaluates to true independently of any database ⇔ ψ is true.

Conjunctive Queries

Definition

Conjunctive queries (CQs) are a special case of first-order queries whose only connective is ∧ and whose only quantifier is ∃ (i.e., ∀, ¬ and ∨ are excluded). Alternatively, CQs can be considered as a single datalog rule

\[ Q : r(u) \iff r_1(u_1) \land \ldots \land r_n(u_n) \]

where \( n \geq 0 \); \( r_1, \ldots, r_n \) are (not necessarily distinct) extensional relation symbols and \( u, u_1, \ldots, u_n \) are lists of terms of appropriate length. Moreover, all variables in \( u \) occur in at least one \( u_i \).

In a Boolean conjunctive query, the head of the rule \( Q \) is the 0-ary intensional relation symbol true() (rather than some arbitrary term \( r(u) \)).

Remark. Conjunctive queries correspond to select-project-join queries in the relational algebra, i.e., unnested select-from-where queries in SQL.

Discussion

Easy Consequences

PSPACE-hardness of QSAT. The above proof of the PSPACE-hardness of FO evaluation together with the above reduction from FO evaluation to QSAT immediately yields the PSPACE-hardness of QSAT.

Narrowing FO evaluation and PSPACE-hardness.

- The first 2 steps in the above reduction from FO evaluation to QSAT allowed us to transform an arbitrary FO formula ϕ over a database with arbitrary finite domain into an FO formula ψ over the domain \{0, 1\}, s.t. the atomic formulas of ψ are equalities only. Moreover, negation can be shifted immediately in front of the equalities.

- Equalities and negated equalities over \{0, 1\} can be represented by relations eq and noteq in the obvious way (this works for any finite domain), i.e., eq = {(0, 0), (1, 1)} and noteq = {(0, 1), (1, 0)}.

- It follows that FO evaluation remains PSPACE-hard even if we disallow equalities and negation in the FO formulas.

Conjunctive Queries

Theorem

The query complexity and the combined complexity of conjunctive queries are NP-complete.

Proof

NP-Membership (of the combined complexity). For each variable \( u \) of the query, we guess a domain element to which \( u \) is instantiated. Then we check whether all the resulting ground atoms in the query body exist in \( D \). This check is obviously feasible in polynomial time.

Hardness (of the query complexity). We reduce the NP-complete 3-Colorability problem to our problem. For this purpose, we consider an input database over the binary relation symbol Edge.
NP-Hardness of query complexity

Since we are considering the query complexity, the database $D$ is fixed (but arbitrarily chosen). We choose $D$ with a single relation $Edge = \{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\}$

Now let $G = (V, E)$ be an arbitrary instance of the 3-Colorability problem. From this, we define the Boolean conjunctive query $Q$ as follows. $Q$ contains the variables $X = \{x_i | v_i \in V\}$. Moreover, we set

$$ans() \leftarrow \bigwedge_{[v_i, v_j] \in E} Edge(x_i, x_j)$$

Clearly, this reduction is feasible in logarithmic space. The correctness is seen as follows: $Q$ is true over the DB $D$ $\Leftrightarrow$ The variables in $X$ can be instantiated to values $\{1, 2, 3\}$, s.t. $Q$ contains only ground atoms occurring in $D$ $\Leftrightarrow$ The graph $G$ has a valid 3-coloring.

Learning Objectives

- The power of unrestricted alternation (in QBF)
- PSPACE as the complexity class of many strategy games
- The relationship of PSPACE and PH
- Complexity of query evaluation, first-order queries