**Complexity Theory**

*Polynomial Space*

Daniel Borchmann, Markus Krötzsch

Computational Logic

2015-12-01

©️♥️️©️️

---

**The Class PSpace**

We defined PSpace as:

\[
PSPACE = \bigcup_{d \geq 1} DSPACE(n^d)
\]

and we observed that

\[P \subseteq NP \subseteq PSPACE = NPSPACE \subseteq \text{ExpTime}.\]

We can also define a corresponding notion of PSpace-hardness:

**Definition 11.1**

- A language \( \mathcal{H} \) is PSpace-hard, if \( \mathcal{L} \leq_p \mathcal{H} \) for every language \( \mathcal{L} \in PSPACE. \)
- A language \( \mathcal{C} \) is PSpace-complete, if \( \mathcal{C} \) is PSpace-hard and \( \mathcal{C} \in PSPACE. \)

---

**Quantified Boolean Formulae (QBF)**

A QBF is a formula of the following form:

\[O_1X_1.O_2X_2.\ldots.O_{\ell}X_{\ell}.\varphi[X_1,\ldots,X_{\ell}]\]

where \( O_i \in \{\exists, \forall\} \) are quantifiers, \( X_i \) are propositional logic variables, and \( \varphi \) is a propositional logic formula with variables \( X_1,\ldots,X_{\ell} \) and constants \( \top \) (true) and \( \bot \) (false)

**Semantics:**

- Propositional formulae without variables (only constants \( \top \) and \( \bot \)) are evaluated as usual
- \( \exists X.\varphi[X] \) is true if either \( \varphi[X/\top] \) or \( \varphi[X/\bot] \) are true
- \( \forall X.\varphi[X] \) is true if both \( \varphi[X/\top] \) and \( \varphi[X/\bot] \) are true

(where \( \varphi[X/\top] \) is "\( \varphi \) with \( X \) replaced by \( \top \), and similar for \( \bot \)."

---

©️ Daniel Borchmann, Markus Krötzsch 2015
Deciding QBF Validity

**True QBF**

*Input:* A quantified Boolean formula $\varphi$.

*Problem:* Is $\varphi$ true (valid)?

**Observation**

We can assume that the quantified formula is in CNF or 3-CNF (same transformations possible as for propositional logic formulae).

Consider a propositional logic formula $\varphi$ with variables $X_1, \ldots, X_\ell$:

**Example 11.2**

The QBF $\exists X_1 \cdots \exists X_\ell \varphi$ is true if and only if $\varphi$ is satisfiable.

**Example 11.3**

The QBF $\forall X_1 \cdots \forall X_\ell \varphi$ is true if and only if $\varphi$ is a tautology.

---

**Solving True QBF in PSPACE**

```plaintext
01 TrueQBF(\varphi) {
02 if \varphi has no quantifiers :
03 return “evaluation of \varphi”
04 else if \varphi = $\exists X \psi$ :
05 return (TrueQBF(\psi[\varphi/\top]) OR TrueQBF(\psi[\varphi/\bot]))
06 else if \varphi = $\forall X \psi$ :
07 return (TrueQBF(\psi[\varphi/\top]) AND TrueQBF(\psi[\varphi/\bot]))
08 }
```

- Evaluation in line 03 can be done in polynomial space
- Recursions in lines 05 and 07 can be executed one after the other, reusing space
- Maximum depth of recursion = number of variables (linear)
- Store one variable assignment per recursive call

$\sim$ polynomial space algorithm

---

**The Power of QBF**

**Theorem 11.4**

*True QBF is PSpace-complete.*

**Proof.**

- **True QBF $\in$ PSpace:**
  
  Give an algorithm that runs in polynomial space.

- **True QBF is PSpace-hard:**
  
  Proof by reduction from the word problem for polynomially space-bounded TMs.

---

**PSpace-hardness of True QBF**

Express TM computation in logic, similar to Cook-Levin

**Given:**

- a polynomial $p$
- a $p$-space bounded 1-tape NTM $M = (Q, \Sigma, \Gamma, \delta, q_0, q_{accept})$
- a word $w$

**Intended reduction**

Define a QBF $\varphi_{p,M,w}$ such that $\varphi_{p,M,w}$ is true if and only if $M$ accepts $w$ in space $p(|w|)$.

**Note**

We show the reduction for NTMs, which is more than needed, but makes little difference in logic and allows us to reuse our previous formulae from Cook-Levin.
**Review: Encoding Configurations**

Use propositional variables for describing configurations:

- \( Q_q \) for each \( q \in Q \) means “\( M \) is in state \( q \in Q \)”
- \( P_i \) for each \( 0 \leq i < p(n) \) means “the head is at Position \( i \)”
- \( S_{a,i} \) for each \( a \in \Gamma \) and \( 0 \leq i < p(n) \) means “tape cell \( i \) contains Symbol \( a \)”

Represent configuration \((q, p, a_0 \ldots a_{p(n)})\)
by assigning truth values to variables from the set

\[
\overline{C} := \{ Q_q, P_i, S_{a,i} | q \in Q, a \in \Gamma, 0 \leq i < p(n) \}
\]

using the truth assignment \( \beta \) defined as

\[
\beta(Q_q) := \begin{cases} 
1 & \text{if } s = q \\
0 & \text{if } s \neq q 
\end{cases}, \\
\beta(P_i) := \begin{cases} 
1 & \text{if } i = p \\
0 & \text{if } i \neq p 
\end{cases}, \\
\beta(S_{a,i}) := \begin{cases} 
1 & \text{if } a = a_i \\
0 & \text{if } a \neq a_i 
\end{cases}
\]

**Review: Transitions Between Configurations**

Consider the following formula \( \text{NEXT}(\overline{C}, \overline{C}') \) defined as

\[
\text{CONF}(\overline{C}) \wedge \text{CONF}(\overline{C}') \wedge \text{NOCHANGE}(\overline{C}, \overline{C}') \wedge \text{CHANGE}(\overline{C}, \overline{C}').
\]

\[
\text{NOCHANGE} := \bigvee_{0 \leq p < p(n)} (P_p \wedge \bigwedge_{i \neq p, a \in \Gamma} (S_{a,i} \rightarrow S_{a,i}'))
\]

\[
\text{CHANGE} := \bigvee_{0 \leq p < p(n)} (P_p \wedge \bigwedge_{a \in \Gamma} (Q_q \wedge S_{a,p} \wedge \bigvee_{(q', b, D) \in \delta(q, a)} (Q_{q'} \wedge S_{b,p} \wedge P_{D(p)}')))
\]

where \( D(p) \) is the position reached by moving in direction \( D \) from \( p \).

**Review: Validating Configurations**

We define a formula \( \text{Conf}(\overline{C}) \) for a set of configuration variables

\[
\overline{C} = \{ Q_q, P_i, S_{a,i} | q \in Q, a \in \Gamma, 0 \leq i < p(n) \}
\]
as follows:

\[
\text{CONF}(\overline{C}) := \bigvee_{q \in Q} (Q_q \wedge \bigwedge_{q' \neq q} \neg Q_{q'})
\]

“TM in exactly one state \( q \in Q \)”

\[
\wedge \bigvee_{p < p(n)} (P_p \wedge \bigwedge_{p' \neq p} \neg P_{p'})
\]

“head in exactly one position \( p < p(n) \)”

\[
\wedge \bigvee_{0 \leq i < p(n)} \bigvee_{a \in \Gamma} (S_{a,i} \wedge \neg S_{b,i})
\]

“exactly one \( a \in \Gamma \) in each cell”

**Observations:**

- \( \text{conf}(\overline{C}, \beta) \) is a potential configuration of \( M \), but it may not be reachable from the start configuration of \( M \) on input \( w \).
- Conversely, every configuration \((q, p, w_0 \ldots w_{p(n)})\) induces a satisfying assignment \( \beta \) or which \( \text{conf}(\overline{C}, \beta) = (q, p, w_0 \ldots w_{p(n)}) \).

**Lemma 11.5**

If \( \beta \) satisfies \( \text{CONF}(\overline{C}) \) then \( |\text{conf}(\overline{C}, \beta)| = 1 \).
*We can therefore write \( \text{conf}(\overline{C}, \beta) = (q, p, w) \) to simplify notation.*
Simulating Polynomial Space Computations

For Cook-Levin, we used one set of configuration variables for every computing step: polynomially time \( \sim \) polynomially many variables.

Problem: For polynomial space, we have \( 2^{O(p(n))} \) possible steps . . .

What would Savitch do?

Define a formula \( \text{CanYield}_0( \overline{C}_1, \overline{C}_2) \) to state that \( \overline{C}_2 \) is reachable from \( \overline{C}_1 \) in at most \( 2^d \) steps:

\[
\text{CanYield}_0( \overline{C}_1, \overline{C}_2) := (\overline{C}_1 = \overline{C}_2) \lor \text{Next}(\overline{C}_1, \overline{C}_2)
\]

\[
\text{CanYield}_{i+1}( \overline{C}_1, \overline{C}_2) := \exists \overline{C}. \text{Conf}(\overline{C}) \land \text{CanYield}_i( \overline{C}_1, \overline{C}) \land \text{CanYield}_i( \overline{C}, \overline{C}_2)
\]

But what is \( \overline{C}_1 = \overline{C}_2 \) supposed to mean here? It is short for:

\[
\bigwedge_{q \in Q} Q^1_q \leftrightarrow Q^2_q \land \bigwedge_{0 \leq i < p(n)} P^1_i \leftrightarrow P^2_i \land \bigwedge_{a \in \Gamma, 0 \leq i < p(n)} S^1_{a,i} \leftrightarrow S^2_{a,i}
\]

Did we do it?

Note: we used only existential quantifiers when defining \( \varphi_{p,M,w} \):

\[
\text{CanYield}_0( \overline{C}_1, \overline{C}_2) := (\overline{C}_1 = \overline{C}_2) \lor \text{Next}(\overline{C}_1, \overline{C}_2)
\]

\[
\text{CanYield}_{i+1}( \overline{C}_1, \overline{C}_2) := \exists \overline{C}. \text{Conf}(\overline{C}) \land \text{CanYield}_i( \overline{C}_1, \overline{C}) \land \text{CanYield}_i( \overline{C}, \overline{C}_2)
\]

\[
\varphi_{p,M,w} := \exists \overline{C}_1, \exists \overline{C}_2. \text{START}_{M, w}(\overline{C}_1) \land \text{Acc-Conf}(\overline{C}_2) \land \text{CanYield}_{dp(n)}(\overline{C}_1, \overline{C}_2)
\]

where we select \( d \) to be the least number such that \( M \) has less than \( 2^{dp(n)} \) configurations in space \( p(n) \).

Lemma 11.7

\( \varphi_{p,M,w} \) is satisfiable if and only if \( M \) accepts \( w \) in space \( p(|w|) \).

Now that’s quite interesting . . .

\begin{itemize}
  \item With only (non-negated) \( \exists \) quantifiers, True QBF coincides with Sat
  \item Sat is in NP
  \item So we showed that the word problem for PSPACE NTMs to be in NP
\end{itemize}

So we found that \( \text{NP} = \text{PSPACE}! \)

Strangely, most textbooks claim that this is not known to be true . . . Are we up for the next Turing Award, or did we make a mistake?
### The Power of QBF

**Theorem 11.4**

True QBF is \( \text{PSpace-complete} \).

**Proof.**

- **True QBF is in \( \text{PSpace} \):**
  
  Give an algorithm that runs in polynomial space.

- **True QBF is \( \text{PSpace-hard} \):**
  
  Proof by reduction from the word problem for polynomially space-bounded TMs.

**Exercise:**

- Why can we just use \( dp(n) \) in the reduction? Don’t we have to compute it somehow? Maybe even in polynomial time?

---

### A More Common Logical Problem in \( \text{PSpace} \)

Recall standard first-order logic:

- Instead of propositional variables, we have atoms (predicates with constants and variables)
- Instead of propositional evaluations we have first-order structures (or interpretations)
- First-order quantifiers can be used on variables
- Sentences are formulae where all variables are quantified
- A sentence can be satisfied or not by a given first-order structure

**Problem:**

\[ \text{Input: A first-order sentence } \varphi \text{ and a finite first-order structure } I. \]

\[ \text{Problem: Is } \varphi \text{ satisfied by } I? \]
First-Order Logic is PSPACE-complete

Theorem 11.8
FOL Model Checking is PSPACE-complete.

Proof.
- FOL Model Checking ∈ PSPACE:
  Give algorithm that runs in polynomial space.
- FOL Model Checking is PSPACE-hard:
  Proof by reduction True QBF \leq_p FOL Model Checking.

Checking FOL Models in Polynomial Space (Sketch)

01 Eval(ϕ, I) {
02 switch (ϕ) :
03   case \( p(c_1, \ldots, c_n) \) : return \( \langle c_1, \ldots, c_n \rangle \in p^I \)
04   case \( \neg \psi \) : return NOT Eval(\( \psi \), I)
05   case \( \psi_1 \land \psi_2 \) : return Eval(\( \psi_1 \), I) AND Eval(\( \psi_2 \), I)
06   case \( \exists x. \psi \) :
07       for \( c \in \Delta^I \) :
08           if Eval(\( \psi[X_1 \mapsto c] \), I) : return TRUE
09       // eventually, if no success:
10       return FALSE
11 }

□

- We can assume \( \phi \) only uses \( \neg, \land \) and \( \exists \) (easy to get)
- We use \( \Delta^I \) to denote the (finite!) domain of \( I \)
- We allow domain elements to be used like constants in the formula

Hardness of FOL Model Checking

Given: a QBF \( \phi = Q_1 X_1. \cdots Q_\ell X_\ell. \psi \)

FOL Model Checking Problem:
- Interpretation domain \( \Delta^I := \{0, 1\} \)
- Single predicate symbol true with interpretation \( true^I = \{1\} \)
- FOL formula \( \phi' \) is obtained by replacing variables in input QBF with corresponding first-order expressions:
  \[ Q_1 x_1. \cdots Q_\ell x_\ell. \psi[x_1 \mapsto true(x_1), \ldots, x_\ell \mapsto true(x_\ell)] \]

Lemma 11.9
\( \langle I, \phi' \rangle \in \text{FOL Model Checking if and only if } \phi \in \text{True QBF}. \)

First-Order Logic is PSPACE-complete

Theorem 11.8
FOL Model Checking is PSPACE-complete.

Proof.
- FOL Model Checking ∈ PSPACE:
  Give algorithm that runs in polynomial space.
- FOL Model Checking is PSPACE-hard:
  Proof by reduction True QBF \leq_p FOL Model Checking.
FOL Model Checking: Practical Significance

Why is FOL Model Checking a relevant problem?

Correspondence with database query answering:
- Finite first-order interpretation = database
- First-order logic formula = database query
- Satisfying assignments (for non-sentences) = query results

Known correspondence:
As a query language, FOL has the same expressive power as (basic) SQL (relational algebra).

Corollary 11.10

Answering SQL queries over a given database is \( \text{PSPACE-complete} \).

Games as Computational Problems

Many single-player games relate to \( \text{NP} \)-complete problems:
- Sudoku
- Minesweeper
- Tetris
- . . .

Decision problem: Is there a solution?
(For Tetris: is it possible to clear all blocks?)

What about two-player games?
- Two players take moves in turns
- The players have different goals
- The game ends if a player wins

Decision problem: Does Player 1 have a winnings strategy?
In other words: can Player 1 enforce winning, whatever Player 2 does?

Games

Coming Up Next

How hard is it to determine if there is a winning strategy?
Which games should we study?
To be continued . . .