Handout

The Class PSPACE and Game Theory

Proseminar Theoretische Informatik WiSe 2020-21
Institut für Informatik
Freie Universität Berlin

valentinpi

5. Januar 2021
(neueste Version)

Abstract

This handout presents the complexity class PSPACE. It presents the most important definitions in the context of the language hierarchy after the P and NP presentations. It then presents its applications in the formula game and the geography game.

Space Complexity and the Class PSPACE One can analyze efficiency in at least three different ways: Time, space, encoding efficiency. Looking at space efficiency leads to the first complexity classes here:

Definition 1. Let $f\colon \mathbb{N} \to \mathbb{R}^+$ be a function. The complexity classes *SPACE* and *NSPACE* of f are:

$$\text{SPACE}(f(n)) \coloneqq \{\ L \mid L \text{ is decided by a deterministic turing machine using } \mathcal{O}(f(n)) \text{ space}\ \}$$
$$\text{NSPACE}(f(n)) \coloneqq \{\ L \mid L \text{ is decided by a nondeterministic turing machine using } \mathcal{O}(f(n)) \text{ space}\ \}$$

Instead of time, to analyze whether problems are in these classes, one has to consider how much space, i.e. memory or tape cells of a TM, an algorithm uses. As you can see, these definitions are analogous to the ones for TIME and NTIME.

Definition 2. In analogy to the classes P and NP, *PSPACE* and *NPSPACE* are defined as:

$$\text{PSPACE} \coloneqq \bigcup_{k \in \mathbb{N}} \text{SPACE}(n^k) \qquad \text{NPSPACE} \coloneqq \bigcup_{k \in \mathbb{N}} \text{NSPACE}(n^k)$$

One very important and surprising result of complexity theory is the following: The researcher Savitch showed that any nondeterministic polynomial space bounded machine can be simulated by such a deterministic machine in about quadratic more space. Since PSPACE $\subseteq$ NPSPACE per definition:

Fact 1 (Corollary from Savitchs Theorem)**.** PSPACE = NPSPACE

As in NP-completeness, completeness is also defined for PSPACE-Problems:

Definition 3. A language B is *PSPACE-complete*, if:

1. $B \in$ PSPACE
2. $A \leq_p B$ for all $A \in$ PSPACE, also called *PSPACE-hard*

- Complete problems exist for many complexity classes, not just NP
- $\leq_p$ denotes *polynomial time reducibility*, not space reducibility. Time is also a bound for the space used, since a machine can not occupy more space than it can move its read/write head.

Fact 2. The following holds:

$$\text{P} \subseteq \text{NP} \subseteq \text{PSPACE} = \text{NPSPACE} \subseteq \text{EXPTIME} \coloneqq \bigcup_{k \in \mathbb{N}} \text{TIME}\left(2^{n^k}\right),\ \text{P} \neq \text{EXPTIME}$$

Quantified Boolean Formulas The following is a generalization of the SAT-Problem with *quantifiers*. Some important terms:

- Any boolean formula can be put into *prenex normal form*: Quantifiers appear first and then the boolean formula without quantifiers. General form: $\mathrm{Q}_1 x_1 \mathrm{Q}_2 x_2 ... \mathrm{Q}_n x_n : \phi$ with $\mathrm{Q}_1, ..., \mathrm{Q}_n \in \{\forall, \exists\}$

  An example:

  $$\phi = \underbrace{\exists x_1 \forall x_2, x_3 :}_{\text{bounded variables}} \underbrace{(x_1 \vee x_2) \vee x_3}_{\text{scope}} \text{ is in prenex normal form}$$

- Such a formula is *fully quantified*, if each variable of the formula appears in the scope of a quantifier, like in the formula above.

- Fully quantified formulas are either true or false, since they are statements themselves.

This generalization of SAT is called *TBQF*: TQBF $\coloneqq \{ \langle \phi \rangle \mid \phi$ is a true fully quantified boolean formula $\}$

Theorem 1. TQBF is PSPACE-complete.

Proof. TQBF $\in$ PSPACE can be shown by giving a recursive algorithm. Since all the quantifiers are first and the variables can only get assigned to a finite number of values - true or false -, our algorithm can check all possible combinations of variable assignments:

Input: $\langle \phi \rangle$, where ϕ is a fully quantified boolean formula.

Function:

1: **if** $\phi = \psi$ has no quantifiers **then**
2: ψ has no quantifiers, only constants. Evaluate ψ, accept or reject depending on result.
3: **else if** ϕ has form $\exists x : \psi$ (ψ is remaining term) **then**
4: Recursive call on ψ, once with x substituted with false, once substituted with true.
5: If either accepts, accept. Otherwise, reject.
6: **else** ϕ has form $\forall x : \psi$
7: Recursive call on ψ, once with x substituted with false, once substituted with true.
8: If both accept, accept. Otherwise, reject.

The additional space the algorithm uses is limited by the number of recursions. Those can be at most n with n being the number of variables that appear in the formula. Therefore the algorithm runs in $\mathcal{O}(n)$ space, which is polynomial.

For the PSPACE-hardness we will only consider the proof idea here. The first idea would be, since this is a generalization of the SAT-problem, to consider the proof from Cook-Levin and simulate execution of the polynomial space bounded TM of some $A \in$ PSPACE using a fully quantified boolean formula.

However, such a TM for A could run in exponential time! So the reduction may produce a simulation, the quantified boolean formula, that is of exponential size. A polynomial time reduction can not produce an exponential sized result.
Using techniques from the Theorem of Savitch, one can use universal quantifiers $\forall$ to divide the formula for the TM into subformulas. So the idea is to use another technique to make the previous formula much smaller. ■

To prove PSPACE-completeness for more problems using TQBF, one can use this corollary from NP-completeness:

Corollary 1. If B is PSPACE-complete and $B \leq_p C$ for $C \in$ PSPACE, then C is PSPACE-complete.

Game Theory and the FORMULA-GAME Problem Here, a *game* is a competition between two opposing players, in which both want to achieve some goal under specific rules. There are casual games like chess and go, but also more serious games, which model war or societal conflict.

Fully quantified formulas decode games. One may recall this analogy from the logic lectures. Player A (*All*) and player E (*Exists*) take turns choosing assignments for the universally ($\forall$) and existentially ($\exists$) quantified variables of such a formula ϕ from left to right. If the formula evaluates to false, A wins. Otherwise, E wins. We call this game the *formula game*. One of these players has a *winning strategy*, if assignments can be chosen so that that player wins no matter what is chosen afterwards.

Example: $\exists x_1 \forall x_2, x_3 \colon x_1 \vee x_2 \wedge x_3$ - If E chooses $x_1 = 1$, the formula will evaluate to true and E wins, no matter what A chooses.

We define the following problem:

$$\text{FORMULA-GAME} \coloneqq \{\ \langle \phi \rangle \mid \text{Player E has a winning strategy in the formula game on } \phi\ \}$$

The following corollary shows that, even though the problems TQBF and FORMULA-GAME are seemingly different - one is about fully quantified boolean formulas and one is about gaming - they are actually the same.

Corollary 2. FORMULA-GAME = TQBF and FORMULA-GAME is PSPACE-complete.

Reason. ($\subseteq$) If player E has a winning strategy for the formula game associated with a fully quantified boolean formula ϕ, then, no matter how A chooses the variables bound to universal quantifiers to be assigned, E will win with the game with the strategy. Similarly, these assignments correspond to variable assignments for the existential quantifiers, and the fact that A can not prevent the loss means that the statement is true for any assignment of the universal quantifiers. Therefore, ϕ must be true, if E has such a strategy.
($\supseteq$) Similarly, if ϕ is a true formula, player E can win by using the variable assignments for variables bound to existential quantifiers as the winning strategy. The problems are identical.

The Game Geography and the GG Problem Another game is *geography*. In geography, two players I and II take turns naming cities from all around the world. The first city is chosen in advance. Player I must then choose a city that starts with the last letter of the first city, no repetitions allowed. Then player II chooses a next city and so on. If one player runs out of city names, the other player wins.

Example: A game starts with Paris. Player I chooses St. Malo, since it starts with S. Player II chooses Orleans, then I chooses Strasbourg. Continue with Grenoble and Epernay. Player II, in this case, does not know any city starting with Y, so I has won this game.

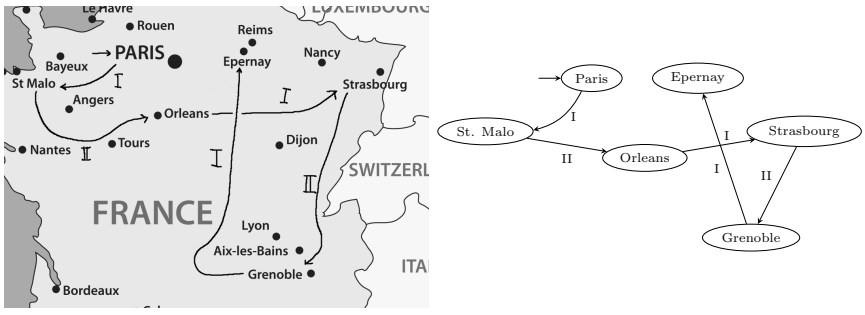


Figure 1: A game of geography on the map of France.[1]

[1] https://i.pinimg.com/originals/f6/e6/c5/f6e6c586d8b62dc6e66fa2e85dad2ad8.png, last accessed: 27.12.2020, 22:30

This example only used French cities, but in geography one generally talks about cities from all around the world. The cities named induce a directed graph over the map of the world.
One can generalize this problem for any directed graph with a start node, called *generalized geography* (GG):

$$\text{GG} \coloneqq \{\ \langle G, s\rangle \mid \text{Player I has a winning strategy for the GG game played on } G, \text{ starting from } s\ \}$$

Theorem 2. GG is PSPACE-complete.

Proof. The proof uses the Corollary 1. First, show that GG $\in$ PSPACE, then show FORMULA-GAME $\leq_p$ GG. The following algorithm decides GG:

Input: $\langle G, s\rangle$, where G is a directed graph and s a node from G.

Function:

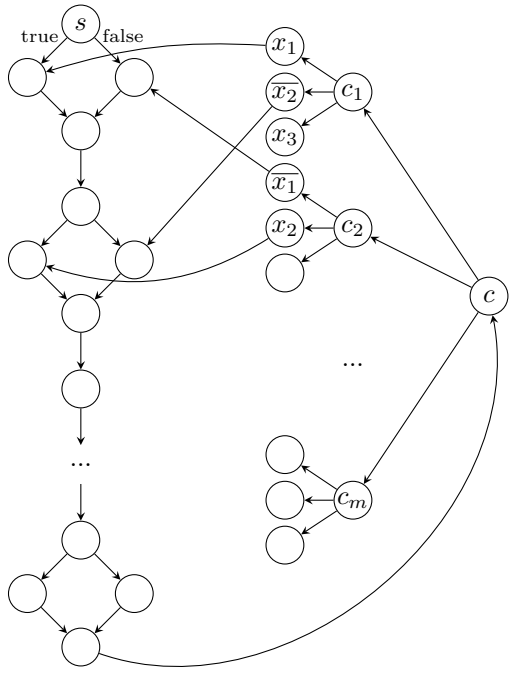
1: **if** $\deg s = 0$ (the degree deg is the number of neighbors of a node) **then**
2: Reject, since Player I instantly looses.
3: Remove s and its edges from G for a new graph G'.
4: For the outgoing neighbors of s, $s_1, ..., s_k$, do a recursive call on $\langle G', s_i\rangle$, $i \in \{1, ..., k\}$.
5: If all accept, reject, since the current player would lose. Otherwise, accept, since the opposing player gets stuck.

During execution, the algorithm implicitely switches perspective from player I to II and vice versa when it recurses. When we reject, the current player loses due to the first if-statement. When we accept, we win.
About the complexity: The machine only needs to store each node when it recurses into the neighbors. Therefore the space complexity is $\mathcal{O}(n)$ with n being the number of nodes, which is polynomial.

Now for the second statement. Let

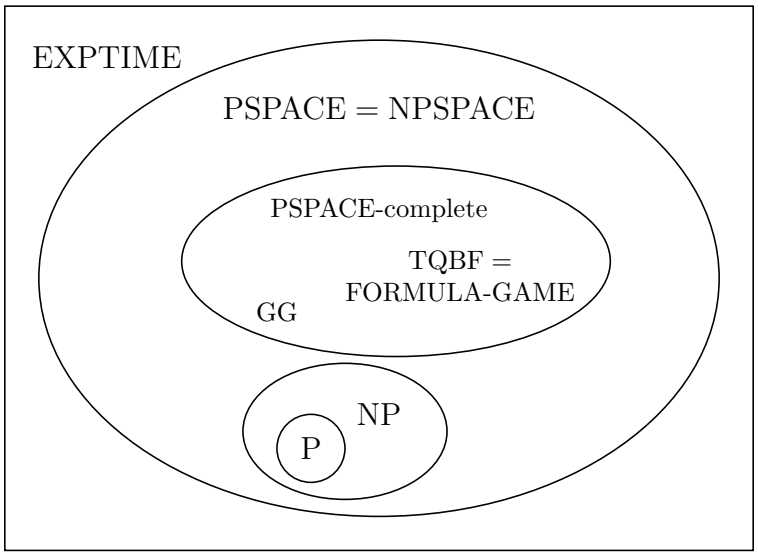
$$\phi = \exists x_1 \forall x_2 \exists x_3 ... \exists x_n : \psi$$

be a formula game with ψ in conjunctive normal form. The quantifiers start and end with existential ones and strictly alternate. If not, insert quantifiers for dummy variables, which are not used in ψ.
An example for the constructed geography game is demonstrated on the right side. For each variable, a diamond-shaped structure is introduced, with which the players implicitely assign truth values to variables. Using dummy nodes between each diamond, the players take their turns choosing. When x_n has been assigned by player I, an assignment of all variables was made. Player II reaches the node c, which represents a choice between the clauses of the formula.
For each clause and each literal, we introduce nodes. ($\Rightarrow$) If the player I has a winning strategy for ϕ, player II can choose any clause of $c_1, c_2, ..., c_m$. Player I then must choose one true literal from the clause, which exists, since the formula is satisfied. The literal only has an edge back to the choice made, but it has already been made, since the literal is true. Player II is stuck and player I wins. If player II has a winning strategy, one node was not visited and II can choose the literal for it. Player I would then get stuck. ($\Leftarrow$) The assignment made directly corresponds to a winning strategy in the formula game, which completes the reduction.



■

Another New Landscape With the introduction of PSPACE, NPSPACE but also EXPTIME the landscape has become larger. Assuming the relations are proper (real subsets, P $\neq$ NP, etc.), from Fact 2 one gets:



All the illustrations were made by myself using *LaTex/Tikz*.