Chapter 10

Polynomial Space

10.1. Problems Complete for $\mathcal{PSPACE}$

In this section we consider a problem analogous to SAT that is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{PSPACE}$. The proofs are almost identical to those in Section 8.1.

Definition 10.1: A *quantified Boolean formula* has the form

$$(\exists y_1)(\forall y_2)(\exists y_3)\cdots(\forall y_{2k})\ F(y_1, y_2, \ldots, y_{2k}),$$

where $F(y_1, y_2, \ldots, y_{2k})$ is a propositional formula with variables $y_1, y_2, \ldots, y_{2k}$. The quantification is over $y_i \in \{0,1\}$, for all $1 \leq i \leq 2k$.

Definition 10.2: QBF is the set of true quantified Boolean formulas, where the propositional part F is in conjunctive normal form with at most three literals per clause.

Notice that $3SAT$ is the same as QBF, except that all quantifiers are existential in $3SAT$. This analogy suggests that QBF should be complete for alternating polynomial time, i.e., $\mathcal{PSPACE}$.

Theorem 10.3 (Stockmeyer and Meyer [44, 43]): QBF is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{PSPACE}$.

Proof:

1. $QBF \in \mathcal{PSPACE}$: It suffices to show that $QBF \in \text{ATIME}\left(n^{O(1)}\right)$. Given a quantified Boolean formula Q with $2k$ variables, use the alternation to mimic the quantifiers of Q while choosing and recording a truth assignment $A \in \{0,1\}^{2k}$. Deterministically evaluate Q at this assignment A, and accept if and only if Q evaluates to true.

2. QBF is $\leq_m^{\mathcal{L}}$–hard for $\mathcal{PSPACE}$: Let A be an alternating Turing machine that runs in polynomial time. The reduction of $L(A)$ to QBF is exactly as in Theorems 8.4 and 8.7, with the following changes:

1. In the proof of Theorem 8.4, assume without loss of generality that A alternates between existential and universal states each step, adding dummy states if necessary. A is then put

in the normal form of alternating when writing y, and then simulating some deterministic Turing machine D on input $x\#y$. The language to which $L(A)$ is reduced is the "quantified circuit" problem:

$$\{C \mid (\exists b_1)(\forall b_2)(\exists b_3)\cdots(\forall b_{2k})(C \text{ outputs 1 on input } (b_1, b_2, \ldots, b_{2k}))\}.$$

2. In the reduction of Theorem 8.7, add a quantifier $(\exists x_w)$ to the end of the quantifier list presented in the quantified circuit input, for each gate w. In order to preserve the alternation of quantifiers, add dummy universally quantified variables.

□

Using similar techniques, Meyer and Stockmeyer [30, 43] presented analogous problems complete for each Σ_k^P and Π_k^P.

The richest source of natural problems complete for $\mathcal{PSPACE}$ has come from two-person games, and was exposed by Schaefer [42]. For a large variety of games, he showed that the problem of deciding if Player 1 has a winning strategy, starting from some given configuration of the game, is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{PSPACE}$. To see why this might be expected, notice that the alternation inherent in these questions may be expressed informally as follows: Is there a move for Player 1 such that, for all next moves by Player 2, there exists a next move by Player 1 such that ... the result is a configuration in which Player 1 has won the game?

An example of one of Schaefer's games is "generalized geography", which is played on a directed graph G with a distinguished start vertex s, as follows. Player 1 begins with s as the current vertex. The players take turns removing any edge (u, v) emanating from the current vertex u, after which v becomes the current vertex. The first player with no move remaining loses.

This game generalizes the children's game of "geography", in which there is a vertex for each of the 26 letters of the alphabet, and an edge (u, v) for every country of the world that begins with the letter u and ends with the letter v.

Definition 10.4: *GEO* is the set of pairs (G, s) such that Player 1 has a winning generalized geography strategy on G starting at s.

Theorem 10.5 (Schaefer [42]): *GEO* is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{PSPACE}$.

Finally, the existence of problems that are complete for $\mathcal{PSPACE}$ makes it very unlikely that $\mathcal{PSPACE} = \mathcal{PH}$:

Theorem 10.6: If $\mathcal{PSPACE} = \mathcal{PH}$, then $\mathcal{PH} = \Sigma_k^P$ for some constant k.

Proof: If $\mathcal{PSPACE} = \mathcal{PH}$, then $QBF \in \Sigma_k^P$ for some k. Since QBF is $\leq_m^{\mathcal{L}}$–hard for $\mathcal{PSPACE}$ and Σ_k^P is closed under $\leq_m^{\mathcal{L}}$, $\mathcal{PH} = \mathcal{PSPACE} \subseteq \Sigma_k^P$. □

10.2. A Lower Bound for Problems Complete for $\mathcal{PSPACE}$

One feature of problems complete for $\mathcal{PSPACE}$ is that we can prove an interesting unconditional lower bound on their space complexity. "Unconditional" here means that the theorem does not rely on any unproven hypothesis such as $\mathcal{P} \neq \mathcal{NP}$.

Theorem 10.7: If L is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{PSPACE}$ then, for some $\delta > 0$, $L \notin \text{DSPACE}(n^\delta)$.

Proof: By Theorem 3.8, there is some language $A \in \mathcal{PSPACE} - \text{DSPACE}(n)$. Since L is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{PSPACE}$, there is some deterministic Turing machine F that reduces A to L and runs in $O(\log n)$ space. Let f be the function computed by F. Because F runs in $O(\log n)$ space, there is a constant $k > 0$ such that $|f(x)| \leq |x|^k$ for all sufficiently long x.

Now suppose that $\delta > 0$ is any constant such that L is accepted by some deterministic Turing machine D that runs in space $O(n^\delta)$. By using the method in the proof of Lemma 5.3, the deterministic Turing machines F and D can be composed to produce a deterministic Turing machine that accepts A in space $O(\log|x| + \log|f(x)| + |f(x)|^\delta) = O(\log n + \log(n^k) + (n^k)^\delta) = O(n^{k\delta})$. Since, by the definition of A, $A \notin \text{DSPACE}(n)$, it must be the case that $\delta > 1/k$; for any $\delta \leq 1/k$, then, $L \notin \text{DSPACE}(n^\delta)$. □

A similar lower bound holds, in fact, for nondeterministic Turing machines:

Corollary 10.8: If L is $\leq_m^{\mathcal{L}}$–complete for $\mathcal{PSPACE}$ then, for some $\delta > 0$, $L \notin \text{NSPACE}(n^\delta)$.

Proof: By Corollary 4.15, if $L \notin \text{DSPACE}(n^\delta)$, then $L \notin \text{NSPACE}(n^{\delta/2})$. □