

CMPT 710 - Complexity Theory: Lecture 13

Valentine Kabanets

October 23, 2007

1 PSPACE-completeness of TQBF

Theorem 1. *TQBF is PSPACE-complete.*

Proof. (1) $TQBF \in PSPACE$: we already showed that above.

(2) $TQBF$ is PSPACE-hard. We need to reduce every language in PSPACE to TQBF. According to our definition of TQBF, the inner formula must be in CNF. However, it will be easier for us to reduce every language in PSPACE to a quantified formula in DNF (disjunctive normal form). This turns out to be sufficient since PSPACE is closed under complementation. (Check this.)

Let L be any language in PSPACE. Say L is decided by some TM M in $Space(n^c)$ for some constant c . To decide if $x \in L$, we consider the configuration graph of M on x . Each configuration will be of size $O(n^c)$; let us denote this size by m . There are at most 2^m different configurations. So, $x \in L$ iff there is a path from the start configuration to the accept configuration of length at most 2^m . We will construct a QBF that will express the existence of such a path.

In a sense, our proof is a restatement of the proof of Savitch's Theorem. Recall the function $Path(a, b, i)$ that we used in the proof of Savitch's Theorem: $Path(a, b, i)$ is True iff there is a path from node a to node b of length at most 2^i . We will define a sequence of QBFs ψ_i , $i = 0, \dots, m$, where $\psi_i(A, B)$ is True iff the variables A and B encode two configurations a and b such that b is reachable from a in at most 2^i steps; note that A and B are groups of m variables each. The required QBF will then be $\psi_m(Start, Accept)$, where $Start$ and $Accept$ are the binary strings encoding the start configuration and accept configuration, respectively.

For $i = 0$, $\psi_0(X, Y)$ is a quantified Boolean formula on free variables X and Y that is True iff either $X = Y$ or Y is reachable from X in 1 step. The formula $\psi_0(X, Y)$ can be written as a DNF of size $\text{poly}(m)$. (*Exercise:* Explain why.)

Given $\psi_i(X, Y)$, we can try to define $\psi_{i+1}(X, Y) = \exists Z[\psi_i(X, Z) \wedge \psi_i(Z, Y)]$. However, this is a *bad* idea: the size of ψ_{i+1} doubles, and so, the size of ψ_m would be exponential. The trick is to "re-use" the formula ψ_i . Here is a correct definition of $\psi_{i+1}(X, Y)$:

$$\exists Z_{i+1} \forall X_{i+1} \forall Y_{i+1} [((X_{i+1} = X \wedge Y_{i+1} = Z_{i+1}) \vee (X_{i+1} = Z_{i+1} \wedge Y_{i+1} = Y)) \Rightarrow \psi_i(X_{i+1}, Y_{i+1})]$$

Note how the formula above is True iff there is a Z_{i+1} such that both $\psi_i(X, Z_{i+1})$ and $\psi_i(Z_{i+1}, Y)$.

The rest of the proof takes care of technical details. First, note that in our definition of ψ_{i+1} all the quantifiers of the subformula ψ_i can be taken to the front, immediately after the quantifiers $\exists Z_{i+1} \forall X_{i+1} \forall Y_{i+1}$. Secondly, we need to turn ψ_{i+1} into a DNF. Since, by the inductive hypothesis, ψ_i is already a DNF, it suffices to turn into a DNF the subformula $\phi = \neg[(X_{i+1} = X \wedge Y_{i+1} = Z_{i+1}) \vee (X_{i+1} = Z_{i+1} \wedge Y_{i+1} = Y)]$. Let us denote the subformulas of ϕ by ϕ_1, \dots, ϕ_4 so that $\phi = \neg[(\phi_1 \wedge \phi_2) \vee (\phi_3 \wedge \phi_4)]$. By simple logical transformations, the equivalent DNF is

$$(\neg\phi_1 \wedge \neg\phi_3) \vee (\neg\phi_1 \wedge \neg\phi_4) \vee (\neg\phi_2 \wedge \neg\phi_3) \vee (\neg\phi_2 \wedge \neg\phi_4).$$

Each of the four subformulas can be expressed as a DNF over the variables of ϕ_i 's. For example, the first subformula $(\neg\phi_1 \wedge \neg\phi_3)$ is

$$\bigvee_{1 \leq s, t \leq m} \alpha_s \wedge \alpha_t,$$

where α_s expresses the fact that the strings X_{i+1} and X differ in position s , and α_t expresses the fact that the strings X_{i+1} and Z_{i+1} differ in position t . Finally, to express that two strings $x_1 \dots x_m$ and $y_1 \dots y_m$ differ in position s , we can use the DNF $(x_s \wedge \neg y_s) \vee (\neg x_s \wedge y_s)$.

We conclude by observing that the formula ψ_m can be constructed in logspace, since ψ_m has a very regular structure; the details are left as an exercise. \square

A few other two-person games (e.g., Go and checkers) are also known to be PSPACE-complete (see Papadimitriou's book).

2 On problems in NP that are neither in P nor NP-complete

From what you've seen so far, you may get an impression that every problem in NP is either in P or NP-complete. For the problems we've seen in this course so far, that happened to be the case, but it is not true in general. There is the following very nice result of Ladner.

Theorem 2 (Ladner's Theorem). *If $P \neq NP$, then there are problems in NP that are neither in P nor NP-complete.*

Thus, in general, there must be problems in NP that are not in P and not NP-complete (unless $NP = P$ and then everything in NP is in P). Are there any natural candidates? Probably the best candidate is the problem Graph Isomorphism, which asks if two given graphs are isomorphic. It is in NP since we can nondeterministically guess an isomorphism and then deterministically check in polytime if it is indeed an isomorphism. It is open if the problem is in P (except for some special cases); it is not known if the problem can be solved by a polytime quantum algorithm, despite considerable interest to this question from the quantum complexity community.

On the other hand, there is some evidence that Graph Isomorphism cannot be NP-complete. If it is, then something unlikely will happen: the polytime hierarchy would collapse.

We'll talk about the polytime hierarchy next.