CS 6743 **Lecture 14**[1] **Fall 2007**

1 PSPACE-Completeness

Recall the NP-complete problem SAT: Is a given Boolean formula $\phi(x_1,\dots,x_n)$ satisfiable? The same question can be stated equivalently as: Is the formula $\exists x_1 \dots \exists x_n \phi(x_1,\dots,x_n)$ true? Also recall the coNP-complete problem TAUT: Is a given Boolean formula $\psi(x_1,\dots,x_n)$ a tautology (i.e., true on all assignments)? This can be stated as: Is the formula $\forall x_1 \dots \forall x_n \psi(x_1,\dots,x_n)$ true?

What if we start alternating the quantifiers? Let us define the language TQBF (True Quantified Boolean Formulas) as the set of true formulas of the form

$$Q_1 x_1 Q_2 x_2 \dots Q_n x_n \phi(x_1, x_2, \dots, x_n),$$

where $Q_i \in \{\exists . \forall\}$ is a quantifier, ϕ is a Boolean formula in CNF (conjunctive normal form), and the sequence of quantifiers $Q_1 \dots Q_n$ alternates between $\exists$ and $\forall$. For example, all odd Q_{2k+1} can be $\exists$, and all even Q_{2k} can be $\forall$.

We can assume, without loss of generality, that $Q_1 = \exists$: if not, then we can always add an $\exists y$ before the formula for some variable y that does not occur in the formula. In this case, one can think of a QBF as a *game* between two players: Player $\exists$ and Player $\forall$. First, Player $\exists$ sets the value of x_1. Then Player $\forall$ sets the value of x_2. Then Player $\exists$ sets the value of x_3, and so on. After n moves, the winner is declared using the following criterion: Player $\exists$ wins iff the assignments to $x_1 \dots x_n$ constructed during the game is *satisfying* for the formula ϕ. It is easy to see that a given QBF is true iff Player $\exists$ has a *winning strategy*, i.e., for any choice of moves by Player $\forall$, Player $\exists$ can play so as to guarantee its win.

How can we decide if a given QBF is true? The following simple recursive algorithm *Truth* does the job.

```
Algo Truth($\Phi$)
if $\Phi$ is quantifier-free then return its value
end if
Let $\Psi = Q_1 x_1 \dots Q_n x_n \phi(x_1,\dots,x_n)$.
$b_0 = Truth(Q_2 x_2 \dots x_n \phi(0, x_2, \dots, x_n))$;
$b_1 = Truth(Q_2 x_2 \dots x_n \phi(1, x_2, \dots, x_n))$;% re-using space
if $Q_1 = \exists$ then return $b_0 \vee b_1$
  else return $b_0 \wedge b_1$
end if
end Algo
```

The algorithm *Truth* runs in polynomial space: the depth of the recursion is n, and the size of each stack record is poly(n).

[1]This lecture is a modification of notes by Valentine Kabanets

Remark: Using some tricks (as those we used to show that every $O(\log n)$-depth formula can be evaluated in $O(\log n)$-space), one can show that TQBF is in $\mathsf{Space}(n)$.

It turns out that PSPACE is probably the best possible we can do for TQBF, as it is PSPACE-complete.

2 PSPACE-completeness of TQBF

Theorem 1. *TQBF is* PSPACE*-complete.*

Proof. (1) $TQBF \in \mathsf{PSPACE}$: 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 $\mathsf{Space}(n^c)$ 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 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, $i = 0, \ldots, m$, 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 $\mathsf{poly}(m)$. (*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 $\phi_1, \ldots, \phi_4$ 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

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

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 $x_1 \ldots x_m$ and $y_i \ldots y_m$ 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. □

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

Definition 2. *A* combined complexity *of model-checking is the complexity of this problem when both the formula and the structure are given as an input (as opposed to the* data complexity *where the input is just the structure and the formula is hardcoded).*

Corollary 3. *The combined complexity of model-checking for first-order logic is PSPACE-complete.*

Proof. The hardness follows immediately from the completeness of TQBF, by setting the structure to have two elements and one relation differentiating between these elements ($X = \{1\}$). To show how to evaluate a first-order formula in PSPACE, note that the algorithm above for evaluating TQBF works for first-order formulas as well, since checking all possible values of an existential or a universal quantifier can be done reusing space. □

3 Polynomial-Time Hierarchy

Recall that TQBF talks about quantified Boolean formulas where the number of alternations is *non-constant*, e.g., $\exists x_1 \forall x_2 \ldots Q_n x_n \phi(x_1, \ldots, x_n)$ has n alternating quantifiers. Also recall that TQBF is PSPACE-complete. What happens if the number of alternating quantifiers is fixed to some constant? Then we obtain complete problems for the Polynomial-Time Hierarchy, defined below.

We call a k-ary relation R *polynomially balanced* if, for every tuple $(a_1, \ldots, a_k) \in R$, the lengths of all a_i's are polynomially related to each other.

Definition For any $i \geq 0$, a language $L \in \Sigma_i^p$ iff there is a polytime decidable, polynomially balanced $(i+1)$-ary relation R such that

$$L = \{x \mid \exists y_1 \forall y_2 \exists y_3 \ldots Q_i y_i R(x, y_1, \ldots, y_i)\}.$$

Here, Q_i is $\exists$ if i is odd, and $\forall$ if i is even.

For example, $\Sigma_0^p = \mathsf{P}$ and $\Sigma_1^p = \mathsf{NP}$.

Definition For any $i \geq 0$, a language $L \in \Pi_i^p$ iff there is a polytime, polynomially balanced $(i+1)$-ary relation R such that

$$L = \{x \mid \forall y_1 \exists y_2 \forall y_3 \ldots Q_i y_i R(x, y_1, \ldots, y_i)\}.$$

For example, $\Pi_0^p = \mathsf{P}$ and $\Pi_1^p = \mathsf{coNP}$.

Note that, in general, for every i, $\Pi_i^p = \mathsf{co}\Sigma_i^p$.

Definition $\mathsf{PH} = \cup_{i \geq 0} \Sigma_i^p$.

Theorem 4. $\mathsf{PH} \subseteq \mathsf{PSPACE}$

Proof. We know that even the general version of TQBF is in PSPACE. Hence, so is the version of TQBF with constant number of alternating quantifiers. $\square$

4 Examples of problems in PH

Unique-SAT $= \{\phi \mid \phi$ is a formula with exactly one satisfying assignment$\}$

Theorem 5. *Unique-SAT is in* Σ_2^p.

Proof. Note that $\phi \in \text{Unique} - \text{SAT}$ iff there is y such that for all z, $z \neq y$, we have $\phi(y)$ is True and $\phi(z)$ is False. $\square$

Min-Circuit $= \{C \mid C$ is a Boolean circuit s.t. no smaller equivalent circuit exists$\}$

Here, the size of a Boolean circuit is the number of logical operations (ANds, ORs, and NOTs), or gates, used in the circuit.

Theorem 6. *Min-Circuit is in* Π_2^p.

Proof. Note that C is in Min-Circuit iff for every smaller circuit C' there is an input x such that $C(x) \neq C'(x)$. $\square$

5 Alternative definition of PH

Definition An *oracle TM* is a TM M with special tape, called oracle tape, and special states $q_?, q_{yes}, q_{no}$. When run with some oracle O (where O is just some language), M can query O on some strings x by writing these x onto its oracle tape, and then entering the state $q_?$. In the next step, TM M (miraculously) finds itself in the state q_{yes} if $x \in O$, or the state q_{no} if $x \notin O$.

This definition of an oracle TM captures the notion of "having access to an efficient algorithm deciding O".

For complexity classes C_1 and C_2, we say that a language $L \in C_1^{C_2}$ if there is an oracle TM M from class C_1 that, given oracle access to some language $O \in C_2$, decides L.

For example, $Unique - SAT \in \mathsf{NP}^{\mathsf{NP}}$: Given a formula ϕ, nondeterministically guess an assignment a. Check that $\phi(a)$ is True. If not, then Reject; otherwise, construct a new formula $\phi'(x_1, \dots, x_n) \equiv$ "$\phi(x_1, \dots, x_n) \wedge [x_1 \dots x_n \neq a_1 \dots a_n]$". Ask the SAT oracle whether ϕ' is satisfiable. If it is, then Reject; otherwise, Accept. (Remark: With a more careful argument one can show that $Unique - SAT \in \mathsf{P}^{\mathsf{NP}}$, with only 2 queries to the SAT oracle. Do you see how?)

Alternative definition of PH. $\Sigma_0^p = \Pi_0^p = \mathsf{P}$. For all $i \geq 0$, $\Sigma_{i+1}^p = \mathsf{NP}^{\Sigma_i^p}$ and $\Pi_{i+1}^p = \mathsf{coNP}^{\Sigma_i^p}$. Finally, set $\mathsf{PH} = \cup_{i \geq 0} \Sigma_i^p$.

Theorem 7. *The original definition and the alternative definition of* PH *are equivalent.*

Proof. The base case of $i = 0$ is immediate: in both definitions, the 0th level is just the class P.

Just for the sake of this proof, let us denote by Σ_i^1 and by Σ_i^2 the ith level of polytime hierarchy according to definitions 1 and 2, respectively. (The first definition is in terms of logical formulas; the second definition is in terms of oracle TMs.)

We need to show that $\Sigma_i^1 = \Sigma_i^2$, for all i. The case of $i = 0$ is already argued. Let us assume the equivalence of the two definitions for i, and prove it for $i + 1$.

Let us start by proving that $\Sigma_{i+1}^1 \subseteq \Sigma_{i+1}^2$. By definition, $L \in \Sigma_{i+1}^1$ iff there is a poly-balanced relation R such that $x \in L \Leftrightarrow \exists y_1 \forall y_2 \dots R(x, y_1, y_2, \dots, y_{i+1})$. Consider the language $L' = \{(x, y) \mid \forall y_2 \dots R(x, y, y_2, \dots, y_{i+1})\}$. It is easy to see that $L' \in \Pi_i^1$, and hence, by the induction hypothesis, $L' \in \Pi_i^2$. Now, to test if $x \in L$ we can do the following: Nondeterministically guess a y, then check if $(x, y) \in L'$ by querying the Π_i^2 oracle. This algorithm shows that $L \in \Sigma_{i+1}^2$.

Let us now prove the other direction, i.e., that $\Sigma_{i+1}^2 \subseteq \Sigma_{i+1}^1$. Consider an arbitrary language $L \in \Sigma_{i+1}^2$. By definition, there is an $\mathsf{NP}^{\Sigma_i^2}$ TM M that decides L. Also, we have that $x \in L$ iff there is an accepting computation of M on x.

For any input x, consider a run of the TM M on x. During that computation, the TM M may ask (up to a polynomial number of) oracle queries to the Σ_i^2 oracle. Some of these oracle queries have the answer Yes, and the others No. Note that the Yes answers can be verified in Σ_i^2, which is equal to Σ_i^1, by the inductive hypothesis. The No answers can be verified in Π_i^2, which is equal to Π_i^1, by the inductive hypothesis.

Thus, to test if $x \in L$, we can guess (using the $\exists$ quantifier) an accepting computation path of M on x together with all answers to the oracle queries, and check the correctness of our path, including all the answers to the oracle queries, in $(\Sigma_i^1 \cup \Pi_i^1)$. Put together, this gives us a way to check whether $x \in L$ by a Σ_{i+1}^1 formula. Hence, we get $L \in \Sigma_{i+1}^1$.

Finally, since $\Pi_i = \mathsf{co}\Sigma_i$ for each of the two definitions, we immediately obtain the equality $\Pi_{i+1}^1 = \Pi_{i+1}^2$. □

6 Collapsing the Polynomial-Time Hierarchy

Many results in complexity theory have the form "Statement X is true, unless the polynomial-time hierarchy collapses". Since it is generally believed that all the levels of the polytime hierarchy are distinct, i.e., that the polytime hierarchy *does not collapse*, such results give us some evidence that the statement X is probably true.

Let us see under what conditions the polytime hierarchy would collapse to some finite level, i.e., $\mathsf{PH} = \Sigma_i^p$, for some constant i.

Theorem 8. *If* $\mathsf{NP} = \mathsf{coNP}$*, then* $\mathsf{PH} = \Sigma_1^p = \mathsf{NP} = \mathsf{coNP}$.

Proof. We want to show that $\Sigma_i^p = \Sigma_1^p$ for every i. Our proof is by induction. The base case of $i = 1$ is obvious. Suppose the truth of the statement for $i \geq 1$, and let us prove it for $i+1$.

Take any $L \in \Sigma_{i+1}^p$. By definition, there is an $(i+2)$-ary polytime polybalanced relation R such that $x \in L$ iff $\exists y_1 \forall y_2 \dots R(x, y_1, \dots, y_{i+1})$. Consider the language $L' = \{(x,y) \mid \forall y_2 \exists y_3 \dots R(x, y, y_2, \dots, y_{i+1})$. It is clear that $L' \in \Pi_i^p$. By the inductive hypothesis, we have $\Pi_i^p = \mathsf{co}\Sigma_i^p = \mathsf{co}\Sigma_1^p = \Pi_1^p$. On the other hand, our assumption is that $\mathsf{NP} = \mathsf{coNP}$, i.e., that $\Sigma_1^p = \Pi_1^p$. Thus, we obtain that $L' \in \Sigma_1^p$.

Finally, observe that $x \in L$ iff $\exists y\ (x,y) \in L'$. Since $L' \in \Sigma_1^p$, it follows that there is a polytime polybalanced relation R' such that $w \in L'$ iff $\exists z R'(w,z)$. Thus, $x \in L$ iff $\exists (y,z) R'((x,y),z)$. Therefore, $L \in \Sigma_1^p$. □

The theorem above can be easily generalized to show the following.

Theorem 9. *If* $\Sigma_i^p = \Pi_i^p$ *for some* $i \geq 1$*, then* $\mathsf{PH} = \Sigma_i^p$.

In other words, if the ith level of the polytime hierarchy (for $i \geq 1$) is closed under complement, then the polytime hierarchy collapses to the ith level. It is easy to see that the converse is also true.

Theorem 10. *If* $\mathsf{PH} = \Sigma_i^p$*, then* $\Sigma_i^p = \Pi_i^p$.

Proof. This follows from the fact that $\mathsf{PH} = \mathsf{coPH}$, i.e., the polytime hierarchy is closed under complement. (Remark: Note that this is different from saying that any *fixed level* Σ_i^p is closed under complement.) For any i, we have that $\Sigma_i^p \subseteq \Pi_{i+1}^p$, and $\Pi_i^p \subseteq \Sigma_{i+1}^p$. □

7 Oracles and unapplicability of diagonalization

It is a big open problem to find out whether PH collapses. A natural approach would be to try techniques that worked well before such as diagonalization. However, it is not possible to use diagonalization directly to prove that the polytime hierarchy is proper.

Definition 11. *We call a proof technique* relativizing *if the proof works in the presence of oracles.*

In particular, diagonalization is a relativizing technique, since the proof stays the same if we are diagonalizing against Turing machines with a given oracle.

Theorem 12 (Baker, Gill, Solovay). *There are two oracles A and B such that $P^A = NP^A$ and $P^B \neq NP^B$.*

Corollary 13. *No relativizing technique can be used to resolve P vs. NP question.*

Another result that could make it more believable that $P \neq NP$ is that $P \neq NP$ with probability 1 with respect to a random oracle. That is, although there is an oracle with respect to which they are the same, for an arbitrary oracles these classes are different. However, this does not mean that $P \neq NP$: there is an example of classes (IP and PSPACE) that were proven equal, although with respect to a random oracle they were distinct.