CS 6810: Theory of Computing **Fall 2023**

Lecture 9: Sept 19, 2023

Lecturer: Eshan Chattopadhyay *Scribe: Thomas Cui*

1 PSPACE completeness

Definition 1.1. *True quantified Boolean formula (TQBF) is the problem of determining the truth value of* $Q_1x_1, Q_2x_2, \ldots, Q_nx_n, \phi(x_1, x_2, \ldots, x_n)$*, where* $Q_1, \ldots, Q_n$ *are quantifiers (*$\forall$ *or* $\exists$*),* $x_1, \ldots, x_n$ *are variables, and* ϕ *is a Boolean formula.*

Theorem 1.2. *TQBF is PSPACE-complete.*

Proof. First notice a TM can brute force $TQBF$ and reuse space for each trial, so $TQBF \in PSPACE$.

Now we want to show $\forall L \in PSAPCE$, $L \leq_p TQBF$. We consider the following polynomial time reduction:

Let M be a TM that computes L in $S(n)$ space and x be an input to M. We know $x \in L$ if and only if there exists a path from v_{start} to v_{accept} in the configuration graph $G_{M,x}$. Note that there can be at most $2^{cS(n)}$ nodes in $G_{M,x}$ for some constant c. Denote $q = cS(n)$.

We define $\phi_i(A, B)$ to be 1 if there exists a path from A to B of length at most 2^i in $G_{M,x}$ and 0 otherwise. Then $\phi_q(v_{start}, v_{accept})$ is the final formula we want (since $G_{M,x}$ has at most 2^q nodes). A crucial observation is that there is path of length at most 2^i from A to B if and only if there exists a configuration $C \in V(G_{M,x})$ such that there are paths each of length at most 2^{i-1} from A to C and from C to B, so $\phi_i(A, B) = \exists C, (\phi_{i-1}(A, C) \wedge \phi_{i-1}(C, B))$.

However, the recursion resulted from the above formula is $T(i) = 2T(i-1) + O(S(n))$. When we unroll the recursion, we would have $T(q) = 2^{O(S(n))}$, which is not polynomial space.

To fix this issue, we introduce additional quantified variables and rewrite ϕ_i as follows

$$\phi_i(A, B) = \exists C, \forall D, \forall E, ((A = D \vee C = E) \wedge (C = D \vee B = E)) \implies \phi_{i-1}(D, E)$$

Therefore, our recursion becomes $T(i) = T(i-1) + O(S(n)) \implies T(q) = \text{poly}(S(n))$, and we have proved $L \leq_p TQBF$. □

2 Boolean Circuits

Now we switch topic to study Boolean circuits, which is a non-uniform model of computation/

Definition 2.1. *A Boolean circuit is a directed acyclic graph with 3 types of nodes:*

1. *Input nodes (fan-in), which have in-degree 0.*
2. *Output nodes (fan-out), which have out-degree 0.*
3. *Logical gates (*$\wedge, \vee, \neg$*), which are all other nodes.*

Definition 2.2. *There are 2 complexity measures for Boolean circuits:*

1. *Size: the number of edges (wires) in the circuit.*

2. *Depth: the length of the longest path from an input node to an output node.*

Given a Boolean circuit C with n input nodes, it naturally computes a Boolean function $f : \{0,1\}^n \to \{0,1\}$ (assuming 1 output node). To make circuits be able to compute languages (and take inputs of any length), we purpose the following definition.

Definition 2.3. *$\mathcal{C} := \{C_n\}_{n\in\mathbb{N}}$ is an $S(n)$-sized circuit family if $\forall C_n \in \mathcal{C}, |C_n| \leq S(n)$. We say $\mathcal{C}$ computes a language L if $\forall n \in \mathbb{N}$, $\forall x \in \{0,1\}^n$, $C_n(x) = L_n(x)$, where $L_n = L \cap \{0,1\}^n$*

Definition 2.4. *Define the complexity class $SIZE(S(n))$ such that a language is said to be in $SIZE(S(n))$ if there exists an $S(n)$-sized circuit family computing it.*

Definition 2.5. *We say a language L is in $P/poly$ if there exists a circuit family $\mathcal{C}$ computing L. Then $P/poly = \bigcup_{c\geq 1} SIZE(n^c)$*

It is worth mentioning the following claims, where **Claim 2.6** is known, and **Claim 2.7** is widely believed to be true.

Claim 2.6. *$P \subseteq P/poly$.*

Claim 2.7. *$NP \nsubseteq P/poly$.*

Also notice that **Claim 2.7** is equivalent to the following claim.

Claim 2.8. *There is no polynomial sized circuit family that computes SAT.*

The complexity class $P/poly$ in fact contains undecidable problems. Here is an example.

Definition 2.9. *A unary language is a subset of $\{1^n | n \geq 0\}$.*

Claim 2.10. *(Unary-)$HALT$ is in $P/poly$.*

To prove this claim, we prove a more general claim.

Claim 2.11. *Every unary language is in $P/poly$.*

Proof. We prove by constructing a polynomial sized circuit family that computes a unary language L. Let k be an arbitrary input length.

- If $1^k \in L$, set $C_k = x_1 \wedge \ldots \wedge x_k$.
- If $1^k \notin L$, set $C_k = 0$.

Obviously, this circuit family is polynomial sized, and it computes L. $\square$

2.1 Circuit lower/upper bounds

Theorem 2.12 (Lower bound)**.** *$\exists c, \forall n \geq n_0, \exists f_n : \{0,1\}^n \to \{0,1\}$ such that no circuit of size $\frac{2^n}{cn}$ can compute f_n.*

Proof. We prove by a counting argument: we will show that there are more such f_n's than circuits of size $\frac{2^n}{cn}$.

First notice that $|f_n| = 2^{2^n}$, i.e. there are 2^{2^n} such functions.

We now look at the number of circuits of size $\leq S$. By definition, there are at most S wires in the circuit, so each wire can be encoded as a bit string of length $2 \log S$, so a circuit can be represented in $c'S\log S$ bits for some constant c', implying there are at most $2^{c'S\log S} = S^{c''S}$ many circuits.

Pick $c = 2c''$. Plugging in $S = \frac{2^n}{cn}$, we have

$$\begin{aligned} S^{c''S} &= \left(\frac{2^n}{cn}\right)^{c''\frac{2^n}{cn}} \\ &= \frac{2^{2^{n-1}}}{(cn)^{c''S}} \end{aligned}$$

Which is much less than 2^{2^n} for large enough n, so there must be some f_n that cannot be computed by a circuit of size at most $\frac{2^n}{cn}$. □

In fact we can get something stronger: appropriately setting c, it can be shown that a random function requires circuits of size $\geq 2^n/cn$ with probability $1 - o(1)$.

Theorem 2.13 (Upper bound)**.** *$\exists c'$ such that any $f_n : \{0,1\}^n \to \{0,1\}$ can be computed by a circuit of size $\frac{2^n}{c'n}$.*

Proof. We provide a start of the proof here.

We can create the truth table of f_n and look at the DNFs (disjunctive normal forms) of f_n. Anding the variables in each DNF (can be done in $O(n2^n)$ wires) and ORing the results (can be done in $O(2^n)$ wires) gives a circuit of size $O(n2^n)$. □

We defer the proof of the theorem to next lecture.