COM S 6810 Theory of Computing January 27, 2009

Lecture 3:

Instructor: Rafael Pass *Scribe: Ashwinkumar B. V*

1 Review

In the last class Diagonalization was introduced and used to prove some of the first results in Complexity theory. This included

- Time Hierarchy Theorem ($DTIME(n \times \log^2(n)) \nsubseteq DTIME(n)$)
- Nondeterministic Time Hierarchy Theorem ($NTIME(n \times \log^2(n)) \nsubseteq NTIME(n)$)
- Ladners Theorem (Proves the existence of Existence of NP-intermediate problems)

After these theorems were proved there was a hope in the community that a clever use of Diagonalization might resolve the $P \neq NP$? question. But then the result that any proof which resolves $P \neq NP$? must be non-relativising with respect to oracle machines showed that $P \neq NP$? cannot be resolved just be the clever use of Diagonalization .

Definition 1 *For every $O \subseteq \{0,1\}^*$,P^O is the set of languages decided by a polynomial-time TM with oracle access to O and NP^O is the set of languages decided by a polynomial-time NTM with oracle access to O.*

Theorem 1 *[1] $\exists$ oracles O_1 and O_2 such that $P^{O_1} = NP^{O_1}$ and $P^{O_2} = NP^{O_2}$*

Proof.

- To show that $\exists$ oracle O such that $P^O = NP^O$

  Define

$$\begin{aligned} O(<M>, x, 1^n) &= 1 \text{ if M(x)=1 and M(x) stops in } 2^n \text{ steps} \\ &= 0 \text{ otherwise} \end{aligned} \tag{1}$$

$$EXP = \bigcup_{c>1} Dtime(2^{n^c}) \subseteq P^O \tag{2}$$

  Simple as the polytime TM can just make a call to the oracle.

$$P^O \subseteq NP^O \text{(Trivially true)} \tag{3}$$

$$NP^O \subseteq EXP \tag{4}$$

Any TM which runs in time EXP can enumerate all the choices of the NTM which runs in poly time and can simulate the oracle

From equations 2,3,4 we can easily see that $P^O = NP^O = EXP$

- To show that $\exists$ oracle O such that $P^O \neq NP^O$

  The above step in the proof was left as an exercise and can be found in Section 3.5 of [2]

2 Space Complexity

2.1 Notations

- $SPACE(s(n))$ =Languages decidable by some TM using $O(s(n))$ space.
- $\mathcal{L} = SPACE(log(n))$
- $PSPACE = \underset{c>1}{\cup} SPACE(n^c)$
- $NSPACE(s(n))$ =Languages decidable by some NTM using $O(s(n))$ space.
- $\mathcal{NL} = NSPACE(log(n))$
- $NPSPACE = \underset{c>1}{\cup} NSPACE(n^c)$

Definition 2 *A function $S(n)$ is said to be space constructible if $\exists$ a TM which can compute $S(n)$ using space $O(S(n))$. Similarly a function $T(n)$ is said to be time constructible if $\exists$ a TM which can compute $T(n)$ and halts in $O(T(n))$ steps.*

2.2 Some theorems

Theorem 2 *If $s(n) \geq \log(n)$ is any space constructible function then $SPACE(s(n)) \subseteq DTIME(2^{O(s(n)})$*

Proof. The result directly follows from pigeon hole principle. Consider $L \in SPACE(s(n))$. Let TM be a turing machine which solves L in polynomial space. First we enumerate the different possible snap shots(configurations) the TM can be in. The different configurations can be counted by counting different states of each part of TM.

- State of the machine-O(1) (because the states of machine are fixed and do not depend on input length)
- Position of the read/write head-O(poly(s(n),n))
- Content of work tape - $2^{O(s(n))}$ $(\sum^{O(s(n)})$

Hence we can see that total configurations$\leq$ $poly(s(n), n) \times 2^{O(s(n))}$. As $s(n) > log(n)$, we have $poly(s(n), n) \leq 2^{O(s(n))}$. Hence total configurations$\leq$ $t = 2^{O(s(n))}$. Now if the TM runs for more than t steps then by pigeon hole principle we see that the TM is in same state at two different time steps. But this implies that the TM runs into an infinite loop which is a contradiction. Hence the TM runs in time $2^{O(s(n))}$.

Theorem 3 *If* $s(n) \geq \log(n)$ *is any space constructible function then* $NSPACE(s(n)) \subseteq DTIME(2^{O(s(n)})$

Proof. The proof is extremely similar to the previous theorem. Let NTM solve $L \in NSPACE(s(n))$. By the argument given in theorem 2 we see that number of configurations $\leq 2^{O(s(n))}$. Now construct a graph with each configuration as a vertex. Then add directed edges from one vertex to another vertex if there is a transition possible by the NTM between the corresponding configurations in the execution. It is simple to see that a DFS on this graph to check if there is a path from start configuration to an accept configuration solves the problem. One can easily see that all the above steps can be done in time $O(2^{O(s(n)})$

3 Reductions and $\mathcal{NL}$-completeness

Definition 3 *L is logspace reducible to* L' *(*$L \leq_{log} L'$*) if* $\exists$ *a log space computable function such that* $x \in L$ *if and only if* $f(x) \in L'$

$\mathcal{NL}$-complete is also defined similar to NP-complete.

Definition 4 *L is* $\mathcal{NL}$*-complete if* $L \in \mathcal{NL}$ *and* $\forall L' \in \mathcal{NL}, L' \leq_{log} L$.

Lemma 4 *If* $L \leq_{log} L'$ *and* $L' \in \mathcal{L} \Rightarrow L \in \mathcal{L}$

Proof. The proof is by giving a TM which can solve L in log space. Intuitively the TM would compute $z = f(x)$. Then run $M_{L'}(z)$. But this proof would be wrong as $f(x)$ can be polynomial in the length of n. But, surprisingly a small modification gives the correct TM. Run $M_{L'}(z)$. If it tries to read a bit in z. Compute that bit in $f(x)$.

Next we give a language which is complete for $\mathcal{NL}$.

Definition 5 *STCONN:- Given a directed graph (V,E) and two vertices s,t decide whether there is a path from s to t.*

Theorem 5 *STCONN is $\mathcal{NL}$-complete*

- *STCONN* $\in \mathcal{NL}$. The proof is by giving a NTM which solves STCONN. The NTM has a counter c initialized to n and curNode=s. The NTM does the following iteratively.
  - If curNode==t then output YES.
  - If c==0 then halt
  - decrement c and move non-deterministically to one of the neighbors of curNode.

  Its easy to see that the space used for counter and index of node is logarithmic in the input size. If there exists a path then non-deterministically it is chosen in n-steps. If,no paths exist it is clear that the NTM terminates in poly time and does not output YES.

- $\forall L \in \mathcal{NL}, L \leq_{log} STCONN$. Let NTM solve L in log space. The proof is by constructing a directed graph. Define the graph as follows.
  - V={u,u is one of the possible configurations of TM}.
  - E={(u,v),there is a possible non-deterministic transitions of TM from u to v}

  It is simple to see that L is equivalent to STCONN between (s, a) where s is starting state of NTM and a is accepting state of NTM(If there are many then another node can be added to which each accepting state has an edge).

  One can easily see that if we need to construct the whole graph the reduction is no longer log space. But, a slight modification makes the reduction log space. This can be done as we don't need the complete graph to solve STCONN by a log space turing machine. We just need an adjacency oracle which gives the nodes adjacent to given node in the graph. This can be easily done by a log space TM.

4 Next Lecture

In the next lecture we will see that $NSPACE(s) \subseteq SPACE(s^2)$ and $\mathcal{NL} = co - \mathcal{NL}$. The first of these implies $\mathcal{NL} \subseteq SPACE(\log^2(n)$ and $NPSPACE = PSPACE$.

References

[1] Theodore Baker, John Gill, and Robert Solovay. Relativizations of the P =? NP question. SIAM Journal on Computing, 4(4):431442, December 1975.

[2] Sanjeev Arora and Boaz Barak. Computational Complexity: A Modern Approach