Lecture 10: NL and coNL

Anup Rao

April 29, 2021

In this lecture, we continue our discussion of space complexity classes. We first introduce a new definition. Given any set of boolean functions S, we write coS to denote the set

$$\{f : 1 - f \in S\}.$$

Thus $co\mathbf{NP}$ is the set of functions for which there is an efficiently verifiable proof that $f(x) = 0$.

Fact 1. $\mathbf{P} = co\mathbf{P}$

Fact 2. $\mathbf{L} = co\mathbf{L}$

Fact 3. $\mathbf{EXP} = co\mathbf{EXP}$

We do not know if $\mathbf{NP} = co\mathbf{NP}$. To show that $co\mathbf{NP} \subseteq \mathbf{NP}$, it would be enough to a polynomial time algorithm that can certify that a boolean formula is *unsatisfiable*.

Fact 4. *If* $\mathbf{P} = \mathbf{NP}$*, then* $\mathbf{NP} = co\mathbf{NP}$*.*

On the other hand, we can show:

Theorem 5. *For space constructible* $s(n)$*,* $\mathsf{NSPACE}(s(n)) = co\mathsf{NSPACE}(s(n))$*.*

Proof As usual we focus on the configuration graph. To prove the theorem, it will be enough to be able to verify that there is *no* path from two vertices u, v in the graph, in $s(n)$ space. This would show that if $f(x) = 1$ can be certified in space $s(n)$, then $f(x) = 0$ can also be certified in space $s(n)$. The other direction is completely symmetric.

We shall prove how to do this by designing a sequence of algorithms. Let C_i denote the set of vertices that are reachable from u in i steps. Suppose the graph is of size at most 2^s.

Claim 6. *Given any vertex v and a number $i \leq 2^s$, there is a non-deterministic space $s(n)$ algorithm such that:*

- *If $v \in C_i$, then some computational path outputs* 1
- *If $v \notin C_i$, then every computational path outputs* 0*.*

The algorithm simply guesses a path from u to v and checks that the path is a valid path of the graph by checking each edge in order.

Claim 7. *Given the size of $|C_{i-1}| = c$, and a vertex v, there is a non-deterministic space $s(n)$ algorithm such that*

- *If $v \notin C_i$, there is some computational path that outputs* 1.
- *If $v \in C_i$, then every computational path outputs* 0.

Since the algorithm is given the size of $C_{i-1}i$, the algorithm guesses each of the vertices of C_{i-1} in increasing order, and for each one, it checks that the vertex is different from the last vertex that was guessed, and then uses Claim 6 to verify that the vertex is indeed a member of C_{i-1}. It also makes sure that the given vertex is not v and not a neighbor of v. It maintains a count of all the number of vertices guessed and checks that $|C_{i-1}|$ vertices are given. If any of the checks fail, the algorithm outputs 0.

Finally, we argue that given the size of C_{i-1}, we can certify the size of $|C_i|$.

Claim 8. *Given the size of $|C_{i-1}| = c'$, there is a non-deterministic space $s(n)$ algorithm such that the algorithm either aborts or outputs $|C_i|$ on every computational path, and there is some computational path on which the algorithm outputs $|C_i|$.*

For each vertex v of the graph (in increasing order), the algorithm uses Claims 6 and 7 to check whether $v \in C_i$ or $v \notin C_i$, and it maintains a count of the number of vertices in C_i.

Thus, we obtain an algorithm that can verify that $v \notin C_n$ in $O(s(n))$ space. We first compute C_n by repeatedly using Claim 8 and then we apply Claim 7 to check whether $v \notin C_n$. ■