Lecture 4

Today's lecture continues the discussion of space complexity. We examine **NL**, the set of languages decidable in $O(\log n)$ space on a nondeterministic Turing machine, introduce log-space reducibility and define nl-completeness. Finally, we discuss Savitch's theorem, and state the main result to be proved next time: **NL** = **coNL**.

1 Reductions in NL

Last time, we introduced **NL**, the class of languages decidable in logarithmic space on a nondeterministic Turing Machine: $\mathbf{NL} = \mathbf{NSPACE}(\log n)$. Now we would like to introduce a notion of completeness in **NL** analogous to the notion of completeness that we have introduced and studied for the class **NP**. A first observation is that, in order to have a meaningful notion of completeness in **NL**, we cannot use polynomial-time reductions, otherwise any **NL** problem having at least a YES instance and at least a NO instance would be trivially **NL**-complete. To get a more interesting notion of **NL** completeness we need to turn to weaker reductions. In particular, we define *log space* reductions as follows:

DEFINITION 1 *Let A and B be decision problems. We say A is log space reducible to B, $A \leq_{\log} B$, if $\exists$ a function f computable in log space such that $x \in A$ iff $f(x) \in B$, and $B \in L$.*

THEOREM 1
If $B \in \mathbf{L}$, and $A \leq_{\log} B$, then $A \in \mathbf{L}$.

PROOF: We consider the concatenation of two machines: M_f to compute f, and M_B to solve B. If our resource bound was polynomial time, then we would use $M_f(x)$ to compute $f(x)$, and then run M_B on $f(x)$. The composition of the two procedures would given an algorithm for A, and if both procedures run in polynomial time then their composition is also polynomial time. To prove the theorem, however, we have to show that if M_f and M_B are log space machines, then their composition can also be computed in log space.

Recall the definition of a Turing machine M that has a log space complexity bound: M has one read-only input tape, one write-only output tape, and uses a log space work tape. A naive implementation of the composition of M_f and M_B would be to compute $f(x)$, and then run M_B on input $f(x)$; however $f(x)$ needs to be stored on the work tape, and this implementation does not produce a log space machine. Instead we modify M_f so that on input x and i it returns the i-th bit of $f(x)$ (this computation can still be carried out in logarithmic space). Then we run a simulation of the computation of $M_B(f(x))$ by using the modified M_f as an "oracle" to tell us the value of specified positions of $f(x)$. In order to simulate $M_B(f(x))$ we only need to know the content of one position of $f(x)$ at a time, so the simulation can be carried with a total of $O(\log |x|)$ bits of work space. □

Using the same proof technique, we can show the following:

THEOREM 2
if $A \leq_{\log} B, B \leq_{\log} C$, then $A \leq_{\log} C$.

2 NL Completeness

Armed with a definition of log space reducible, we can define **NL**-completeness.

DEFINITION 2 *A is **NL**-hard if $\forall B \in \mathbf{NL}, B \leq_{\log} A$. A is **NL**-complete if $A \in \mathbf{NL}$ and A is **NL**-hard.*

We now introduce a problem STCONN (s,t-connectivity) that we will show is **NL**-complete. In STCONN, given in input a directed graph $G(V, E)$ and two vertices $s, t \in V$, we want to determine if there is a directed path from s to t.

THEOREM 3
*STCONN is **NL**-complete.*

PROOF:

1. STCONN $\in$ **NL**.

   On input *G(V,E), s,t*, set p to s. For $i = 1$ to $|V|$, nondeterminsitically, choose a neighboring vertex v of p. Set $p = v$. If $p = t$, accept and halt. Reject and halt if the end of the *for* loop is reached. The algorithm only requires $O(\log n)$ space.

2. STCONN is **NL**-hard.

   Let $A \in \mathbf{NL}, andlet M_A$ be a non-deterministic logarithmic space Turing Machine for A. On input x, construct a directed graph G with one vertex for each configuration of $M(x)$, and an additional vertex t. Add edges (c_i, c_j) if $M(x)$ can move in one step from c_i to c_j. Add edges (c, t) from every configuration that is accepting, and let s be the start configuration. M accepts x iff some path from s to t exists in G. The above graph can be constructed from x in log space, because listing all nodes requires $O(\log n)$ space, and testing valid edges is also easy.

□

3 Savitch's Theorem

What kinds of tradeoffs are there between memory and time? STCONN can be solved deterministically in linear time and linear space, using depth-first-search. Is there some sense in which this is optimal? Nondeterministically, we can search using less than linear space. Can searching be done deterministically in less than linear space?

We will use Savitch's Theorem to show that STCONN can be solved deterministically in $O(\log^2 n)$, and that every **NL** problem can be solved deterministically in $O(\log^2 n)$ space. In general, if A is a problem that can be solved nondeterministically with space $s(n) \geq \log n$, then it can be solved deterministically with $O(s^2(n))$space.

THEOREM 4
*Every problem in **NL** can be solved deterministically in $O(\log^2 n)$ space.*

Proof: Consider a graph $G(V,E)$, and vertices s,t. We define a recursive function REACH(u,v,k) that accepts and halts iff v can be reached from u in $\leq k$ steps. If $k=1$, then REACH accepts iff (u,v) is an edge. If $k \geq 2, \forall w \in V - \{u,v\}$, compute REACH$(u,w,\lfloor k/2 \rfloor)$ and REACH$(w,v,\lceil k/2 \rceil)$. If both accept and halt, accept. Else, reject.

Let $S(k)$ be the worst-case space use of REACH$(\cdot,\cdot,k)$. The space required for the base case $S(1)$ is a counter for tracking the edge, so $S(1) = O(\log n)$. In general, $S(k) = O(\log n) + S(k/2)$ for calls to REACH and for tracking w. So, $S(k) = O(\log k * \log n)$. Since $k \leq n$, the worst-case space use of REACH is $O(\log^2 n)$. □

Comparing to depth-first-search, we find that we are exponentially better in space requirements, but we are no longer polynomial in time.

Examining the time required, if we let $t(k)$ be the worst-case time used by REACH$(\cdot,\cdot,k)$, we see $t(1) = O(n+m)$, and $t(k) = n(2 * T(k/2))$, which solves to $t(k) = n^{O(\log k)} = O(n^{O(\log n)})$, which is super-polynomial. Savitch's algorithm is still the one with the best known space bound. No known algorithm achieves polynomial log space and polynomial time simultaneously. However, there are interesting results for STCONN in undirected graphs. There exists an algorithm running in polynomial time and $O(\log^2 n)$ space (but the polynomial has very high degree), due to Nisan [Nis94]. There is also an algorithm that has $O(\log^{4/3} n)$ space complexity and superpolynomial time complexity, due to Armoni, Ta-Shma, Nisan and Wigderson [ATSWZ97], improving on a previous algorithm by Nisan, Szemeredy and Wigderson [NSW92].

Theorem 5 (Savitch's Theorem)
For every function $s(n)$ computable in space $O(s(n))$, $\mathbf{NSPACE}(s) = \mathbf{SPACE}(O(s^2))$

Proof: We begin with a nondeterministic machine M, which on input x uses $s(|x|)$ space. We define $REACH(c_i, c_j, k)$, as above, which accepts and halts iff $M(x)$ can go from c_i to c_j in $\leq k$ steps. We compute $REACH(c_0, c_{\text{acc}}, 2^O(s|x|))$ for all accepting configurations c_{acc}. If there is a call of REACH which accepts and halts, then M accepts. Else, M rejects. If REACH accepts and halts, it will do so in $\leq 2^{O(|x|)}$ steps.

Let $S_R(k)$ be the worst-case space used by REACH$(\cdot,\cdot,k)$: $S_R(1) = O(s(n)), S_R(k) = O(s(n)) + S_R(k/2)$. This solves $S_R = s(n) * \log k$, and, since $k = 2^O(s(n))$, we have $S_R = O(s^2(n))$. □

4 coNL

We now show interesting results for **NL** and **coNL**. Let $\overline{A}$ denote the complement of A — that is, if A is a set of strings for which M_A accepted, then $\overline{A}$ consists of exactly those strings for which M_A did not accept. For a complexity class $\mathcal{C}$, we define the *complement class* $\mathbf{co}\mathcal{C}$ as follows: a decision problem $A \in \mathbf{co}\mathcal{C}$ iff $\overline{A} \in \mathcal{C}$.

Observe that, for deterministic classes, the complement of a class is the class itself. However, for nondeterministic classes, complementarity involves an exchange of the quantification: one goes from $\exists$ something satisfying such a condition to $\forall$ cases, there is nothing satisfying that condition. If $A \in \mathbf{NP}$, $X \in A \Leftrightarrow y, |y| \leq p(x).V(x,y)$ accepts. If $A \in \mathbf{coNP}$,

$x \in A \Leftrightarrow \forall y, |y| \leq p(x).V(x,y)$ rejects. It considered very unlikely that **NP** = **coNP**, however the situation is different for space-bounded nondeterministic classes. The following theorem is due to Immerman [Imm88] and Sezelepscenyi [Sze88], and we will prove it next time.

THEOREM 6
NL = **coNL**

For the time being, let us see how the theorem reduces to finding a nondeterminisitc log space procedure for the complement of the STCONN problem.

LEMMA 7
Suppose $\overline{STCONN} \in$ **NL**, *then* **NL** = **coNL**.

PROOF: Consider a generic problem A in **coNL**. $\overline{A} \in$ **NL**, so $\overline{A} \leq_{\log} STCONN$. This means that there is a function f computable in logarithmic space such that $x \in \overline{A} \Leftrightarrow f(x) \in STCONN$. But then we also have $x \in A \Leftrightarrow f(x) \in \overline{STCONN}$, and so $A \leq_{\log} \overline{STCONN}$. Since, by assumption, we have $\overline{STCONN} \in$ **NL**, and **NL** is closed under logarithmic space reductions, we conclude $A \in$ **NL**, and since A was arbitrary, we have **coNL** $\subseteq$ **NL**.

Take now any problem in **NL**: $A \in$ **NL**, hence $\overline{A} \in$ **coNL** but by above, $\overline{A} \in$ **NL**, $A \in$ **coNL**, so **NL** $\subseteq$ **coNL**. $\Box$

References

[ATSWZ97] R. Armoni, A. Ta-Shma, A. Wigderson, and S. Zhou. $SL \subseteq L^{\frac{4}{3}}$. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, pages 230–239, 1997.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on Computing*, 17:935–938, 1988.

[Nis94] N. Nisan. RL $\subseteq$ SC. *Computational Complexity*, 4(1), 1994.

[NSW92] N. Nisan, E. Szemeredi, and A. Wigderson. Undirected connectivity in $O(\log^{1.5} n)$ space. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 24–29, 1992.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26:279–284, 1988.