Notes on Complexity Theory Last updated: September, 2011

Lecture 7

Jonathan Katz

1 Configuration Graphs and the Reachability Method

1.1 NL and NL-Completeness

Coming back to problems on graphs, consider the problem of *directed connectivity* (denoted CONN). Here we are given a directed graph on n-vertices (say, specified by an adjacency matrix) and two vertices s and t, and want to determine whether there is a directed path from s to t.

Theorem 1 CONN *is* NL-*complete.*

Proof To see that it is in NL, we need to show a non-deterministic algorithm using log-space that never accepts if there is no path from s to t, and that sometimes accepts if there is a path from s to t. The following simple algorithm achieves this:

if $s = t$ accept
set $v_{\text{current}} := s$
for $i = 1$ to n:
    guess a vertex v_{next}
    if there is no edge from v_{current} to v_{next}, reject
    if $v_{\text{next}} = t$, accept
    $v_{\text{current}} := v_{\text{next}}$
if $i = n$ and no decision has yet been made, reject

The above algorithm needs to store i (using $\log n$ bits), and at most the labels of two vertices v_{current} and v_{next} (using $O(\log n)$ bits).

To see that CONN is NL-complete, assume $L \in \mathsf{NL}$ and let M_L be a non-deterministic log-space machine deciding L. Our log-space reduction from L to CONN takes input $x \in \{0,1\}^n$ and outputs a graph (represented as an adjacency matrix) in which the vertices represent configurations of $M_L(x)$ and edges represent allowed transitions. (It also outputs $s = \mathsf{start}$ and $t = \mathsf{accept}$, where these are the starting and accepting configurations of $M(x)$, respectively.) Each configuration can be represented using $O(\log n)$ bits, and the adjacency matrix (which has size $O(n^2)$) can be generated in log-space as follows:

For each configuration i:
    for each configuration j:
        Output 1 if there is a legal transition from i to j, and 0 otherwise
        (if i or j is not a legal state, simply output 0)
Output start, accept

The algorithm requires $O(\log n)$ space for i and j, and to check for a legal transition. ■

We can now easily prove the following:

Theorem 2 *For $s(n) \geq \log n$ a space-constructible function,* $\text{NSPACE}(s(n)) \subseteq \text{TIME}(2^{O(s(n))})$.

Proof We can solve CONN in linear time (in the number of vertices) using breadth-first search, and so CONN $\in \mathcal{P}$. By the previous theorem, this means $\mathsf{NL} \subseteq \mathcal{P}$ (a special case of the theorem).

In the general case, let $L \in \text{NSPACE}(s(n))$ and let M be a non-deterministic machine deciding L using $O(s(n))$ space. We construct a deterministic machine running in time $2^{O(s(n))}$ that decides L by solving the reachability problem on the configuration graph of $M(x)$, specifically, by determining whether the accepting state of $M(x)$ (which we may assume unique without loss of generality) is reachable from the start state of $M(x)$. This problem can be solved in time linear in the number of vertices in the configuration graph. ■

Corollary 3 $\mathsf{NL} \subseteq \mathcal{P}$.

Summarizing what we know,

$$\mathsf{L} \subseteq \mathsf{NL} \subseteq \mathcal{P} \subseteq \mathcal{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}.$$

By the hierarchy theorems (and Savitch's theorem, below) we know NL is a strict subset of PSPACE, and $\mathcal{P}$ is a strict subset of EXP. But we cannot prove that any of the inclusions above is strict.

1.2 Savitch's Theorem

In the case of time complexity, we believe that non-determinism provides a huge (exponential?) benefit. For space complexity, this is surprisingly not the case:

Theorem 4 (Savitch's Theorem) *Let $s(n) \geq \log n$ be a space-constructible function. Then* $\text{NSPACE}(s(n)) \subseteq \text{SPACE}(s(n)^2)$.

Proof This is another application of the reachability method. Let $L \in \text{NSPACE}(s(n))$. Then there is a non-deterministic machine M deciding L and using space $O(s(n))$. Consider the configuration graph G_M of $M(x)$ for some input x of length n, and recall that (1) vertices in G_M can be represented using $O(s(n))$ bits, and (2) existence of an edge in G_M from some vertex i to another vertex j can be determined using $O(s(n))$ space.

We may assume without loss of generality that M has a single accepting configuration (e.g., M erases its work tape and moves both heads to the left-most cell of their tapes before accepting). $M(x)$ accepts iff there is a directed path in G_M from the starting configuration of $M(x)$ (called start) to the accepting configuration of $M(x)$ (called accept). There are $V = 2^{O(s(n))}$ vertices in G_M, and the crux of the proof comes down to showing that reachability on a general V-node graph can be decided in deterministic space $O(\log^2 V)$.

Turning to that general problem, we define a (deterministic) recursive algorithm Path with the property that $\mathsf{Path}(a, b, i)$ outputs 1 iff there is a path of length at most 2^i from a to b in a given graph G; the algorithm only needs the ability to enumerate the vertices of G and to test for directed edges between any two vertices i, j in this graph. The algorithm proceeds as follows:

$\underline{\mathsf{Path}(a,b,i)}$:

- If $i = 0$, output "yes" if $a = b$ or if there is an edge from a to b. Otherwise, output "no".
- If $i > 0$ then for each vertex v:
  - If $\mathsf{Path}(a, v, i-1)$ and $\mathsf{Path}(v, b, i-1)$, return "yes" (and halt).
- Return "no".

Let $S(i)$ denote the space used by $\mathsf{Path}(a, b, i)$. We have $S(i) = O(\log V) + S(i-1)$ and $S(0) = O(\log V)$. This solves to $S(i) = O(i \cdot \log V)$.

We solve our original problem by calling $\mathsf{Path}(\mathsf{start}, \mathsf{accept}, \log V)$ using the graph G_M, where G_M has $V = 2^{O(s(n))}$ vertices. This uses space $O(\log^2 V) = O(s(n)^2)$, as claimed. ■

We have seen the next result before, but it also follows as a corollary of the above:

Corollary 5 $\mathsf{PSPACE} = \mathsf{NPSPACE}$.

Is there a better algorithm for directed connectivity than what Savitch's theorem implies? Note that the algorithm implies by Savitch's theorem uses polylogarithmic space but superpolynomial time (specifically, time $2^{O(\log^2 n)}$). On the other hand, we have *linear*-time algorithms for solving directed connectivity but these require linear space. The conjecture is that $\mathsf{L} \neq \mathsf{NL}$, in which case directed connectivity does not have a log-space algorithm, though perhaps it would not be earth-shattering if this conjecture were proven to be false. Even if $\mathsf{L} \neq \mathsf{NL}$, we could still hope for an algorithm solving directed connectivity in $O(\log^2 n)$ space and polynomial time.

1.3 The Immerman-Szelepcsényi Theorem

As yet another example of the reachability method, we will show the somewhat surprising result that non-deterministic space is closed under complementation.

Theorem 6 $\overline{\text{CONN}} \in \mathsf{NL}$.

Proof Recall that

$$\overline{\text{CONN}} \stackrel{\text{def}}{=} \left\{ (G, s, t) : \begin{array}{c} G \text{ is a directed graph in which} \\ \text{there is } no \text{ path from vertex } s \text{ to vertex } t \end{array} \right\}.$$

Let V denote the number of vertices in the graph G under consideration. We show that $\overline{\text{CONN}} \in \mathsf{NL}$ using the certificate-based definition of non-deterministic space complexity. Thus, we will show a (deterministic) machine M using space $O(\log V)$ such that the following holds: if there is *no* directed path in G from s to t, then there exists a certificate that will make $M(G, s, t)$ accept. On the other hand if there *is* a directed path in G from s to t, then no certificate can make $M(G, s, t)$ accept. Note the difficulty here: it is easy to give a proof (verifiable in space $O(\log V)$) proving the *existence* of a path — the certificate is just the path itself. But how does one construct a proof (verifiable in space $O(\log V)$) proving *non-existence* of a path?

We build our certificate from a number of 'primitive' certificates. Fix (G, s, t), let C_i denote the set of vertices reachable from s in at most i steps, and let $c_i = |C_i|$. We want to prove that $t \notin C_V$. We already know that we can give a certificate $\mathsf{Path}_i(s, v)$ (verifiable in logarithmic space) proving that there is a path of length at most i from s to v. Now consider the following:

- Assuming c_{i-1} is known, we can construct a certificate $\mathsf{noPath}_i(s,v)$ (verifiable in logarithmic space) proving that there is no path of length at most i from s to v. (I.e., $v \notin C_i$.) The certificate is
$$v_1, \mathsf{Path}_{i-1}(s,v_1), \ldots, v_{c_{i-1}}, \mathsf{Path}_{i-1}(s,v_{c_{i-1}}),$$
for $v_1, \ldots, v_{c_{i-1}} \in C_{i-1}$ in ascending order. This certificate is verified by checking that (1) the number of vertices listed is exactly c_{i-1}, (2) the vertices are listed in ascending order, (3) none of the listed vertices is equal to v or is a neighbor of v, and (4) each certificate $\mathsf{Path}_{i-1}(s,v_j)$ is correct. This can all be done in $O(\log V)$ space with read-once access to the certificate.

- Assuming c_{i-1} is known, we can construct a certificate $\mathsf{Size}_i(k)$ (verifiable in logarithmic space) proving that $c_i = k$. The certificate is simply the list of all the vertices $v_1, \ldots$ in G (in ascending order), where each vertex is followed by either $\mathsf{Path}_i(s,v)$ or $\mathsf{noPath}_i(s,v)$, depending on whether $v \in C_i$ or not. This certificate can be verified by checking that (1) all vertices are in the list, in ascending order, (2) each certificate $\mathsf{Path}_i(s,v)$ or $\mathsf{noPath}_i(s,v)$ is correct, and (3) the number of vertices in C_i is exactly k.

(Note that the verifier only needs the ability to detect edges between two given vertices of G.) Observing that the size of $C_0 = \{s\}$ is already known, the certificate that $(G,s,t) \in \overline{\text{CONN}}$ is just
$$\mathsf{Size}_1(c_1), \mathsf{Size}_2(c_2), \ldots, \mathsf{Size}_{V-1}(c_{V-1}), \mathsf{noPath}_V(s,t).$$
Each certificate $\mathsf{Size}_i(c_i)$ can be verified in logarithmic space, and after each such verification the verifier only needs to store c_i. Thus the entire certificate above is verifiable in logarithmic space. ■

Corollary 7 *If $s(n) \geq \log n$ is space constructible, then* $\text{NSPACE}(s(n)) = \mathsf{co}\text{NSPACE}(s(n))$.

Proof This is just an application of the reachability method. Let $L \in \mathsf{co}\text{NSPACE}(s(n))$. Then there is a non-deterministic machine M using space $s(n)$ and with the following property: if $x \in L$ then $M(x)$ accepts on *every* computation path, while if $x \notin L$ then there is some computation path on which $M(x)$ rejects. Considering the configuration graph G_M of $M(x)$ for some input x of length n, we see that $x \in L$ iff there is *no* directed path in G_M from the starting configuration to the *rejecting* configuration. Since G_M has $V = 2^{O(s(n))}$ vertices, and the existence of an edge between two vertices i and j can be determined in $O(s(n)) = O(\log V)$ space, we can apply the previous theorem to get a non-deterministic algorithm deciding L in space $O(\log V) = O(s(n))$. ■

Corollary 8 $\mathsf{NL} = \mathsf{coNL}$.