IAS/PCMI Summer Session 2000
Clay Mathematics Undergraduate Program
Basic Course on Computational Complexity

Lecture 4: Graph Reachability and Space-Bounded Computation

David Mix Barrington and Alexis Maciel
July 20, 2000

1. Graph Reachability Revisited

Recall the graph reachability problem: given a directed graph G and two nodes s and t, determine whether there is a path from s to t in G. In the previous lectures, we showed that this problem is in P and NL. In this lecture, we show that it is both in AC^1 and $\mathrm{DSPACE}(\log^2 n)$. (Note that $\log^2 n$ means $(\log n)^2$ and not $\log \log n$.) In the next section, we use these upper bounds to establish a general simulation of space-bounded nondeterministic machines by space-efficient deterministic machines.

Theorem 1 *Graph reachability is in* AC^1.

Proof Let G be an input graph with n nodes numbered $1, \ldots, n$. Let A be the $n \times n$ 0-1 matrix defined by $A_{ij} = 1$ if and only if $i = j$ or there is an edge from node i to node j. This matrix should be easy to compute from any reasonable encoding of G. Also, note that $A_{ij} = 1$ if and only if there is a path of length at most 1 from i to j.

Now consider $A^2 = A \times A$ defined as follows: $A^2_{ij} = 1$ if and only if there is k such that $A_{ik} = 1$ and $A_{kj} = 1$. This implies that $A^2_{ij} = 1$ if and only if there is a path of length at most 2 from i to j. In general, $A^r_{ij} = 1$ if and only if there is a path of length at most r from i to j.

If there is a path from s to t, there must be one of length at most n. Therefore, to solve the reachability problem, we only need to determine whether $A^n_{st} = 1$. To do this, our circuit will compute A^n by using a binary tree of matrix multiplications. The depth of this tree will be $O(\log n)$.

Each node in the tree computes the square of some power of A. Since for any matrix B, $B^2_{ij} = 1$ if and only if there is k such that $B_{ik} = 1$ and $B_{kj} = 1$, we see

that these matrix multiplications can be carried out by a simple depth-2 circuit of polynomial size. Therefore, the entire circuit is of logarithmic depth and polynomial size, so graph reachability is in AC^1. □

Theorem 2 *Graph reachability is in* $\mathrm{DSPACE}(\log^2 n)$.

Proof We follow essentially the same strategy and determine whether $A^n_{st} = 1$ as follows. For each node k, recursively verify whether $A^{n/2}_{sk} = 1$ and $A^{n/2}_{kt} = 1$. If both tests succeed, accept. Otherwise, go to the next node k. If none of the nodes k worked, then reject.

At each level of the recursion, we need to store the current nodes s, t and k and the current exponent of A. All of these take space $O(\log n)$. The depth of the recursion is $O(\log n)$ so the entire algorithm uses $\log^2 n$ space.

One detail is that we cannot store the entire matrix A. Instead, we compute its entries as needed from the encoding of G. □

Note that the strategy used in the preceding results is essentially equivalent to a *middle-first search* where you determine whether there is a path of length n from s to t by verifying whether there are paths of length $n/2$ from s to u and from u to t for all possible nodes u. As stated, these recursive algorithms run in $O(\log^2 n)$ space and $n^{\log n}$ time, but it is possible to modify them so they run in polynomial time (and space), using the technique of dynamic programming.

2. Savitch's Theorem

As mentioned in the previous lecture, with respect to time, it is not known whether nondeterministic machines can be simulated efficiently by deterministic ones. In particular, $\mathrm{NP} \subseteq \mathrm{EXP}$ but it is not known whether $\mathrm{NP} \subseteq \mathrm{P}$.

In the case of space, however, such efficient simulations exist. In this section, we will show that $\mathrm{NSPACE}(s)$ is contained in $\mathrm{DSPACE}(s^2)$, provided $s(n) \geq \log n$. We will also show that every $\mathrm{NSPACE}(s)$ language can be computed by unbounded fan-in circuits of depth $O(s)$ and size $2^{O(s)}$. In particular, this implies that NL is contained in both AC^1 and $\mathrm{DSPACE}(\log^2 n)$ and that NPSPACE = PSPACE.

Before proving these results, a general comment about space-bounded machines. Let M be a machine running in space s. Given an input x of length n, a *configuration of M on x* consists of the state of M, the position of each of its heads

and the contents of its memory. If M uses at most s space, then M has at most $O(1)ns^{O(1)}O(1)^s = n2^{O(s)}$ different possible configurations. If M is deterministic, then during its computation on x, no configuration can repeat. Otherwise, M would be in an infinite loop and the computation would not halt. Therefore, M must run in time $n2^{O(s)}$. If $s(n) \geq \log n$, then this is simply $2^{O(s)}$. In particular, this implies that L $\subseteq$ P and that PSPACE $\subseteq$ EXP.

Now let N be a nondeterministic machine that runs in space s. It is still true that N has at most $n2^{O(s)}$ different possible configurations on any input of length n. Given an input x of length n, consider the directed graph whose nodes are all the possible configurations of N on x. Connect two configurations with an edge if the first one can lead to the other in one step of the computation of N. This can be easily established by examining the program of N. Since N is nondeterministic, more than one edge can come out of any configuration. But determining whether N accepts x is now equivalent to determining whether there is a path in this graph from the start configuration to any configuration that contains the accept state. We have therefore reduced the simulation of a nondeterministic machine to a graph reachability problem. This is the key idea in the results that follow.

Theorem 3 *Every language in* NSPACE(s) *can be decided by a family of unbounded fan-in circuits of depth* $O(s)$ *and size* $2^{O(s)}$*, provided* $s(n) \geq \log n$*.*

Proof By the previous discussion, we only need to solve up to $2^{O(s)}$ graph reachability problems on the graph of configurations of the nondeterministic machine. That graph has size at most $2^{O(s)}$ and can be easily computed from the input by a simple circuit. Note that this circuit will depend on the details of the program of the machine.

According to the results of the previous section, these graph reachability problems can be solved by unbounded fan-in circuits of depth logarithmic in $2^{O(s)}$ and size polynomial in $2^{O(s)}$. The result follows. □

Corollary 4 NL *is contained in* AC^1*.*

Theorem 5 NSPACE(s) *is contained in* DSPACE(s^2)*, provided* $s(n) \geq \log n$*.*

Proof We use the same idea and solve up to $2^{O(s)}$ graph reachability problems using a deterministic machine. This can be done in space $O((\log 2^{O(s)})^2) = O(s^2)$.

However, a detail must be dealt with. The deterministic machine must be able to compute the adjacency matrix of the graph. This can be done by incorporating

the program of the nondeterministic machine into the program of the deterministic machine. But to run the simulation, and in particular to generate configurations, the deterministic machine needs to know the value of $s(n)$.

This value can be computed as follows. Consider the graph that consists of all the configurations that use space at most $\log n + 1$. Determine if any of the configurations that uses space exactly $\log n + 1$ can be reached from the start configuration. If so, repeat on the graph of configurations that use space at most $\log n + 2$, then $\log n + 3$, etc. At some point, we will find a value r such that no configuration that uses space r can be reached from the start configuration. This means that N runs in space at most $r - 1$ and we can use this value as $s(n)$. □

Corollary 6 $\mathrm{NL} \subseteq \mathrm{DSPACE}(\log^2 n)$ *and* $\mathrm{NPSPACE} = \mathrm{PSPACE}$.

3. Exercises

1. A circuit is *semi-unbounded* if its AND gates have unbounded fan-in but its OR gates have fan-in 2, or vice-versa. Show that the AC^1 circuit for graph reachability is semi-unbounded.

2. Show that the reachability problem for graphs of out-degree 1 is in L. That is, given a directed graph G in which every node has out-degree 1 and two nodes s and t, determining whether there is a path from s to t can be done in L.

3. By using a simple argument based on the number of possible configurations, show that NL is contained in NP. (Note: It is not true that for any given sequence of choices, no configuration can repeat.)

4. Show that NL is contained in P.

5. Where is the assumption $s(n) \geq \log n$ used in the proof of Savitch's Theorem? What would the simulation of $\mathrm{NSPACE}(\log \log n)$ give you?

6. Given a directed graph G and a node s, show that computing the number of nodes reachable from s in G can be done in NL in the following sense: there is a nondeterministic machine N running in logarithmic space such that for every input, N will accept and output the correct number for at least one sequence of choices and N will reject for all other sequences of choices.

7. Use the result of the preceding exercise to show that NL is closed under complementation.