CS 221 Computational Complexity, Lecture 5 Feb 6, 2018

Space and Nondeterminism

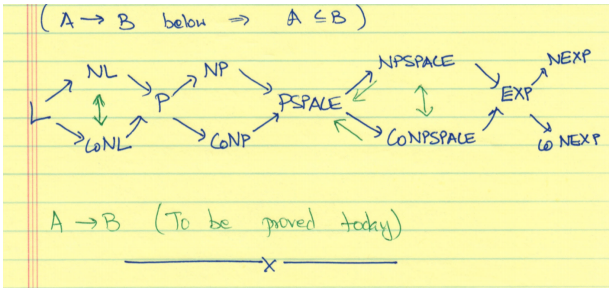
Instructor: Madhu Sudan *Scribe: Yong Wook Kwon*

1 Topic Overview

Today we'll talk about space and non-determinism. For some context, we have the crude inclusions:

$$TIME(t) \subset NTIME(t) \subset SPACE(t) \subset NSPACE(t) \subset TIME(2^t) \tag{1}$$

The first and third inclusions are obvious. The second inclusion follows as one cannot consume space faster than the actual run-time. The final inclusion follows from envisioning the total possible states (2^t) and modelling the algorithm as a walk on the directed graph of these states (from the start state to the end-state). The crude bounds above can be summarized in the following diagram.



As the diagram suggests, the main goal of this lecture is to prove the following:

1. Equality between NSPACE, PSPACE, and CoNPSPACE
2. Equality between NL and Co-NL

While the former result essentially follows from Savitch's theorem and the work we did for the first problem set, the latter result is much more surprising. It is yet unknown whether NL and L are equal or not, and the fact that we can prove equality between the NL and CoNL without going through L is striking.

After we are done proving these identities, we will describe the formalism of QBFs and games, which will give us a description of an illustrative PSPACE-complete problem.

2 PSPACE = NPSPACE = CoNPSPACE

First, we shall describe the Path Problem, which is NL-complete.

Definition 1. *The Path Problem takes as input a directed graph G and two vertices s and t, and determines whether there exists a path in G from s to t.*

Theorem 2. *Path is NL-complete under log-space reductions.*

Proof. One can represent any non-deterministic log-space algorithm using a configuration graph of size $2^{\log n} = n$. Then, the machine M accepting an input x is equivalent to there being a path from the start state to the accepting state in the configuration graph $G_{M,x}$. □

Next, as review, we shall state and prove Savitch's Theorem:

Theorem 3 (Savitch)**.** $NL \subset L^2$ *(SPACE(*$\log^2 n$*))*

Proof. This is what we did in the problem set. First, we showed that given a boolean matrix A, one can compute A^n (boolean multiplication) in $\log^2 n$ space by recursively squaring the matrix.

Now, let A be the adjacency matrix of the directed graph G. Suppose that we also add self-loops for each vertex, representing the possiblity that the algorithm may do nothing (this corresponds to having the diagonal be 1). Then, it is easy to see that there exists a path from s to t in k steps if and only if $A^k[s,t] = 1$. As we added self-loops, $A^k[s,t] = 1$ if and only if there is a path from s to t of at most length k. As any connected points have path of length at most n, suffices to compute A^n, which takes $\log^2 n$ space, and as we showed that the NL-complete problem PATH was in L^2, we are done. □

Corollary 4. *Padding arguments will thus imply that SPACE(*$s(n)$*) is contained in SPACE(*$s(n)^2$*).*

Exercise 5. *Make the explicit padding argument.*

Corollary 6. *NPSPACE is contained in PSPACE, and hence we have equality. Also, as PSPACE is closed under complement, NPSPACE = PSPACE = CoNPSPACE*

Remark The idea behind the proof of Savitch's theorem, where we can compute powers via repeated squaring with low space, will be useful later on.

3 NL = CoNL

Immerman '87 and Szelepcsenyi '88 gave the following proof for NL = CoNL.

Essentially, they give a non-deterministic algorithm (log space) for $\overline{Path}$, i.e. the complement of the Path problem. In other words, the set-up is as follows:

1. An input tape that contains the directed graph G, etc
2. The output tape, as usual
3. The proof tape π: interestingly, the log space constraint implies that we cannot read the proof tape in both directions, and only have one-directional access.
4. The memory tape, which can store $\log n$ bits.

As is common, we can reduce to the problem Path(s, t, k), i.e. whether "no paths exist from s to t of length at most k". Thus, suffices to prove that there exists such a proof π, which we can check using log space, that allows us to answer this problem. The key idea is as follows: we can non-deterministically prove that "no paths from s to k of length at most k exist" given N_k.

Definition 7. N_k *is the number of vertices reachable from* s *in* $\leq k$ *steps.*

Lemma 8. *There exists a proof such that conditional on knowing* N_k*, one can determine the above question.*

Proof. The proof π consists of the pairs (i, b_i, p_i), where i ranges over the vertices of G. $b_i = 1$ iff i is reachable from s in $\leq k$ steps. If $b_i = 1$, p_i is such a path from s to i, and if $b_i = 0$, p_i is arbitrary.

Given π and N_k, one can solve the problem using log space. Simply read through b_i, and count $\sum_i b_i$, and verify the authenticity of p_i for which $b_i = 1$. Finally, if $\sum_i b_i = N_k$, and $b_t \neq 1$, we have a proof. □

In fact, we can go further:

Lemma 9. *There exists a proof, denoted as* Π_k^-*, such that conditional on knowing* N_{k-1}*, one can determine the above question.*

Proof. Now, the proof π consists of the pairs (i, b_i, p_i) with i ranging over the vertices of G such that $b_i = 1$ iff i is reachable from s in $\leq k-1$ steps. p_i is then such a path, which one can verify easily.

The key now is for us to check that i is then *not connected to* t. This is simple combinatorics: if t does not have a neighbor (including itself) not reachable in $k-1$ steps, it certainly is not reachable in k steps. We conclude checking the proof by verifying $\sum_i b_i = N_{k-1}$. □

Alright. But how do we obtain N_{k-1}? One can obtain N_{k-1}, and hence the proof for the original problem, via "inductive counting". In other words, conditional on knowing the value of N_{k-1}, we can give a proof for the value of N_k.

We start at $N_0 = 1$. Next is the inductive step: Π^k is a proof for the value of N_k given the value for N_{k-1}. Π^k consists of the triplet $(i, b_i, \Pi^{\pm}_{i,k})$, where i ranges over the vertices of G. b_i is 1 iff there is a path from s to i in at most k steps.

If $b_i = 1$, it is followed by $\Pi^+_{i,k}$, which is the proof, i.e. the said path. On the other hand, if $b_i = 0$, it is followed by $\Pi^-_{i,k}$, which is the proof that there is no such path! By our lemma, as long as we know the value of N_{k-1}, there is such a valid proof $\Pi^-_{i,k}$. We conclude our verification by checking $\sum_i b_i = N_k$.

Hence, we have given you a proof Π, which starts from N_0, and inductively proves the value of N_k, which then yields the answer to $\overline{Path}$, as desired. Thus, the proof is complete, and our proof is also complete.

Remark Just to clarify, the algorithm is nondeterministic, but the proof itself (Π) is deterministic. The proof is also called a witness among cryptographers. To reiterate, proofs capture the essence of nondeterminism. A problem can be nondeterministically solved if and only if I can verify the solution in the same complexity.

Remark There is no explicit limit to the length of the proof, although a bound is possible as we are traversing along all possible configurations of the machine with $\log n$ memory.

Exercise 10. *From the description of* Π *above, give a more explicit computation of the length of* Π.

Remark Note that if one can give a log-space verifiable proof of an NP statement (such as a three-colorability of a graph), this will imply via $NL \subset P$ that P = NP! Hence, we probably cannot do this.

However, one can come close - there is a randomized log space algorithm which can verify three-colorability with high probability.

So we reduced CoNL to NL. What about the other direction, to complete the proof?

We shall show in this week's problem set that there is an appropriate log space reduction:

Exercise 11. *Given* (G, s, t)*, a directed graph* G *and two vertices* s*,* t*, show that there is a* (G', s', t') *such that there is a path from* s' *to* t' *in* G' *iff there is no path from* s *to* t *in* G.

4 Polynomial Space Complete Problems: QBF and Games

Leaving the space of NL, let us move up a hierarchy and focus on PSPACE complexity. Naturally, we wish to describe a PSPACE complete problem, just as we have Path for NL.

One way is to find an appropriate generalization of Path:

Definition 12. *The Succinct Path problem takes as input a circuit* $C : \{0,1\}^n \times \{0,1\}^n \mapsto (0,1)$*. One can define the corresponding adjacency matrix of a graph of size* 2^n*, where* $A[s,t] = C(s,t)$*. SUCCPATH(s, t) asks whether there is a path from* s *to* t.

Exercise 13. *Prove that SUCCPATH is PSPACE-complete.*

Succinct path, however, does not give us a good structural understanding of PSPACE. Instead, let us introduce the notion of QBF (quantified boolean formulas).

Suppose there is a fixed formula (say, 3CNF) ϕ, and inputs x_1, x_2, ... x_n. Let n be even. The question we are trying to answer is not satisfiability, but a stronger one.

Definition 14. *QBF asks whether there exists x_i such that there exists an x_1 such that for all x_2, there exists x_3 such that for all x_4..... there exists x_{n-1} such that for all x_n $\phi(x_1, x_2, ...x_n) = 1$. ($\exists x_1 \forall x_2 ....\exists x_{n-1} \forall x_n$ such that $\phi(x) = 1$)*

Theorem 15. *QBF is NPSPACE (and hence PSPACE) complete.*

Before we describe the proof, it is helpful to get a better picture of what QBF is. Think of the process as a game between the existential player and the universal player. The existential player wins if $\phi(x_1, x_2, ...x_n) = 1$ (which we can easily verify).

Then, QBF asks whether there is a winning strategy by the existential player (characterized by $(x_1, x_3, x_5, ..., x_{n-1})$) such that no matter what the universal player plays, player 1 will win.

One can generalize QBFs to games:

Definition 16. *Now, x_i need not be bits, but general inputs. Then, suppose there is a polynomial time input verification predicate $V(x_1, ...x_n)$. The associated game question asks whether there exist $(x_1, x_3, ...x_{n-1})$ such that player 1 guarantees a win regardless of player 2's input.*

Obviously, QBF reduces to games, but really, they can be seen as equivalent. Formally, one can have player 2 choose to ignore each bit of the transmission of x_1 until its completion, and similarly for player 1, etc.

Before we prove that QBF is PSPACE complete, we first need to show that QBF is in fact in PSPACE.

Theorem 17. *QBF and GAMES are both in PSPACE.*

Proof. Let us give a sketch of the proof. We shall prove this only for QBF. One can think about the associated game tree, which is a binary tree, with the leaves being the value of $\phi(x_1, x_2, ...x_n)$, where x is the history of the game play.

One can then reduce the original QBF problem into finding the value at the base of the tree, where player 2 nodes (universal nodes) compute the AND of their children, while the player 1 nodes (existential nodes) compute the OR of their children.

Then, one can compute this in PSPACE by essentially doing a depth-first search. At each search, remember all the left node children of the vertices belonging to the path, which only requires polynomial space. This is sufficient to compute the value at the base. □

Exercise 18. *Fill out the details of the above proof. In particular, verify that QBF is equivalent to computing the base value of the associated game tree, explicitly spell out the DFS algorithm, and finally confirm that this takes polynomial space.*

Now, let us finally prove that QBF is NPSPACE complete.

Suppose we have a general NPSPACE problem. Given, say, a space of n, there exists 2^n total possible configurations. Let the initial configuration be C_I and the final accepting configuration be C_F. Any algorithm reduces to proving there exists a path from C_I to C_F in 2^n steps. We shall reduce that problem to GAMES.

1. First, player 1 gives the state C_M, and claims that there is a path from C_I to C_M^1 in 2^{n-1} steps, and a path from C_M^1 to C_F in 2^{n-1} steps.
2. Player 2 then casts doubt on one of the two assertions.
3. Suppose WLOG that player 2 chose the first half. Then, player 1 furbishes C_M^2 between C_I and C_M^1, with a path connecting C_I to C_M^2 and a path connecting C_M^2 to C_M^1 in both 2^{n-2} steps.
4. Player 2 then casts doubt on one of the two assertions, and so forth.
5. Repeat the process until player 1 communicates D, E.
6. Player 1 wins if D and E are indeed adjacent, which can be easily verified in polynomial space.

One can easily see that the GAME described above has a winning strategy for player 1 if and only if there is indeed such a path of length 2^n from the initial state to the end state. If there does not exist such a path, the player 2 can pinpoint the error in the assertion and win. Furthermore, the game terminates after n rounds, or "alternations", and hence the total computation is polynomial time. Furthermore, each communication takes up n space (the second player only uses one bit per round, of course). Finally, one can verify that the game was played legally (i.e. player 1 conformed with player 2's moves,), and the outcome of the game (whether the final states are adjacent) using only polynomial space, as desired.

Consequently, we have shown that PSPACE reduces to games, which reduces to QBF. Hence, the best way of thinking about PSACE is via QBFs.

5 Looking ahead

What we have seen in the discussion of the previous section is that PSPACE can be represented by a game with n-length quantifiers, n verification time, and n alternations (rounds of the game).

Alternations itself is an interesting concept. We can formulate the following complexity class:

Definition 19. *ATISP(a(n), t(n), s(n)) is defined to be the type of problems solvable via games using $t(n)$ time, $s(n)$ space, and $a(n)$ rounds.*

$a(n)$ need not even be a function: what can you do with, say, 10 alternations, with polynomial space and time? These questions are difficult.

Even a single iteration of the existential player yields the class NP (using 3SAT, for example), and a single iteration of a universal player yields the class coNP. One can think about QBF for two rounds: $\exists x \forall y \phi(x, y)$ contains MINDNF: there exists an x such that for no shorter y $\phi(x, y)$ is satisfied.

We shall investigate the complexity ATISP in future lectures in more detail.