6.841/18.405J: Advanced Complexity Theory February 19, 2002

Lecture 4

Lecturer: Madhu Sudan *Scribe: Ben Barden*

In this lecture, we will discuss Fortnow's time/space lower bound for SAT, and see why alternation is considered a powerful tool for proof. We will also have an introduction to the polynomial hierarchy

1 Fortnow's Theorem

1.1 groundwork

Before we begin, we will be using a new class for this lecture, and pretty much for this lecture only: LIN, or near-linear time.

Definition 1 *LIN* $= \bigcup_c$ *TIME*$(n \log^c n)$ *for* $c > n$

Now, let us consider SAT. Specifically, let us consider an aspect of SAT that we know very little about, the lower bounds on its deterministic time/space requirements. We really don't know much. We do, however, have some pretty strongly held beliefs and intuitions about them.

belief 1 *We believe that SAT is not solvable in deterministic polynomial time.*

belief 2 *We believe that SAT is not solvable in deterministic logarithmic space.*

belief 3 *We believe that SAT is not solvable in nearly linear time.*

Note that if belief number 1 is true, then it follows that the other two are true as well. So, beliefs 2 and 3 are, as such things go, fairly weak statements. Why, then, are they interesting? Well, in the case of 3, we note that non-deterministically, SAT is solvable in near-linear time, and, in fact, is complete for non-deterministic near-linear time. Therefore, if we could actually prove that it required more than near-linear time on a deterministic machine, that would be a clear and provable case of the power of non-determinism. We know, though, that alternation is powerful for small space computation, as. The other interesting thing is that, in spite of their weakness, neither 2 nor 3 has ever been proven. They are not entirely without support, though. In 1997, Fortnow managed to prove that at least one of the two had to be true, even if he could not be certain of which one.

Theorem 2 *either SAT cannot be solved deterministically in logarithmic space or it cannot be solved deterministically in near-linear time.*

The proof depends heavily on the following fact

Fact 3 *if* $a' \leq a - 1$ *and* $t' \leq \frac{t}{logt}$ *then* *ATIME*$[a, t] \not\subseteq$ *ATIME*$[a', t']$

which can be displayed through relatively simple diagonalization. Fortnow's shows us that if SAT is solvable in both deterministic log space and deterministic linear time, it is possible to show that ATIME$[a, t] \not\subseteq$ ATIME$[a', t']$ for those relative definitions of a, a', t, and t', thus causing a contradiction, and proving that at least one of the assumptions must be untrue. Now, we will not be giving the full proof. For simplicity's sake, we will only be showing that we can reach a contradiction if SAT $\in$ Time$(n \log n)$ and SAT $\in L$, and we are only giving the main steps, leaving many of the details of the proof as exercises

1.2 step 1

Fact 4 *If a language L is in NTIME(t), and x is a string of length n, then* $\exists$ *an SAT instance* ϕ *of size* $t(n) \log t(n)$ *such that* $x \in L$ *iff* $\phi \in$ *SAT*

The proof here can either be taken as an exercise by the reader, or referenced in Cook's 1971 paper on the topic.

1.3 step 2

Fact 5 *Given our assumptions, ATIME[a, t] is contained in NTIME[$t(\log t)^{2a}$]*

proof as follows: To begin, we make the first of our two assumptions.

Assumption 6 *SAT can be solved in deterministic, $n \log n$ time.*

One of the oddly more useful aspects of this statement for our purposes is the fact that if SAT can be solved deterministically in $n \log n$ time, then coSAT can also be solved deterministically in $n \log n$ time. Additionally, as this implies that SAT is in P, P=NP=coNP, and thus that coNP qualifies as a language L in NTIME($n \log n$). From this we can know by Fact #4, that it can be simulated by a SAT instance of size $n \log n \log n \log n$, or $n(\log n)^2$.

1.4 step 3

Fact 7 *Given our assumptions, NTime[$t(\log t)^{2a}$] is in SPACE[$logt + a \log\log t$]*

and now, our second assumption.

Assumption 8 *SAT can be solved in deterministic, logarithmic space.*

given this assumption, then we can know that NTime[$t(\log t)^{2a}$] is in SPACE[$logt + a \log\log t$].

1.5 step 4

Fact 9 *SPACE[s] is contained within ATISP[$b, 2^{\frac{s}{b}}, bs$], which in turn is contained in ATIME[$b, 2^{\frac{s}{b}}$]*

1.6 results and considerations

Finally, we end, after a roundabout path and a bit of variable replacement, with the following conclusion

Fact 10 *Given our assumptions, ATIME[a, t] $\subset$ NTIME[$t(\log t)^2 a$] $\subset$ SPACE[$\log t + a \log\log t$] $\subset$ ATISP[$b, 2^{\frac{\log t + a \log\log t}{b}}$,*
$ab \log\log t$] $\subset$ ATIME[$b, 2^{\frac{\log t + a \log\log t}{b}}$]

note that this is for any b. If we choose $b = a - 1$, and clean it up a bit, we come up with

Fact 11 *Given our assumptions, ATIME[a, t] $\subset$ ATIME[$b, 2^{\frac{\log t + a \log\log t}{a-1}}$]*

now, $a \geq b$, and $2^{\log t - \log\log t}$ (another form of $\frac{t}{logt}$) is obviously bigger than $2^{\frac{\log t + a \log\log t}{a-1}}$ for any a greater than 2. Our very first fact said that ATIME[a, t] cannot be contained within ATIME[$b, 2^{\frac{\log t + a \log\log t}{a-1}}$], and now we discover that if our assumptions are both true, it must be. Therefore, at least one of our assumptions must be false. Note that here is a sterling example of the power of alternation as a tool for proof. The statement to be proven said nothing about alternation. The proof used it heavily and integrally.

2 The Polynomial Hierarchy

2.1 definitions and preliminary notes

Definition 12 Σ_i^p *is that class of languages that can be accepted by an ATM that starts in an existential state, has up to i alterations, and runs in polynomial time.*

Definition 13 Π_i^p *is that class of languages that can be accepted by an ATM that starts in a universal state, has up to i alterations, and runs in polynomial time.*

Definition 14 *The Polynomial Hierarchy is the union over all constant i of all languages in* Σ_i^p.

Now, every language in any Σ_i^p is contained easily within Π_{i+1}^p, as they can be simulated merely by ignoring the first universal state and alternating immediately to the first existential state. Π_i^p is contained in Σ_{i+1}^p for similar reasons. Thus, the polynomial hierarchy also contains all Π_i^p for constant i. Also note that every language in Π_i^p has its complement in Σ_i^p and vice versa. By general notation, P is both Π_0^p and Σ_0^p, thus making Σ_1^p and Π_1^p NP and coNP respectively.

2.2 iTQBF

As it turns out, the polynomial hierarchy has a number of complete problems, for every level. The canonical complete problem is iTQBF, the limited-alternation version of the PSPACE-complete problem TQBF. iTQBF is the collection of languages that take problems either of the form $\forall\{a_1, a_2, a_3..., a_i\}\exists b_1, b_2, b_3...b_i\forall...$ and a boolean formula, such that there are no more than i groups of no more than O(i) variables each, or of a nearly identical form, differing only in that the first quantifier is universal rather than existential. In either case, each language in the collection accepts if the formula can be true under the restrictions, and rejects otherwise. The proof that it is complete is fairly easy for polynomial time reductions, as each member of the collection is essentially the definition of its particular place in the hierarchy in language form. Let us consider a random language A in Σ_i^p. Now, For each such A, there exists a language B in Π_{i-1}^p, and some constant c less than infinity, such that $x \in A$ iff $\exists y$ such that $||y|| < ||x * c||$ and $(x, y) \in B$. All of the existential decisions required before the first universal decision can easily be encoded on such a string. Similarly, for all languages A' in Π_i^p there exists a language B', and some constant c less than infinity, such that $x \in A'$ iff $\forall y$ such that $||y|| < ||x * c||$, $(x, y) \in B$. Again, any universal decisions could be easily coded on such a string. Given these two, we can derive that for any random language A in, for example, Σ_{i+1}^p, that there exists a language Q in either Σ_1^p or Π_1^p such that $x \in A$ iff $\exists y_1\forall y_2\exists y_3\forall y_4...y_i$ such that $\{x, y_1, y_2, y_3...y_i\} \in Q$ Note that Now, if we examine iTQBF again, we discover that the version for Σ_1^p is equivalent to SAT, and that for Π_1^p equivalent to coSAT. We can thus reduce any Q in either Σ_1^p or Π_1^p to the appropriate iTQBF, so long as the information on the input strings are available. The information on the input strings, in turn, can be described exactly as the universal and existential input groups on the higher levels of the proof.

2.3 minDNF

Some of the PH-complete problems, though, are not so easily proven. minDNF is a good example. MinDNF is a language that takes (ϕ, k), where ϕ is a TM and k is a number, and returns true if there exists a TM ψ such that $||\psi|| < k$ (using some appropriate TM size rubric) and ψ returns the same output as ϕ on all inputs. This language was so difficult to prove complete that, even though it was first described during the 70s in Meyer and Stockmeyer's original paper on PH, it wasn't until 2000 that it was proven Σ_2^p-complete buy Umans. We will not be sketching the proof here.

3 Coming Attractions

In our next lecture, we will examine the PH Collapse Hypothesis, a rather strong assumption that states that $\forall i, \Sigma_i^p \neq \Pi_i^p$. Among a number of other things, it implies both that P $\neq$ NP, and that NP $\neq$ coNP.

4 References

S. Cook. The complexity of theorem-proving procedures. In Proceedings of the 3rd ACM Symposium on computing, pages 151-158. ACM, New York, 1971