COS522: Computational Complexity Fall 2011
Princeton University
Instructor: Sanjeev Arora

Lecture 2 notes Sept 20. Diagonalization

1. Review of P, NP. Time(t(n)) and Ntime(t(n)).
2. Diagonalization.
3. Review: undecidability of halting problem.
4. Each TM has finite description, so can effectively enumerate all possible TMs M1, M2, M3,.. (Some TMs may be nonsensical or crash; hence their output may be undefined.)
5. Suppose H decides halting problem.
6. On input 1^i, ask H whether Mi accepts this input.(In other words, simulate H on the input (Mi, 1^i).) Flip the answer.
7. Now we want to show that Time(n^3) has languages that Time(n^2) doesn't.
8. Ideas?
9. We explicitly describe such a language. Runs in time n^3. Advantage over every n^2 time machine: can simulate it, figure out its answer and flip.
10. Try 1: On input 1^i simulate Mi on it for i^{2.5} steps. If it gives an answer in that time, flip the answer else reject.
11. Problems with this?
12. Fix: Assume every Mi appears an infinite number of times in our enumeration. Thus we will run Mi (represented infinitely different ways) on larger and larger inputs. At some point the input is large enough that i^{2.5} exceeds the running time of Mi on 1^i.
13. Nondeterministic time hierarchy theorem. What goes wrong if we try the above proof? Don't have enough time to flip the answer!
14. Lazy diagonalization; flip answer in sufficiently large interval.
15. Why lazy diagonalization works: proof by picture.
16. Can diagonalization resolve P vs NP? Diagonalization treats TM as a black box.
17. Oracle TM. "New world" where every machine has a new capability. Eg P^{SAT}. Clearly, a proof that treats every machine as a black box still works if all machines have the same oracle.
18. Theorem: Exists A st P^A =NP^A. Exists B st P^B \neq NP^B.
19. Interpretation: Diagonalization alone cannot resolve P vs NP.
20. Construction of B. U_B = {1^n: some string of length n is in B} Clearly U_B is in NP^B for every B. We construct a B such that it is not in P^B. Infinite process. Initially throw all strings out of B. Now do following for i=1, 2, 3, .. Take machine Mi and let n be such that thus far we have never queried the oracle on any string of length n, and Mi's running time is less than n^{log n} (say). Now simulate Mi on 1^n. Whenever Mi asks about a string whose status is already determined, answer consistently with before. Otherwise say "No." At the end if Mi accepts, then declare all remaining strings of length n to

also be out of B. If Mi rejects then pick one string that Mi has not asked about, and declare it to be in B. (Thus Mi was wrong about 1^n.)

21. Construction of A: It is just EXPTIME.
22. How convincing is the relativization result to you?? Will see usefulness of diagonalization later on.