**CS 221: Computational Complexity** **Prof. Salil Vadhan**

**Lecture Notes 1**

**January 25, 2010** **Scribe: Richard Liu**

# 1 Course Overview

**Reading:** Arora-Barak, Introduction.

Computational complexity aims to understand the limits of "efficient" computation. We ask:

*Which problems can and cannot be solved with constrained computational resources?*

There are two perspectives from which one can approach this question. One is that the main object of study is *computation*: we start with models of computation and the associated computational resources, and ask what can they achieve. A second is that the main object of study is *computational problems*: we start with problems, and ask how efficiently can they be solved with respect to a variety of resources. Needless to say, these two perspectives complement and inform each other, and we will see both in this course (but with a bit more emphasis on the former).

**Some models of computation**

- Turing machine
- RAM model
- Boolean circuits
- Arithmetic circuits
- Interactive Protocols
- Quantum Computing
- Parallel Computers

**Computational resources, associated complexity classes and some problems they contain**

| Resources | Classes (Problems within the class) |
|---|---|
| Time | **P** (Linear Prog., Max Flow), **EXP** (Logic), **EE** |
| Space | **L** (Basic Arithmetic), **PSPACE** (Games) |
| Nondeterminism | **NP** (Many), **NL** (Connectivity) |
| Circuit Size | **P/poly** |
| Circuit Depth | **NC** |
| # of Arithmetic Ops. | **VP**, **VNP** (Matrix Mult./Determinant/Permanent) |
| Randomization | **BPP**, **RP** (Markov Chain Monte Carlo) |

**Some Goals of Complexity Theory**

- Algorithms: upper bounds on resources required. Mostly done in algorithms courses, but will also appear here (especially for algorithms involving non-traditional resources like nondeterminism, interaction).
- Lower bounds: a major goal, but very difficult. We'll get some taste of some beautiful (but modest) lower bounds known, and some of the barriers to proving things like $\mathbf{P} \neq \mathbf{NP}$.
- Relations: between computational problems (via reductions) and between resources (simulations). This is where complexity theory has had its greatest successes. Surprising and powerful relations include a general equivalence between approximate counting and random sampling (of witness/solutions to some **NP** problem), and an equivalence between the hardness of finding approximate solutions to **NP** optimization problems and super-fast probabilistic verification of mathematical proofs (the PCP Theorem).

# 2 Computational Model

**Reading:** Arora-Barak, Ch. 1. (Textbook by Sipser for more detail.)

## 2.1 Turing Machines

Although we can choose from any of many computational models, it is important to fix a precise model to compare complexities.
The Multi-Tape Turing Machine

- Has a read-only input tape, a write-only output tape,[1] and multiple read/write work tapes, each equipped with a head pointing to a particular position on the tape. Initially, the input (followed by blanks) is put on the first "input tape", and the rest of the tapes are empty. Initially, the heads point to the leftmost entry of the tape.
- Has a finite set of states, including a halt state.
- Has a transition function that, given the current state and the symbols the heads are currently reading, can change the symbols under the heads, move the heads position, and update the state.

Then, for a TM $M$ acting on input $x$, we have

$$M(x) = \begin{cases} \text{output tape contents} & M \text{ halts on input } x \\ \perp & \text{otherwise} \end{cases}$$

For example, $M$ decides a language $L$ if for every input $x \in \{0,1\}^*$,

$$x \in L \Rightarrow M(x) = 1$$
$$x \notin L \Rightarrow M(x) = 0.$$

[1] Arora–Barak allows the output tape to be read/write, but this messes up accounting of space complexity.

## 2.2 Basic Resources

*M runs in time* $T(n)$ if $\forall$ input $x \in \{0,1\}^*$, $M$ takes at most $T(|x|)$ steps (worst case complexity). This gives rise to complexity classes

- $\mathbf{DTIME}(T(n)) = \{L : L \text{ decided by some } M \text{ in time } O(T(n))\}$.
- $\mathbf{P} = \bigcup_c \mathbf{DTIME}(n^c)$.
- $\mathbf{EXP} = \bigcup_c \mathbf{DTIME}(2^{n^c})$ (complexity theorists' exponential time)
- *cf.,* $\mathbf{E} = \bigcup_c \mathbf{DTIME}(2^{cn})$ (cryptographers' exponential time)

*M uses space* $S(n)$ if $\forall$ input $x \in \{0,1\}^*$, the number of cells traversed by $M$ on its work tapes is at most $S(|x|)$. We obtain

- $\mathbf{L} = \bigcup_c \mathbf{SPACE}(c \log n)$
- $\mathbf{PSPACE} = \bigcup_c \mathbf{SPACE}(n^c)$

Allowing a TM $M$ to have *nondeterminism* (be a *NTM*) means having potentially many transitions available at each step, and we define $M$ to accept $x$ if there exists at least one accepting computation. This yield the classes $\mathbf{NTIME}(T(n))$, $\mathbf{NSPACE}(S(n))$, $\mathbf{NP}, \mathbf{NEXP}, \mathbf{NL}, \mathbf{NPSPACE}$ defined analogously as above.

While we believe that nondeterminism adds a lot of power to time, in particular $\mathbf{P} \neq \mathbf{NP}$, in a few weeks, we will see that this is not true for space — $\mathbf{PSPACE} = \mathbf{NPSPACE}$!

## 2.3 Church–Turing Thesis

There are two ways of expressing this statement:

Our intuitive notion of ...
Every physically realizable model of ... computation can be simulated by TMs

This is relatively uncontroversial; there have been no convincing challenges to the Church–Turing thesis.

However, what's more important for complexity theory than computability theory is the *Extended* Church-Turing Thesis, which states the same thing, *but with only a polynomial slowdown.* This justifies $\mathbf{P}$ as a model-independent (albeit coarse) notion of "efficient computation". But there are challenges to the Extended Church–Turing thesis: What if we consider randomized computation? Parallel computation? Or, most strikingly, Quantum computation (where factoring has a polytime algorithm)?

There are two possible answers...

- Even if the Extended Church–Turing thesis is false, $\mathbf{P}$ still captures very natural and important class of resource-constrained computation.
- Just stick "deterministic and sequential" in front of "computation can be simulated by TMs." Then randomization, parallelism, and quantum information can be thought of as additional resources to computation. By studying $\mathbf{P}$, we are just studying the basic form of computation. (These additional resources certainly merit study too, hence $\mathbf{BPP}$, $\mathbf{NC}$, $\mathbf{BQP}$,...)

## 2.4 Universal TM

**Theorem 1** *There exists TM $U$ such that for every TM $M$ and input $x \in \{0,1\}^*$, $U(\lfloor M \rfloor, x) = M(x)$, where $\lfloor M \rfloor$ is the description of $M$ as a string. Moreover, the running time for $U(\lfloor M \rfloor, x)$ is at most* $\text{poly}(|\lfloor M \rfloor|) \cdot \mathit{Time}_M(x)^2$ *and the space is at most $O(|\lfloor M \rfloor| + (\log |\Gamma_M|) \cdot \mathit{Space}_M(x))$, where $\Gamma_M$ is the tape alphabet of $M$. (Time blow-up quadratic; space blow-up linear.)*

**Improvements**

- $\text{Time}_M(x)^2 = T^2$ can actually be improved to be $T \log T$ via a much more sophisticated simulation.
- $U$ can be made *oblivious*, meaning that the locations of the heads of $U$ at each time step $t$ are independent of $U$'s input $\lfloor M \rfloor, x$, and depend only on $t$.

See Arora–Barak for sketches of how to achieve both of these properties.
Next time: hierarchy theorems!