CS221: Computational Complexity Prof. Salil Vadhan

Lecture 5: Space and Nondeterminism

September 30, 2002 *Scribe: Trevor Bass*

Contents

1 Agenda

- Space and Nondeterministic complexity classes
- Nondeterministic hierarchy theorem
- Relating time, space, and nondeterminism
- Reductions and completeness

2 Hierarchy Theorems, Continued

Theorem 1 (Almost everywhere hierarchy) *If f is a proper complexity function, there exists a language $L \in \mathbf{TIME}(2^{O(n)} f(n)^2)$ such that any M with $\mathcal{L}(M) = L$ (i.e. with no errors) takes time $> f(n)$ almost everywhere (i.e. L is "hard" a.e.).*

Last time we constructed L, but we didn't verify $L \in \mathbf{TIME}(2^{O(n)} f(n)^2)$. Where does the exponential factor come from? To decide L on input x, recall that we needed to look at all strings $y \leq x$ (lexicographically). Fully verifying the time bound is left as an exercise.

Corollary 2 *There exists a language $L \in \mathbf{EXP}(= \mathbf{TIME}(2^{n^{O(1)}}))$ such that deciding L takes time $> 2^n$ a.e.*

Another form of "L is hard often" introduces the notion of "average-case complexity": $\forall M$ running in time $f(n)$, the fraction of inputs on which $M(x) \neq \chi_L(x)$ is at least a constant δ asymptotically as $n \to \infty$. Noting that the all-1 or all-0 TM is correct 50% of the time asymptotically, we could ideally hope to have δ to be something like 49%.

3 Space Complexity Classes

Definition 3 $\mathbf{SPACE}(f(n))$ *is the set of languages decidable by a TM in space* $f(n)$.

Recall that "space" refers only to workspace, which does not include the input and output.

Definition 4 ("Log space") $\mathbf{L} = \bigcup_c \mathbf{SPACE}(c \log n)$.

Definition 5 ("Polynomial space") $\mathbf{PSPACE} = \bigcup_c \mathbf{SPACE}(n^c)$.

Log space is the smallest complexity class we will study in this course. Constant space can be hardwired into the finite state control of a TM (i.e. $\mathbf{SPACE}(O(1)) = \mathbf{SPACE}(0)$). In fact:

Remark 6 $\mathbf{SPACE}(\log \log n) = \mathbf{SPACE}(O(1)) = \mathbf{SPACE}(0) = \{\text{regular languages}\}$.

Notice that $\log n$ bits are needed even to keep a pointer in the input or to calculate the size of the input. $\mathbf{L}$ is thus a good measure of feasible computation for massive data sets: with it one can keep track of a constant number of pointers, but cannot do too much more.

The following operations are typical of problems in $\mathbf{L}$: addition (trivial to prove this), multiplication (on problem set 1), and division (difficult—solved in 2001).

On the other hand, $\mathbf{PSPACE}$ is much more powerful; typical problems are games (e.g. does player 1 have a winning strategy from a given position in some game?), such as go, chess, and geography.

On problem set 1, we will see that there is a hierarchy theorem for space. A corollary to this theorem is:

Corollary 7 $\mathbf{L} \neq \mathbf{PSPACE}$.

4 Nondeterminism

Recall that a nondeterministic TM (NTM) is just like a deterministic TM except that at any time step there can be several options for the possible next states, what it writes, and so on, such that the transition function becomes a transition relation. Thus on a given input, there can be many different possible computations, corresponding to all possible nondeterministic choices.

Definition 8 *An NTM* M accepts x *if there exists a computation of* M *on* x *that halts and outputs 1. Similarly, its language* $\mathcal{L}(M) = \{x : \exists$ *computation of* M *on* x *that halts and outputs* $1.\}$ *(i.e. "guess and verify").*

The running time of an NTM on x is the maximum running time used on input x over all computations, even those that do not output 1, and is denoted $\mathbf{NTIME}(f(n))$. The space used on x is the maximum workspace used on x over all computations, and is denoted $\mathbf{NSPACE}(f(n))$. Note that if the NTM doesn't halt on a certain set of choices, it can be redefined by limiting all computations to a particular running time. Analogous to the deterministic class we define $\mathbf{NP} \stackrel{\text{def}}{=} \bigcup_c \mathbf{NTIME}(n^c)$, $\mathbf{NEXP} = \bigcup_c \mathbf{NTIME}(2^{n^c})$, $\mathbf{NL} = \bigcup_c \mathbf{NSPACE}(c \log n)$, and $\mathbf{NPSPACE} = \bigcup_c \mathbf{NSPACE}(2^{n^c})$.

Nondeterminism is not a realistic/physical resource. Then why study it? It turns out that it captures many important computational problems, and helps us understand more realistic resources, such as **TIME** and **SPACE**.

Recall an alternate characterization of **NP**:

Proposition 9 *$L \in$ **NP** iff there exists a relation $R \subset \Sigma^* \times \Sigma^*$ such that*

(1) $x \in L$ iff $\exists y$ such that $(x, y) \in R$ (y is called a "witness" for x),

*(2) deciding R is in **P** ("polynomially decidable"), and*

(3) $\exists$ fixed polynomial p such that if $(x, y) \in R$ then $|y| \leq p(|x|)$ ("polynomially balanced").

Proof: $\Leftarrow$: The nondeterministic choices allow witnesses (of polynomial length) to be guessed, and $(x, y) \in R$ can b checked in polynomial time.

$\Rightarrow$: (Sketch) Define R such that witnesses y correspond to the sequence of nondeterministic choices of the NTM M deciding L and $R(x, y) = 1$ is $M(x)$ accepts with nondeterministic choices y. ■

4.1 Examples of Problems in NP

- SAT $= \{\varphi(x_1, \ldots, x_n) : \varphi$ is a propositional formula such that $\exists a = (a_1, \ldots, a_n) \in \{0,1\}^n$ with $\varphi(a) = 1\}$. (Note: φ can be evaluated in polynomial time on a given assignment, so such a witness is checkable in polynomial time.)
- CLIQUE $= \{(G, k)$: graph G has complete subgraph on k vertices$\}$. (Note: The completeness of the witness, a complete subgraph if it exists, is checkable in polynomial time.)
- GRAPH-ISOMORPHISM $= \{(G_1, G_2) : G_1$ is the "same" as G_2 up to relabeling$\}$.
- SHORT-PROOF $= \{(\varphi, 1^n) : \varphi$ is a mathematical theorem with a proof of length $\leq n\}$. (Note: The proof and theorem are in ZFC set theory, say.)

4.2 Consequences of P=NP

- Mathematicians can be replaced by algorithms (pointed out in a letter from Gödel to von Neumann in 1955).
- All reasonable search problems can be solved efficiently, which would make Artificial Intelligence and computational learning dramatically easier.
- All optimization problems (such as chip designs to minimize power usage) can be solved efficiently.
- Cryptography falls apart, since schemes become easier to break.
- Randomized algorithms are not substantially more efficient than deterministic ones.

Determining whether **P** = **NP** is viewed as one of the most important questions in math and science, and as a testament it is one of the million-dollar questions sponsored by the Clay Institute of Mathematics.