Lecture #8: Introduction to Nondeterministic Computation
Exercises and Review

Additional Exercises

1. Consider a language $L \subseteq \Sigma^\star$, for an alphabet Σ, such that $L \in \mathcal{P}$. We may associate with this another language $\widehat{L} \subseteq \{1\}^\star$ such that, for every non-negative integer n, the string 1^n belongs to $\widehat{L}$ ***if and only if*** there exists at least one string $\omega \in \Sigma^\star$ such that $|\omega| = n$ and $\omega \in L$.

   Prove that $\widehat{L} \in \mathcal{NP}$.

2. Recently, some authors have described a more limited kind of nondeterministic Turing machine: We will say that a nondeterministic k-tape Turing machine

   $$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

   is a ***restricted nondeterministic Turing machine*** if $|\delta(q, \sigma_1, \sigma_2, \ldots, \sigma_k)| \leq 2$ for every state $q \in Q$ and for all tape symbols $\sigma_1, \sigma_2, \ldots, \sigma_k \in \Gamma$. In effect, this means that the Turing machine can only nondeterministically guess the answer to a Yes-or-No question every time it takes a step, and its computation trees can be thought of as ***binary trees***.

   Explain why the complexity classes NTIME(f) — for time-constructible functions f — and $\mathcal{NP}$ would not be changed if these complexity classes were defined using restricted nondeterministic Turing machines, instead of the (more general) nondeterministic Turing machines that have been introduced in this course.

Questions for Review

1. This question concerns ***nondeterministic Turing machines***.

   (a) Give the definition of a nondeterministic Turing machine.

   (b) How is a nondeterministic Turing machine different from the "deterministic" Turing machines that have been used, in this course, before this?

   (c) Describe how the computation of a nondeterministic Turing machine M on an input string can be represented as a ***tree*** of configurations (rather than as a sequence of them).

   (d) If a nondeterministic Turing machine M has input alphabet Σ, what does it mean for M to ***accept*** a string $\omega \in \Sigma^\star$?

   (e) If a nondeterministic Turing machine M has input alphabet Σ, what does it mean for M to ***recognize*** a language $L \subseteq \Sigma^\star$?

   (f) If a nondeterministic Turing machine M has input alphabet Σ, what does it mean for M to ***decide*** a language $L \subseteq \Sigma^\star$?

   (g) Define the ***time*** used by a nondeterministic Turing machine M (with input alphabet Σ) on an input string $\omega \in \Sigma^\star$.

   (h) If M is a nondeterministic Turing machine with input alphabet Σ, $L \subseteq \Sigma^\star$, and $f : \mathbb{N} \to \mathbb{N}$, then what does it mean for M to ***decide*** the language L in ***time*** f?

   (i) For a function $f : \mathbb{N} \to \mathbb{N}$, give a definition of NTIME(f) using the above.

2. This question concerns ***verification***.

   (a) Give the definition of a ***verifier*** of a language L.

   (b) What is the ***certificate alphabet*** of a verifier?

   (c) What is a ***certificate*** for a string $\omega \in L$? (This also depends on some "verifier" M for L.)

   (d) Describe what it means for a language $L \subseteq \Sigma^\star$ to be in NTIME$_{\mathsf{V}}(f)$, for a function $f : \mathbb{N} \to \mathbb{N}$.

3. Describe, as precisely as you can, how the complexity classes NTIME(f) and NTIME$_{\mathsf{V}}(f)$ are related. Don't forget any ***assumptions*** about f that are made, here!

4. What is $\mathcal{NP}$?

5. Describe, as precisely (and in as much detail) as you can what is known about the relationship between *deterministic* time-complexity classes and *nondeterministic*-time complexity classes.

6. Describe what else is generally *believed* about the relationship between deterministic-time complexity classes and non-deterministic-time complexity classes, but not proved.