CMPUT 675: Computational Complexity Theory **Winter 2019**

Lecture 3 (Jan 15): The Class **NP**

Lecturer: Zachary Friggstad *Scribe: Joshua Teitz*

3.1 NP

Definition 1 *A turing machine M is a polynomial-time verifier (PTV) if it reads two inputs (x, y) and for all input pairs $(x, y) \in \{0,1\}^*$, M halts after $p(|x|)$ steps for some polynomial $p(n)$.*

Note that the running time of a verifier only depends on the first input x.

Definition 2 *A language L is decidable by a PTV M with running time $p(n)$ if for all $x \in \{0,1\}^*$,*

1. *if $x \in L$, there exists $y \in \{0,1\}^{p(|x|)}$ such that M accepts (x, y) and*
2. *if $x \notin L$, for every $y \in \{0,1\}^*$, M rejects x, y.*

Definition 3 *The class* **NP** *is all languages decided by a PTV.*

3.1.1 Relations between P, NP and PSPACE

Theorem 1 **P $\subseteq$ NP $\subseteq$ PSPACE**

Proof.

P $\subseteq$ NP.

Fix $L \in \mathbf{P}$. Let M be a poly-time TM that decides L. Define M' as the PTV that simulates $M(x)$ for every input pair $x, y \in \{0,1\}^*$. Fix $x \in \{0,1\}^*$. Observe that if $x \in L$, then M' accepts (x, y) for any y, and if $x \notin L$, then M rejects (x, y) for every $y \in \{0,1\}^*$. So M' is a PTV that decides L and thus **P** $\subseteq$ **NP**.

NP $\subseteq$ PSPACE.

Fix $L \in$ **NP** and a PTV M with running time $p(n)$ that decides L Define a TM $M'(x)$ that iteratively simulates M. In each iteration, M' picks the "next" bit string y from an emumeration of $\{0,1\}^{p(|x|)}$ and simulates $M(x, y)$. If M accepts (x, y), then M' accepts x. Otherwise, M' proceeds to the next iteration.

As observed in lecture 2, the calculation $M(x, y)$ runs in time $p(|x|)$ so it uses polynomial space for each y. The different y can also be enumerated using $O(p(|x|))$ space with a simple counter. Thus, M' uses $O(p(|x|))$ space overall. Thus $L \in$ **PSPACE** and so **NP** $\subseteq$ **PSPACE**.

■

3.1.2 Non-deterministic Turing Machines

Definition 4 *A non-deterministic Turing machine (NDTM) M is defined the same way as a regular Turing machine except that it has two transition functions δ_0 and δ_1. At each step of the computation, M arbitrarily chooses between δ_0 and δ_1.*

Definition 5 *A language L is decided by a NDTM M if for all $x \in \{0,1\}^*$, M eventually halts for every sequence of transition functions*[1]*, and*

1. *if $x \in L$, there exists some sequence of transitions that ends in q_{accept} and*
2. *if $x \notin L$, every sequence of transitions ends in q_{reject}.*

Definition 6 *A NDTM M has running time $p(n)$ if for every $x \in \{0,1\}^*$, all sequences of transitions stop within $p(|x|)$ steps.*

Definition 7 $\mathbf{NTIME}(f(n))$ *is all languages decidable by a NDTM with running time $O(f(n))$.*

Theorem 2 $\mathbf{NP} = \bigcup_{c \geq 1} \mathbf{NTIME}(n^c)$

Proof.

$\bigcup_{c\geq 1} \mathbf{NTIME}(n^c) \subseteq \mathbf{NP}$.

Fix $L \in \bigcup_{c\geq 1} \mathbf{NTIME}(n^c)$ and a NDTM M that decides L in time $p(n)$ for some polynomial p. For all $x \in \{0,1\}^*$ and $y \in \{0,1\}^{p(n)}$, define M' as the PTV that simulates $M(x)$ with transition sequence defined by y. That is, in the i'th step the simulation of $M(x)$ should use transition function δ_{y_i}. M' accepts x iff M accepts x on the sequence defined by y. So $L \in \mathbf{NP}$ and thus $\bigcup_{c\geq 1} \mathbf{NTIME}(n^c) \subseteq \mathbf{NP}$.

$\mathbf{NP} \subseteq \bigcup_{c\geq 1} \mathbf{NTIME}(n^c)$.

Fix $L \in \mathbf{NP}$ and let M be a PTV that decides L. Fix $x \in \{0,1\}^*$. Define a NDTM M' that uses its non-determinism to create a bit string y of length $p(|x|)$. More specifically, M' generates y by writing $p(|x|)$ bits where each bit will be a 0 if δ_0 is used when generating the bit or a 1 if δ_1 is used when generating the bit. Then M' then simulates M on input (x, y). M' accepts x iff M' can generate $y \in \{0,1\}^{p(|x|)}$ such that M accepts (x, y). Since M is a PTV and it takes $p(|x|)$ transitions to generate $y \in \{0,1\}^{p(|x|)}$, $L \in \bigcup_{c\geq 1} \mathbf{NTIME}(n^c)$. So $\mathbf{NP} \subseteq \bigcup_{c\geq 1} \mathbf{NTIME}(n^c)$.

■

3.2 Reducibility and NP-completeness

Definition 8 *$L \subseteq \{0,1\}^*$ is poly-time Karp reducible to $L' \subseteq \{0,1\}^*$ if there exists a poly-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that $x \in L$ iff $f(x) \in L'$. We write $L \leq_p L'$ to mean that L is poly-time reducible to L'.*

[1] Meaning there is no infinite sequence of transitions that does not enter a halting state when starting from the initial snapshot for x.

Definition 9 *L' is* **NP***-hard if for all $L \in$* **NP***, $L \leq_p L'$.*

Definition 10 *L' is* **NP***-complete if L' is* **NP***-hard and $L' \in$* **NP**

Theorem 3 *For any languages L, L', L'',*

1. *$L \leq_p L'$ and $L' \leq_p L'' \Rightarrow L \leq_p L''$*
2. *If L is* **NP***-hard and $L \in$* **P***, then* **P** = **NP**.
3. *If L is* **NP***-complete, then $L \in$* **P** *iff* **P = NP**.

Proof.

1. Fix $x \in \{0,1\}^*$. Since $L \leq_p L'$ and $L' \leq_p L''$, there exists poly-time computable f and g such that $x \in L$ iff $f(x) \in L'$ and $f(x) \in L'$ iff $g(f(x)) \in L''$. So $x \in L$ iff $(g \circ f)(x) \in L''$. And since the composition of poly-time computable functions remains poly-time computable, $L \leq_p L''$.
2. Suppose L is **NP**-hard and $L \in$ **P**. Let M be a poly-time TM deciding L. By Theorem 1, **P** $\subseteq$ **NP**. Let $L' \in$ **NP**. Since L is **NP**-hard, $L' \leq_p L$. Say f is the reduction showing $L' \leq_p L$. To decide if some string $x \in L$, a TM could first compute $f(x)$ and then output the result of the computation $M(f(x))$. In this way, L' is be decided in polynomial time. Thus, **NP** $\subseteq$ **P**.
3. Suppose L is **NP**-complete.

   Suppose $L \in$ **P**. The previous part then shows **P** = **NP**. Now suppose **P** = **NP**. Since L is **NP**-complete, $L \in$ **NP**. And since **P** = **NP**, $L \in$ **P**.

■

3.2.1 Example of an NP-complete language

Definition 11 TMSAT $= \{(\alpha, x, 1^n, 1^t) :$ there exists $u \in \{0,1\}^n$ such that $M_\alpha(x,u)$ accepts within t steps$\}$

Note this definition assumes we can enumerate all verifiers using strings $\alpha \in \{0,1\}^*$ much like we can enumerate all TMs. This is easily doable, just like it is with TMs.

Theorem 4 TMSAT *is* **NP***-complete.*

Proof. TMSAT $\in$ **NP**: a verifier M for TMSAT expects, when given input $(\alpha, x, 1^n, 1^t)$, a string $u \in \{0,1\}^n$ as its second input. Then M simulates M_α on input (x,u) for t steps and accepts if and only if this simulation accepted.

To prove TMSAT is **NP**-hard, fix $L \in$ **NP** and $x \in \{0,1\}^*$. So there exists a PTV M_α with running time $p(n)$ such that $x \in L$ iff there exists $y \in \{0,1\}^{p(|x|)}$ such that M_α accepts (x,y). Note, x can be mapped in polynomial time to $(\alpha, x, 1^{p(|x|)}, 1^{p(|x|)})$. In this way, $(\alpha, x, 1^{p(|x|)}, 1^{p(|x|)}) \in$ TMSAT iff there exists $u \in \{0,1\}^{p(|x|)}$ such that $M_\alpha(x,u)$ accepts within $p(|x|)$ steps. This occurs iff $x \in L$. So $L \leq_p$ TMSAT and thus TMSAT is **NP**-hard. Therefore, TMSAT is **NP**-complete. ■