CSC 2401, Introduction to computational complexity
Lecture #6, Tuesday, October 2^{nd}, 2001
Lecturer: Allan Borodin
Scribe: Anastasia Bezerianos

1 Non deterministic Turing Machines (NTM) Continued

Since we have the notion of NTM running in time $t(n)$ (space $s(n)$) we immediately have the notion of $NTIME(t(n))$ (respectively $NSPACE(s(n))$) classes.

Note: As in deterministic TM we can have time speedup and space compression for NTM.

2 Non Deterministic Complexity Classes

Analogous to deterministic cases:

$$\begin{aligned} NP &= \bigcup_k NTIME(n^k) \\ N\mathcal{L} = NLOG &= \bigcup_c NSPACE(c\ \log n) = NSPACE(\log n) \\ NPSPACE &= \bigcup_k NSPACE(n^k) \end{aligned}$$

We will not often see $NPSPACE$, because as we will show $PSPACE = NPSPACE$.

Simple Facts:

$$NTIME(t(n)) \subseteq DSPACE(t(n)) \subseteq NSPACE(t(n)) \subseteq \bigcup_c DTIME(c^{t(n)})$$

Observations:

1. When executing a NTM $\mathcal{M}$ on an input w, we obtain a computation tree where each path corresponds to a legal sequence of configurations (since at every configuration there can be more than one possible next configurations).

2. At each state we assume a maximum (finite) number d of choices (an upper bound of next configurations).

   A particular path in the computation tree is described by a sequence $\in \{1, ..., d\}^{t(n)}$.

3. To show that $NTIME(t(n)) \subseteq DSPACE(t(n))$:
   Do an exhaustive search of each path in the tree using an additional tape to determine and sequence through all the path choices. Each configuration is at most $t(n)$ long.

Aside: Later we will show

$$NSPACE(s(n)) \quad \subseteq \quad DSPACE(s^2(n))$$

In order to show that $NSPACE(s(n)) \subseteq \bigcup_c DTIME(c^{s(n)})$,
let's consider again how many possible configurations we can have.
Given $\mathcal{M}$ and input w, the number of all possible configurations is:

$$|Q| \cdot n \cdot s(n) \cdot \tilde{c}^{s(n)} = \tilde{c}^{\mathcal{O}(s(n))} = c^{s(n)}$$

From a given $(\mathcal{M}, w)$ we want to produce a directed graph:

$$\mathcal{M}, w \to G_{\mathcal{M},w} = (V, E)$$

where

$$\begin{aligned} V &= \{\text{all possible configurations}\} \\ E &= (u,v) \in E \text{ iff } u \underset{\mathcal{M},w}{\mid\!\to} v \end{aligned}$$

In other words, if u, v are possible configurations of the NTM $\mathcal{M}$ they are nodes in G. If in one step we go from configuration u to v with NTM $\mathcal{M}$ ($u \underset{\mathcal{M},w}{\mid\!\to} v$), then there is an edge e between u, v in G.
Without loss of generality we can assume that $\mathcal{M}$ has a unique accepting configuration u_f (i.e. when reaching an accepting state, $\mathcal{M}$ erases all its work tapes and moves into a designated final state).
So we get the equivalent two statements:

$$w \in \mathcal{L} \begin{cases} \text{iff} & \exists \text{ a computation path from initial configuration } u_0 \text{ to the accepting configuration } u_f \\ \text{iff} & \exists \text{ directed path in } G_{\mathcal{M},w} \text{ from } u_0 \text{ to } u_f \end{cases} \tag{1}$$

By a standard graph traversal algorithm (eg. Dijktra's algorithm) we can determine if such a directed path exists in $|E| \cdot \log |V|$ steps, where

$$\begin{aligned} |E| &= d \cdot c^{s(n)} \\ |V| &= c^{s(n)} \end{aligned}$$

Claim: Given $\mathcal{M}, w$, can construct $G_{\mathcal{M},w}$ in logspace (and hence polytime) (log of max{input,output } length). $\square$
We will soon see that $NPSPACE = PSPACE$. Given the above, we restate the complexity class inclusions:

$$\underbrace{D\mathcal{L} \subseteq N\mathcal{L} \subseteq P \subseteq NP \subseteq PSPACE}_{\text{bulk of complexity theory attention}} \subseteq \bigcup_k DTIME(2^{n^k})$$

EVERY ONE OF THE ABOVE INCLUSIONS "BELIEVED" TO BE PROPER BUT NO INCLUSION HAS BEEN PROVEN TO BE PROPER.

$D\mathcal{L} \subseteq N\mathcal{L}$**:** Although the conjecture (and prevailing opinion) is that this inclusion is proper, there is some "evidence" that makes us question this conjecture and it might very well be that this inclusion could turn out to be an equality. (See the next lecture.)

$N\mathcal{L} \subseteq P$**:** This inclusion is certainly believed to be proper.

$P \subseteq NP$**:** This is for most people THE fundamental complexity issue and popular opinion strongly believes that this inclusion is proper. The consequences of $P = NP$ would be dramatic to say the least.

3 Source to Target Connectivity Problem

We define *Source to Target Connectivity Problem*:

$$\begin{aligned} STCON &= \{(G, u, v) \mid G \text{ is a directed graph and} \\ &\quad \exists \text{ directed path from node } u \text{ to node } v\} \end{aligned}$$

Determining if there exist a path from some source node u to target node v is a directed graph can be easily seen to be computed in non deterministic log space (by guessing a path).

In more modern terms: Let $STCON \in N\mathcal{L} = NSPACE(\log n)$.

4 Reducibilities

Definition: $L_1 \leq_p L_2$ (respectively, $L_1 \leq_{logspace} L_2$) if $\exists$ poly time computable (respectively, logspace computable) function f such that:

$$f : w \in L_1 \text{ iff } f(w) \in \mathcal{L}_2$$

Fact: $L_1 \leq_{logspace} L_2 \Longrightarrow L_1 \leq_p L_2$

Nomenclature We can restate what we have :
Let $L \in N\mathcal{L}$, then $L \leq_{logspace} STCON$. That is, $STCON$ is *hard* for $N\mathcal{L}$ wrt $\leq_{logspace}$.

More generally, whenever we have a class of languages $\mathcal{C}$ and a "reducibility" $\leq_R$, we can define the concept of a language $\mathcal{L}$ being $\leq_R$-*hard* for $\mathcal{C}$ or $\mathcal{C}$-*hard* (wrt to $\leq_R$).

L is $\leq_R$-*complete* for $\mathcal{C}$ (or $\mathcal{C}$-*complete* wrt $\leq_R$) if

1. $L \in \mathcal{C}$
2. L is $\leq_R$-*hard* for $\mathcal{C}$

When the reduction is understood, we simply say that L is $\mathcal{C} - complete$ (as in $NP - complete$).

Fact: STCON is logspace-complete for $N\mathcal{L}$ ($N\mathcal{L}$-complete wrt $\leq_{logspace}$)

Soon: SAT is NP-complete (wrt $\leq_{logspace}$ and hence wrt $\leq_P$).

What properties do we want for a reducibility $\leq_R$?

1. $\leq_R$ reflexive
2. $\leq_R$ transitive
3. $L_1 \leq_R L_2$ and L_2 is "easy", then L_1 is "easy" ; in particular, $\mathcal{C}$ is closed under $\leq_R$.

Claim: Both $\leq_P$ and $\leq_{logspace}$ have the desired reducibility properties.

Intuition behind claim:

A For $\leq_p$

1. Very obvious
2. If

$$\left.\begin{array}{lll} \mathcal{L}_1 \leq_P \mathcal{L}_2 & \text{via} & f \\ \mathcal{L}_2 \leq_P \mathcal{L}_3 & \text{via} & g \end{array}\right\} \Longrightarrow \mathcal{L}_1 \leq_P \mathcal{L}_3 \text{ via } g \circ f$$

3. "easy" $\approx NP$ (or something smaller like P or RP = randomized poly time)

B For $\leq_{logspace}$

1. Obvious

2. Deserves a little thought. We want

$$\left.\begin{array}{lll} L_1 \leq_{logspace} L_2 & \text{via} & f \\ L_2 \leq_{logspace} L_3 & \text{via} & g \end{array}\right\} \Longrightarrow L_1 \leq_{logspace} L_3 \text{ via } g \circ f$$

   We cannot get $f(w)$ and put it on a tape to compute it immediately. Instead, when we need an input of g we recompute $f(w)$, so one symbol is used for g each time. This does not require more than $\mathcal{O}(log)$ squares on the worktape. Thus logspace transformations are also transitive.

3. "easy" $\approx P$ or something smaller like $N\mathcal{L}$ or DŁ. If L is logspace complete (like $STCON$) for $N\mathcal{L}$ and $L \in D\mathcal{L}$, then $N\mathcal{L} \subseteq D\mathcal{L}$.