Complexité avancée - Homework

Benjamin Bordais

September 29, 2021
Due October 6, 2021

**NL alternative definition**

A Turing machine with *certificate tape*, called a verifier, is a deterministic Turing machine with an extra read-only input tape called *the certificate tape*, which moreover is read once (i.e. the head on that tape can either remain on the same cell or move right, but never move left). A verifier takes as input a word $x$, along with a word $u$ written in the certificate tape.

Define $\text{NL}_{\text{certif}}$ to be the class of languages $L$ such that there exists a polynomial $p : \mathbb{N} \rightarrow \mathbb{N}$ and a verifier $V$ running in logarithmic space such that:

$$x \in L \iff \exists u, |u| \leq p(|x|) \text{ and } V \text{ accepts on input } (x, u).$$

1. Show that $\text{NL}_{\text{certif}} = \text{NL}$.

2. What complexity class do you obtain if you remove the read-once constraint in the definition of a machine with certification tape? Justify your answer. You may use the fact that SAT is NP-complete for logspace reductions.

**Solution:**

1. $\text{NL}_{\text{certif}} \subseteq \text{NL}$: Let $A$ be a language in $\text{NL}_{\text{certif}}$. If $V$ is a verifier for $A$, we set a counter to $p(|x|)$ and then simulate the run of $V$ on an input $x$ by guessing the next letter of $u$ and decrementing the counter and accepting $x$ if and only if $V$ would have accepted on $(x, u)$ (if the counter has not reached 0). This runs in logarithmic space. Indeed, $V$ runs in logarithmic space and we only need to remember one letter of $u$ at a time, because the certificate tape is read only once.

$\text{NL} \subseteq \text{NL}_{\text{certif}}$: Let there be $A$ in $\text{NL}$, and $M$ a non deterministic Turing Machine running in logarithmic space and recognizing $A$. For a given input $x$, a run of $M$ on $x$ can be characterized by all the non deterministic choices done in its execution tree, which can be encoded in a word $y$ over $\{0,1\}^*$ with $|y| \leq p_M(|x|)$ for some polynomial function $p_M$ bounding the running time of $M$ (which exists since $M$ runs in logarithmic space hence polynomial time). Note that this polynomial function $p_M$ does not depend on the input, but only on the Turing Machine $M$ (specifically, its number of states, the size of its alphabet, and its number of work tapes). We can then construct a verifier $V$ simulating $M$ that consults its certificate tape each time $M$ uses non determinism. The runs of $V$ are deterministic and require logarithmic space.
2. We call the new class $\text{NL}^{\prime}_{\text{certif}}$.

$\text{NL}^{\prime}_{\text{certif}} \subseteq \text{NP}$: Let there be $A$ in $\text{NL}^{\prime}_{\text{certif}}$ and consider a verifier $V$ for $A$. We build a non-deterministic Turing Machine $M$ running in polynomial time accepting $A$. First, $M$ guesses the certificate (which is polynomial in the size of the input), and stores it in a worktape since perhaps it will have to be read several times. And then, $M$ runs $V$ on $(x, u)$, which takes polynomial time since a logspace verifier like $V$ runs in polynomial time.

$\text{NP} \subseteq \text{NL}^{\prime}_{\text{certif}}$: First, let us show that $\text{SAT} \in \text{NL}^{\prime}_{\text{certif}}$. Consider a propositional formula $\varphi$. We build a verifier $V$ that checks if the certificate given in the certification tape that gives a boolean valuation $(a_1, \ldots, a_n)$ of the variables satisfies the formula $\varphi$. Specifically, $V$ loops over all the clauses and over the literals in these clauses. The truth value of a literal is given by the certificate tape, that $V$ checks to see the value of the literal. If one literal is satisfied by the valuation, $V$ goes to the next clause, otherwise another literal is checked. If no literal is satisfied, $V$ rejects. If all clauses are satisfied, $V$ accepts. This takes logarithmic space since we only need a pointer on the clause and literal we are considering. Note that the certificate tape is consulted each time a literal is evaluated (which would not be possible in the read-only-once setting).

Let us now show that $\text{NL}^{\prime}_{\text{certif}}$ is closed under logspace reduction, that is if $L \in \text{NL}^{\prime}_{\text{certif}}$ and $L'$ reduces in logspace to $L$, then $L' \in \text{NL}^{\prime}_{\text{certif}}$. Let us denote by $f$ the reduction from $L'$ to $L$ that can be computed in logarithmic space and by $V$ a verifier accepting $L$. We build a verifier $V'$ running in logspace accepting $L'$. On an input $w$, $V'$ simulates the verifier $V$ on $f(w)$. $f(w)$ can be constructed in logarithmic space, however, as in the proof of Lemma 2.11 in the course, it is not possible, a priori, to copy $f(w)$ on a work tape since the space taken may exceed the logarithmic space bound. Instead, each time a bit of $f(w)$ is required by the simulation of $V$, it is computed once again in logarithmic space. This ensures that the space used is in $O(\log(|w|))$. Hence, we use the same trick than for the speed-up theorem. Then, as $f$ is computable in logspace, it is also in polynomial time. That is, $|f(w)| = O(|w|^c)$ for some $c \geq 0$. That is, the space used by $V$ on $f(w)$ is in $O(\log(f(|w|))) = O(c \cdot \log(|w|)) = O(\log(|w|))$. Hence, we obtain a verifier $V'$ running in logarithmic space accepting $L'$ (with the speed up theorem).

We conclude that $\text{NP} \subseteq \text{NL}^{\prime}_{\text{certif}}$ by using the fact that $\text{SAT}$ is $\text{NP}$-complete for logarithmic space reductions.