

Homework 1

1.1 A single-tape Turing machine M is called predictable if its head moves in the same way for every input word (formally: if there exists a function $\text{pos} : \mathbb{N} \rightarrow \mathbb{N}$ such that after k steps of every run of M , regardless of the input word, the head is over the tape cell number $\text{pos}(k)$). Is it the case that for every nondeterministic Turing machine M there exists a predictable nondeterministic Turing machine M' recognizing the same language as M and working at most polynomially slower?

Solution

For given non-deterministic machine M I will construct predictable machine N that simulates M . After each M 's turn we do following things:

- go back to beginning of tape (TOLEFT phase)
- go right while not on blank (TORIGHT phase)
- put special char $\#$ at blank*

* - there is a case, that instead of $\#$ we put legal character. It happens when simulated head is at right-most position and we simulate right-transition. Detecting such situation is feasible and described below.

Let's observe, that presented movement schema is predictable: we can easily compute $\text{pos}(k)$. During each cycle like above, we do find M 's head and simulate transition. Let's expand initial alphabet with special circum character ($\hat{}$), that marks M 's head position during simulation. Transition simulating description: During TOLEFT phase we are seeking for head, when found we check whether transition is to left or right. If it's left, we do it immediately, switching state and letter on left cell.

TORIGHT phase is similar: detecting head and right transition we emulate it. There is a small caveat, associated with *-annotation above: we need to remember in state fleeting information, whether transition was computed one step ago. If it is the case and we see blank, then we stop TORIGHT without putting $\#$ character (instead we put legit character and tape length is proper.)

Indeed, presented N machine is predictable and at most polynomially slower and recognizes same language that M .

1.2 A function $f: \Sigma^* \rightarrow \Gamma^*$ is called a morphism if $f(w \cdot v) = f(w) \cdot f(v)$ for all words $w, v \in \Sigma^*$ (the symbol “ \cdot ” denotes concatenation of words). A morphism f is nonabbreviating if $|f(w)| \geq |w|$ for all $w \in \Sigma^*$ (i.e., f does not decrease the length of words). For a set of words $L \in \Sigma^*$ we define $f(L) = \{f(w) | w \in L\}$. We say that a class C is closed under images of nonabbreviating morphisms if for every L in C , and for every nonabbreviating morphism f , also $f(L)$ belongs to C . Prove that the complexity class P is closed under images of nonabbreviating morphisms if and only if $P = NP$.

Solution

1. $P = NP \Rightarrow P$ is closed

Let's consider any language $L \in P$ and any nonabbreviating morphism f . If we show, that there exists non-deterministic, polynomial time Turing machine that recognizes language $f(L)$, then based on assumption that $P=NP$ we will show implication.

Let's observe, that by definition, if f is morphism, then $f(ab) = f(a) \cdot f(b)$, where $a, b \in \Sigma^1$ (each letter is transformed independently of context). For all $g: \Sigma^* \rightarrow \Gamma^*$ let's define $g': \Sigma \rightarrow \Gamma^*$ as a finite-domain function upon single letters from finite alphabet ($g'(a) = g(a)$).

Let's define a Turing machine running over word $f(w)$, that is guessing proper letter-split-position (it is non-deterministic, because f -images of each letter may overlap). After guessing split position, it checks, whether a infix between last and actual splitting positions belongs to f' image and if it's the case, it writes down on a separate tape a letter that is inverse image of that infix, and if it isn't, it just finishes run with fail. Reaching end of word we are having set of proper input word inverse images, we can now easily check for each whether it belongs to language L we started with (there exists proper P -time machine).

2. P is closed $\Rightarrow P = NP$

Let's observe, that we can prove it, if we find $L \in P$ and morphism f , such that $f(L)$ is NP-complete. Let's consider problem of logical formula evaluation and it's word-representation *formula#values_to_check*. This problem belongs to P class (simply substitute given values). We would like to find such f , that it transforms evaluation problem to SAT problem, which is NP-complete. It's rather simple, we just abandon values, while transforming formula representation 1:1. To make it nonabbreviating, we substitute values by $\#$'s, and obtain word *formula#####....*. There exists NP-machine recognising transformed language, as SAT problem (returns true iff formula is satisfiable). Thus we proved implication.