Computational Complexity
Mateusz Biegański
mb385162

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 pos : N N such that after k steps of every run of M, regardless of the input word, the head is over the tape cell number 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. First, our machine goes right-most, and when it reaches blank it swaps it with special #, then starts iterating. At the beginning of each iteration N's head is right-most and we generally do following things:

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

Let's observe, that presented movement schema is predictable: we can easily compute pos(k). During each cycle like above, we do find M's head and simulate transition. Let's expand each letter of initial M's alphabet with special circum character (ˆ), that marks M's head position during simulation.

Transition simulating description:

During TOLEFT phase we are seeking for a head, when found we check whether transition is to left or right. If it's left, we simulate it, switching state and writing proper letter. Let's observe, that letter under head is decorated (ˆmarks head). Assuming that it is $\hat{a}$, we do M's transition for a letter seen. It is also necessary to remember one bit of information in state, whether we did transition a moment ago, for setting new head up next turn.
TORIGHT phase is similar: detecting head and right transition we emulate it. Also, we handle fixed-head transitions here (might be handled also in TOLEFT, no difference) easily (just $\hat{a} \to \hat{b}$ for $a \to b$ M's transitions). Meeting padding # characters we omit them and go right, meeting blank we put # and finish.

During single TOLEFT phase there may be multiple transitions simulated, providing each moves head to left. Similarly with TORIGHT. # character is treated as a blank for transitions, but we use it for predictability, not to finish TORIGHT phase too early.

Let's take any run of M, that takes k moves (thus memory range is also limited by k). N machine does at most 2k * (k + 1) moves to simulate that run.

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

1.2 A function $f\colon \Sigma* \to \Gamma*$ 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 $f(ab) = f(a) \cdot (b)$, where $a, b \in \Sigma^1$ (each letter is transformed independently of context). For all $g\colon \Sigma* \to \Gamma*$ let's define $g'\colon \Sigma \to \Gamma*$ 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 ad 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 jut 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 satisfable). Thus we prooved implication.