Problem-set 02

Mostafa Touny

Contents

Lectures: Sipser 3: Ch. 3.2, 7.1 (ignore nondeterminism). 4: Ch. 7.2 (up to "examples of problems in P"), 4.2 (ignore "Turing unrecognizability").

Ex. 1

The set of all unary languages is uncountably infinite but the set of all Turing machines is countably infinite, Hence some unary language isn't recognizable by any turing machine.

Let's prove uncountability of unary languages. Observe the set Σ^* equals

ϵ	s_1
1	s_2
11	s_3
111	s_4
..	..

Any unary language is a subset of Σ^*, and can be re-interpreted as an infinite binary sequence, where string s_i is in the language if and only if sequence bit b_i is 1. Clearly, there is bijective map from unary languages to infinite binary sequences. But the set of infinite binary sequences is uncountably infinite.

Ex. 2

a

Assume the input is conveniently given as *w#, and machine's tape-1 is on first character of w (or on # if w is ϵ). Assume also tape-2 is given with* character at its beginning, and tape-2 head is one-step right to it. For brevity we ignore such implementation details.

Initial state is $q_{\text{rightDouble1}}$. We assume characters of the alphabet have corresponding other marked characters. e.g 1 has 1'. We indicate by saying marking a character, replacing it with its corresponding marked one.

$q_{\text{leftUntill*}}$: move tape-1 left untill * is read, Then move right and $q_{\text{rightDouble1}}$.

$q_{\text{rightDouble1}}$: sequentially, move tape-1 right once, write 1 in tape-2 twice in two sequential slots, untill # is read by tape-1 then move left and $q_{\text{isAllMarked}}$.

$q_{\text{isAllMarked}}$: if a marked character or * is read by tape-1, then $q_{\text{unmarkInput}}$. if a non-marked character is read, then $q_{\text{leftUntill*}}$.

$q_{\text{unmarkInput}}$: move tape-1 left replacing marked characters by their unmarked counterparts untill * is read, then move right and $q_{\text{lefttape2}}$.

$q_{\text{lefttape2}}$: move tape-2 left untill * is read, then move right.

It is easy to transform tape-1 to be exactly w and tape-2 to be popped out of * at its left-most. We ignore these implementation details.

b

From a, We know there's a 2-tape turing machine T that can prepare tape-2 with string $1^{2|w|^2}$. We wish to think of this string as a counter of number of steps taken by the machine. It is possible for it to be augmented, to simulate machine M, while ticking tape-2, Accepting if M accepts and tape-2 isn't completely ticked, and rejecting otherwise. The 1-tape turing machine M can be constructed by simulating T.

T runs in polynomial time, since preparing tape-2 is upperbounded by n^2, and ticking tape-2 while simulating M doesn't cost any more steps. The simulation of T by M' is polynomially overheaded by T, and hence M' is polynomially upperbounded.

Ex. 3

a

Let T be the turing machine recognizing L. It is possible to construct another turing machine T' whose *accept* and *reject* states are swapped. So, $w \in L$ iff T accepts w iff T' rejects w iff $w \notin L^c$.

The number of steps made by T' is exactly the same as T, and hence of a polynomial complexity.

b

Let T_1 and T_2 be two turing machines recognizing L_1 and L_2 respectively. It is possible to construct a new turing machine T that simulates T_1 and memorizes its result, Then instead of termination, simulates T_2 on the same input and memorizes its result also. It is easy for T to be designed such that it accepts input w if and only if either the simulation of T_1 or T_1 accepted.

The complexity of T is $\mathcal{O}(poly(n)) + \mathcal{O}(poly(n)) + C = \mathcal{O}(poly(n))$

c

Similarly to b, but T accepts input w if and only if both the simulations of T_1 and T_2 are accepted.

d

Similarly to b, but T accepts input w if and only if at least two out of the three simulations of T_1, T_2, and T_3 accepts.

e

Let T be the turing machine recognizing L. For any $m \in \mathcal{N}$, It is possible to construct a m-tape turing machine T_m, that simulates T on w_i on the ith tape. Hence, complexity of T_m is $\sum_{i=0}^{m} \mathcal{O}(poly(|w_i|)) = \mathcal{O}(poly(|w|))$. Clearly the processing required to copy substrings w_i on ith tapes is polynomial also, Hence T_m's established upperbound remains the same.

Ex. 4

Any turing machine needs at least a linear scan of cost n, to behave in number of steps, as a function of n. In other words, number of steps T, must be $T(n) \geq n$, so that $T = f(n)$ for some function f. Since T is upperbounded by $\sqrt{n} = \wr(n)$, $T(n) \neq f(n)$ for any f. In other words, T isn't based on input size n. Therefore, must be a constant.

The intuition is very strong that a turing machine cannot behave in relation to n if its memory doesn't contain n's value; I am not aware of a more rigorous proof.

As final note, For any constant C, we can always find some n, such that $C\sqrt{n} < n$. Hence, always guaranteeing the turing machine can't read its whole input.