Problem 1.

Are the following languages Turing decidable, Turing acceptable but not Turing-decidable, or not even Turing acceptable?

- $L = \{p(M)p(w) : M$ uses a finite number of tape cells when running on input $w\}$.
- $L = \{p(M)p(w)01^n0 : M$ uses at most n tape cells when running on input$w\}$.

Here, "using n cells" means that the head of the (deterministic) TM M reaches the n-th cell from the left during its computation. Justify your answer clearly: both exercises require careful thinking.

Solution

Problem 2.

Are the following languages Turing-decidable? Turing-acceptable but not Turing-decidable? Not even Turing-acceptable? For each answer, just give an intuitive explanation of your reasoning, no formal proof is required (just as in class, M is a generic deterministic Turing machine, w is a generic input string to it, and p is an encoding function).

Solution

In all the problems below, I discuss using a counter to keep track of iterations run by M, this counter is stored at some arbitrary point on the tape. Not that important, but maybe worth mentioning.

Part (a)

$$L = \{p(M) : |L(M)| \leq 10\}$$

acceptable Assume we are given a machine for a language such that $|L(M)| \leq 10$. We want to verify this condition given only the machine M. To do so would requiring enumerating all possible strings and checking whether or not M accepts them, if so we would increment a counter. After enumerating all strings we would check that counter ≤ 10 and if so we would accept. Note that this requires checking **all** possible strings and this is the problem. Strings can be arbitrarily large to infinite (though not infinite) and thus we cannot possible check all strings. Therefore L is not Turing-acceptable and cannot possibly then be Turing-decidable (note it cannot be decidable without first being acceptable).

Part (b)

$$L = \{p(M) : |L(M)| \geq 10\}$$

acceptable Assume we are given a machine for a language such that $|L(M)| \geq 10$. We have a finite alphabet we are working with that is encoded into $p(M)$. We can enumerate every possible string for sizes counting from $0, 1, 2, ...$ and so on keeping a counter on when we find a string accepted by M. Once this counter reaches 11, we halt and accept. Because we know M has at least 11 such strings, and these strings have finite length, we must therefore be guaranteed to halt in finite time. Therefore L is Turing-acceptable.

decidable We employ a similar approach here, this time given a machine such that $|L(M)| < 10$. Our machine must verify this condition given only M. However doing this would require permuting all possible input strings and making sure less than 10 are produced by M, if so our machine would halt and reject. However input strings can be arbitrarily large to infinite, thus our machine will loop forever checking every possible string. Therefore L is **not** Turing-decidable.

Part (c)

$$L = \{p(M)p(w) : M \searrow w \text{ in 10 steps or less } \}$$

acceptable Assume we are given a machine M and string w and assume that the condition "$M \searrow w$ in 10 steps or less" is true. This is easy to verify, simply run M on w keeping a counter on each step, once our counter reaches 11 we halt and reject. If M halts before that our machine will halt and accept. Because it is given that M will halt in 10 steps or less on w, we know that our machine L will halt as well (and then accept). Therefore L is Turing-acceptable.

decidable Assume we are given a machine M and string w for which M will not halt in 10 steps or less, this could mean that it halts in finite time or runs indefinitely. L should reject this input. Again, following the outline given above, we will halt and reject as soon as our counter reaches 11, thus even if M does not halt on w we don't care as L won't need to run (simulate) past step 11 (of M). Therefore L is Turing-decidable.

Part (d)

$$L = \{p(M)p(w) : M \searrow w \text{ in 10 steps or more } \}$$

acceptable Assume we are given a machine M and a string w for which M will halt in 10 steps or more. This will be some finite positive integer, call it $n \geq 10$. Our machine L will run M on w and keep a counter on the number of iterations if M halts and the counter is less than 10 we halt and reject otherwise we halt and accept. We know that M will halt and specifically will halt with our counter set to n. Further we know that $n \geq 10 \Rightarrow \text{counter} \geq 10$, thus our machine is guaranteed to halt and accept the input of M and w. Therefore L is Turing-acceptable.

decidable If our machine M halts on w and does so in less than 10 steps we will obviously not have troubles. However the case of M not halting on w is trouble. In this case in fact we just have the halting problem. We cannot decide whether or not M will halt in finite time using L. Therefore L is **not** Turing-decidable.

Problem 3.

Use reduction to prove that the language

$$L = \{p(M_1)p(M_2) : L(M_1) \subseteq L(M_2)\}$$

is not decidable (M_1 and M_2 are Turing machines, of course).

Solution