Problem 1.

Define an unrestricted grammar for the language $L = \{ww : w \in \{0,1\}^*\}$.

Solution

Let $G = (\{S, R, T, \#, Q\}, \{0,1\}, P, S)$ with productions P given by

$$\begin{aligned} S &\to R\# \\ R &\to 0R0 \mid 1R1 \mid T \\ T &\to TQ \\ Q00 &\to 0Q0 \\ Q01 &\to 0Q1 \\ Q10 &\to 1Q0 \\ Q11 &\to 1Q1 \\ Q0\# &\to \#0 \\ Q1\# &\to \#1 \\ T\# &\to \epsilon \end{aligned}$$

The idea behind this grammar is relatively straightforward. R generates a string of the format $ww^T : w \in \{0,1\}^*$. S ensures that there is a non-terminal # at the end, this is our pivot point which we will be flipping w^T over. On completion, R will take the form wTw^T. Thus the entire string looks like $wTw^T\#$. The production $T \to TQ$ produces a non-terminal Q, who's productions carry the first bit to its right and inserts it over the pivot. By definition of these productions, $Q's$ cannot pass each other while moving bits. Thus, as T can generate a TQ for each bit, bits will be dumped from left to right over the pivot in reverse order, producing $w^{TT} = w$. After these operations complete we have a string of the form $wT\#w$. We apply the final production $T\# \to \epsilon$ to arrive at a string of terminals exactly of the form $\{ww : w \in \{0,1\}^*\}$. Note if we continued to create $Q's$ via T we would have $wTQ^n\#w, n \in \mathbb{N}, n \geq 1$, no production can be applied to $Q^n\#$ and thus the string will never be able to remove its non-terminals.

Problem 2.

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

Solution

Part (a)

$$L = \{p(M)p(w) : M \text{ uses a finite number of tape cells when running on input } w\}.$$

acceptable Assume we are given some machine M and a string w for which our condition holds, that is M uses a finite number of tape cells when running on input w. We want L to halt and accept this input configuration. Note that it is not necessary for M to halt, it is possible that M loops over a finite subset of the tape, thus never halting, but never using infinite tape cells. If this is the case (finite memory), note that Q, Σ are both finite, the tape head can only be on a finite number of positions, and the tape stores a finite amount of data. Given this, we can produce some sort of finite 2-tuple (machine state and the finite state of the tape) which encodes every possible state this machine can be in (I reiterate we are using the strong assumption that M uses finite resources). Since we know that there are finite such possible 2-tuples (there is a finite number of unique configurations our machine can be in), and that the machine never halts, we can conclude that the machine will enter the exact same configuration at least twice (infinitely in particular) for at least one configuration of the machine, this follows from the Pigeon-Hole Principle (infinitely iterations cannot fit in a finite number of states). Further if a machine enters a given state, call it S and later enters that same exact state S, we can guarantee that the machine will infinitely enter state S as it runs. This is because our current configuration is exactly the same and one run following S led to another S, all subsequent occurrences of S will lead to another.

From all this we conclude if the machine does not halt but uses finite resources, it MUST repeat some configuration of the machine (including state and tape layout). Since S is repeated infinitely often, we know that it is impossible that we are writing to more cells in the tape then are used in S, else the next time we reach S we would have a different tape configuration contradicting $S = S$ (that is repeated configurations $\Rightarrow$ finite memory).

Given input M and w our machine will halt and accept if M halts. If M halts it must have done so in finite steps $\Rightarrow$ it used finite resources $\Rightarrow$ we should accept it. For each step of M we will encode it's current configuration (as an 2-tuple) at some empty point on our tape.

Then we will check all configurations this machine has been in (which will always be finite, but arbitrarily large) and if any configuration is repeated we will also halt and accept. This is because as argued above M has entered an infinite loop via repeated configurations $\Rightarrow$ M is using at most the number of cells used in that repeated configuration $\Rightarrow$ M is using a finite number of cells. In both cases (M halts and M does not halt) L will halt and accept.

If M uses an infinite number of tape cells, L will not halt (thus will not accept), this is argued below for decidability but mentioned to verify that L does not accept these configurations. Therefore L is Turing-acceptable.

decidable Assume we are given some machine M and a string w for which M uses an infinite number of cells. As M can only modify one tape cell in one step, use of infinite tape cells requires infinite steps $\Rightarrow$ M does not halt. Further, we can make now assumptions about the behavior of M with regards to not halting (above we could make assumptions about space usage). Thus we would need to be able to solve the halting problem in order to conclude to halt and reject M. But as the halting problem is undecidable, L must also be **not** decidable.

Part (b)

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

acceptable Here we have a similar setup to part (a) with the following modifications: after each iteration of M, check the simulated tape for M, if the number of used tape cells exceeds n (encoded as a string of 1's), then halt and reject. If M halts, then it never exceeded n tape cells (else we would have already halted and rejected), thus we can halt and accept. If M repeats a configuration (and enters a loop as noted above), then we know that the number of currently used tape cells is the maximum it will ever use (argued in part (a)), therefore we can halt and accept (again if it was using more than n we would have already rejected it).

Assume we are given M, w and n encode into L and that M computes w using at most n tape cells. Then, if M halts our machine must also halt, and since it never used more than n tape cells, we never rejected it, therefore we have accepted it. If M does not halt, n is finite $\Rightarrow$ M uses a finite amount of memory $\Rightarrow$ M will enter a repeat configuration. That configuration will not use more than n tape cells, therefore we will have never rejected the input and will instead halt and accept. In both cases, L halts and accepts, therefore L is Turing-acceptable.

decidable

case M halts:

Since L simply simulates M with a couple of extra steps in between each iteration, if M halts then so must L this is fairly trivial. Further if M has used more than n tape cells, we must have rejected it whenever it past that barrier. L halts and rejects in this case.

case M does not halt, uses finite tape cells:

M must have entered a repeat configuration and we must have halted at latest at that point. If M uses more than n tape cells then we rejected as soon as it did so. Thus L both halts and rejects in this case.

case M does not halt, uses infinite tape cells:

M uses infinite resources therefore it must use exactly $n+1$ resources at some point. Further it reaches $n+1$ resources in finite time, if it reached it in infinite time that would contradict the idea that M uses infinite tape cells. L is designed to halt and reject as soon as $n+1$ tape cells are used, and as these are used in finite time. L halts and rejects in finite time in this case.

In all cases, L halts and rejects. Therefore L is Turing-decidable.