ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

HS 2020

Prof. R. Wattenhofer
Pál András Papp and Robin Fritsch

Computational Thinking
Sample Solution to Exercise 13

1 Undecidable problems

For each of the subtasks, we prove undecidability through a reduction from the halting problem.

a) Let us build a TM T' as follows: given a TM T and an input x, T' simulates the running of T on x. If T ever reaches its halting state, then we insert a further state into the simulation at the end where T' puts a special symbol \$ on the tape (i.e. a symbol which is never used by T). Hence when running T' on x, we put a symbol \$ on the tape if and only if T halts on x. Hence deciding if such a symbol \$ occurs would also allow us to solve the halting problem.

b) Given the halting problem on TM T and input x, we create two new TMs T_1 and T_2. The machine T_1 checks if its input y equals x; if it does, then it simulates machine T on x, but otherwise, it simply goes into an infinite loop. The machine T_2, however, loops infinitely for every input. Hence a desired input y (where T_1 halts but T_2 does not) exists if and only if T halts on x. Thus once again, solving this problem would allow us to solve the halting problem.

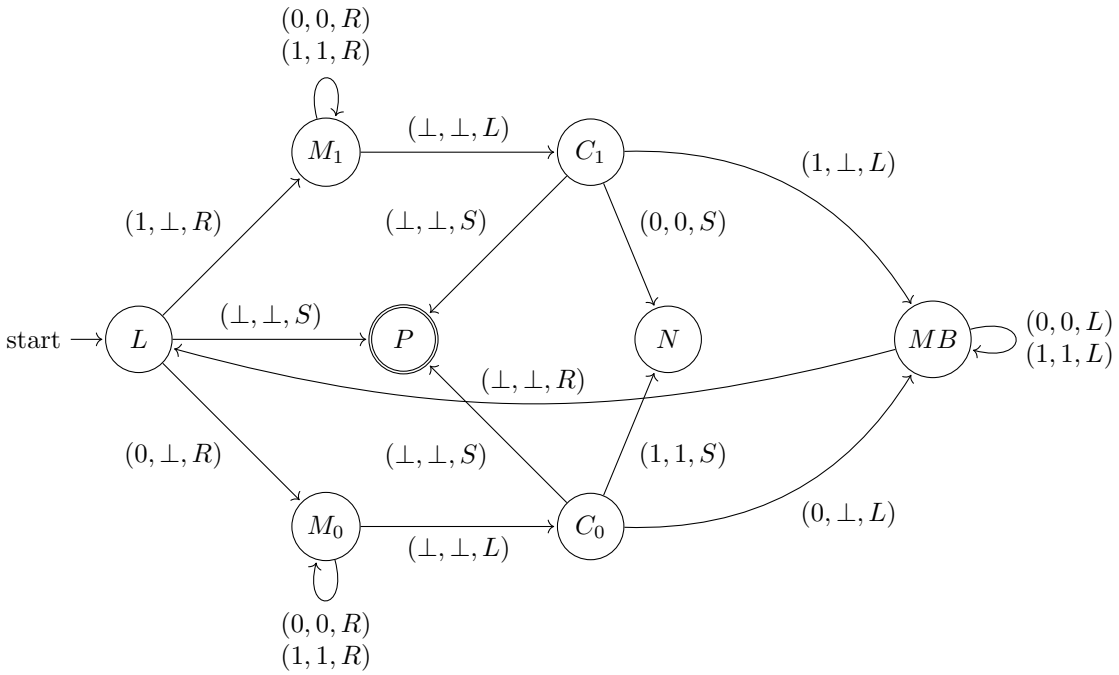
2 Turing Machines

a) A possible design of the TM is shown in the diagram below. Regarding the notation: A tuple $(1, \perp, R)$ on an arrow from state A to state B means that, if the TM is in state A and reads a 1 on the tape, it will write $\perp$ on the tape, move its pointer to the right, and go into state B. The accepting state is circled twice.

The general idea is to repeatedly check if the outer two characters of the string match and remove them. If this leaves an empty string or a single character, the string is accepted as a palindrome.

More precisely, the TM begins in state L ("left") on the leftmost character of the string. If it finds an empty string, it goes straight into the accepting state P ("palindrome"). If it reads a 1 (for a 0 it proceed similarly), it removes the 1 and moves along the string in state M_1 ("move after 1"). When it arrives at the other end, it goes into state C_1 ("check for 1"). If it finds a 0 now, it goes into state N ("no palindrome"). If it finds an empty cell, only a single character was left, so it goes into state P ("palindrome"). If it finds a 1, it removes the 1, goes into state MB ("move back") and moves back to the left of the string.

See the notebook for the code corresponding to this diagram which you can run on `https://turingmachine.io`.



b) See the notebook for a solution that repeatedly increases the left number by one and decreases the right number by one. For a solution that adds by carrying the bits, see the sample solution for binary addition on `https://turingmachine.io`.

3 One-dimensional tiling

a) Since S is a finite set, there are only finitely many colors that appear on the right side of a given tile. This means that if we have a correct tiling of our infinite horizontal line, then there must be two positions x_1 and x_2 (with $x_1 < x_2$) where the tiles have the same color c on their right side. In this case, tile $x_1 + 1$ must also have the color c on its left side for a correct tiling. However, then the segment $[x_1 + 1, x_2]$ has color c on its outer face at both of its endpoints, and hence a periodic repetition of this finite segment allows us to tile the entire line correctly.

b) This observation implies that in order to find a solution, we only need to find a finite segment that (i) corresponds to a correct tiling, and (ii) starts and ends with the same color (let us call this a *periodic segment*). If such a segment does not exist, then our set of tiles does not allow a solution.

Let $n = |S|$; this also means that there are at most n different colors appearing on the left side of a tile. This implies that if there exists a periodic segment at all, then the shortest periodic segment can have length at most n: if the shortest periodic segment was larger, then it would already contain two tiles with the same color on the left side, which would allow us to find an even shorter subsegment that is also periodic.

Given n different types of tiles, we can only have

$$n + n^2 + ... + n^n \leq n \cdot n^n = n^{(n+1)}$$

possibilities for such a finite sequence. We can enumerate and check all of these in finite time to see if they form a periodic segment. If we fins such a segment, then a periodic repetition of this segment indeed gives a valid tiling of the line. On the other hand, if we find no such segment, then a valid tiling does not exist.