CSCI 2400 – Models of Computation, Section 3

Solutions to Homework 9

Problem 1.

For convenience, we will call the problem of determining whether a Turing machine M with input w ever writes symbol a on the tape, as the "symbol-writing problem".

We want to prove that the symbol-writing problem is undecidable. We will prove this by reducing the halting problem to the symbol-writing problem. Namely, we will prove that if the symbol-writing problem is decidable then the halting problem is also decidable. (Therefore, it must be that the symbol-writing problem is undecidable, since we know that the halting problem is undecidable.)

Let's assume that the symbol-writing problem is decidable. We have that there is an algorithm that decides the symbol-writing problem. Using this algorithm, we can design the algorithm that decides the halting problem as follows (see also Figure 1).

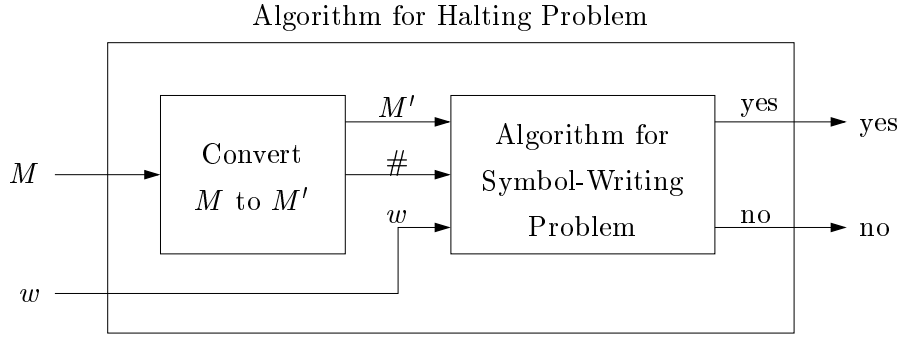

Figure 1: The algorithm that decides the halting problem

`Algorithm that decides the halting problem:`

Input: the input of the halting problem is a machine M and a string w, and we want to find if machine M halts on input w.

- We convert machine M to machine M' such that machine M' writes a special symbol on the tape, say symbol #, if and only if machine M halts. We choose the special symbol # to be different from any other symbol written on the tape.

  We can easily construct machine M' as follows. Machine M' is identical with machine M. The only difference is that in M' we add transitions from every halting state of M to a new state, and in these transitions we

write the symbol # on the tape. (We can easily identify the halting states of M from the fact that these states have undefined transitions for some symbols of the tape. For all these undefined transitions we add the new transitions where we write symbol #.)

- We run the decision algorithm of the symbol-writing problem with input parameters the machine M', the input string w, and the symbol #.

  The answer to the above symbol-writing problem will be the answer to the halting problem. This is because machine M' with input w writes symbol # on the tape if and only if machine M halts on input w.

Problem 2.

For convenience, we will call the problem of determining whether machine M halts on all input, as the "all-input halting problem".

We want to prove that the all-input halting problem is undecidable. We will prove this by reducing the halting problem to the all-input halting problem. Namely, we will prove that if the all-input halting problem is decidable then the halting problem is also decidable. (Therefore, it must be that the all-input halting problem is undecidable, since we know that the halting problem is undecidable.)

Let's assume that the all-input halting problem is decidable. We have that there is an algorithm that decides the symbol-writing problem. Using this algorithm, we can design the algorithm that decides the halting problem as follows (see also Figure 2).

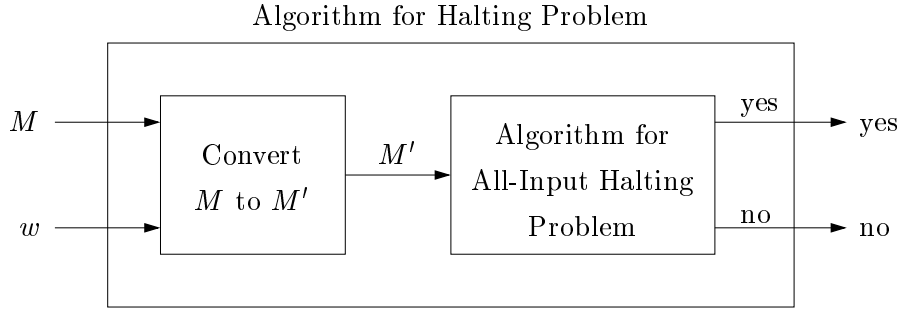

Figure 2: The algorithm that decides the halting problem

`Algorithm that decides the halting problem:`

Input: the input of the halting problem is a machine M and a string w, and we want to find if machine M halts on input w.

- We convert machine M to machine M' such that machine M' halts on all input if and only if machine M halts on input w.

We can easily construct machine M' as follows. Machine M' is identical with machine M with the difference that in M' we add transitions from every halting state of M to a new final state. (We can easily identify the halting states of M from the fact that these states have undefined transitions for some symbols of the tape. For all these undefined transitions we add the new transitions to the new final state.)

Furthermore, for any input string, machine M' first copies w on the tape and then continues the computation with input string w (ignoring the original input string). If machine M halts on w then machine M' enters a final state and accepts the original input string. Thus, if machine M halts on w then machine M' always accepts the input string.

- We run the decision algorithm of the all-input halting problem with input parameter the machine M'.

  The answer to the above all-input halting problem will be the answer to the halting problem.

Problem 3.

Notice that every head move corresponds to a computation move. Therefore, function $b(n)$ is the same with function $f(n)$ described in Example 12.3, page 319. The solution is the same with that Example.

Problem 4.

We will call the problem of determining for two recursively enumerable languages L_1 and L_2 whether $L_1 \subseteq L_2$ as the "subset problem". We want to prove that the subset problem is undecidable.

From Theorem 12.3, page 322, we know that the problem of determining whether $L_1 = \emptyset$ is undecidable. Let's call this problem as the "empty-set problem".

We will reduce the empty-set problem to the subset problem. Take $L_2 = \emptyset$. It is easy to see that L_2 is recursively enumerable, since we can construct a Turing machine with a single state (non-final) and with no transitions that accepts no string.

Obviously, if we can determine the answer to the subset problem $L_1 \subseteq \emptyset$, then we can determine the answer to the empty-set problem $L_1 = \emptyset$.

Problem 5.

(a) Yes there is a PC-solution. The sequence of strings is: $3, 4, 1$. The string we obtain is: 11101001.

(b) There is no MPC-solution, since the first strings 001 and 01 cannot match.