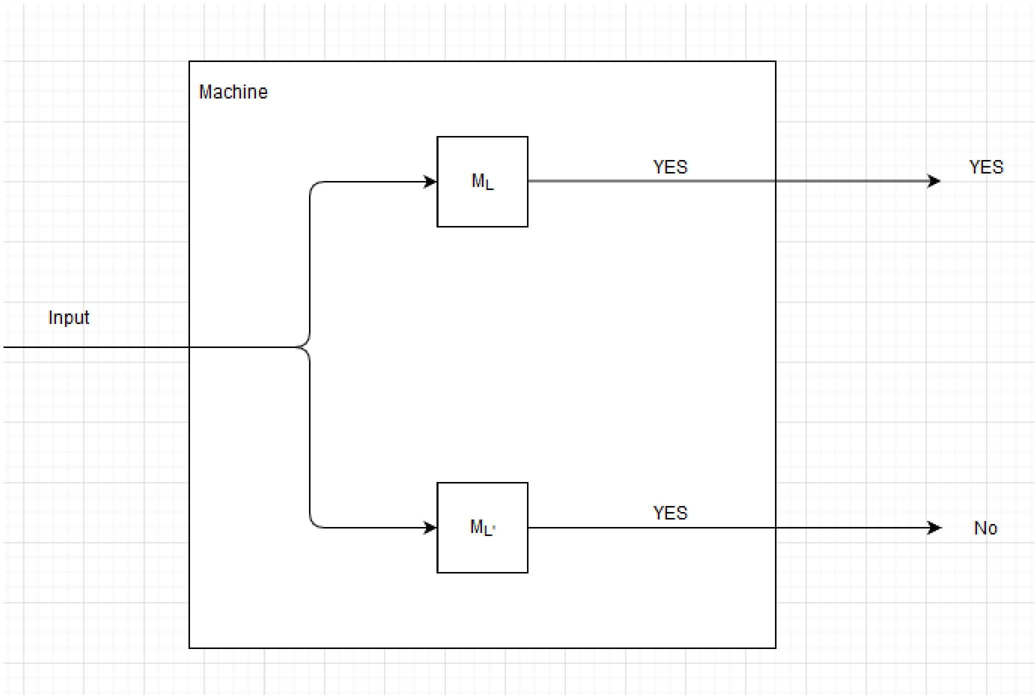
1.

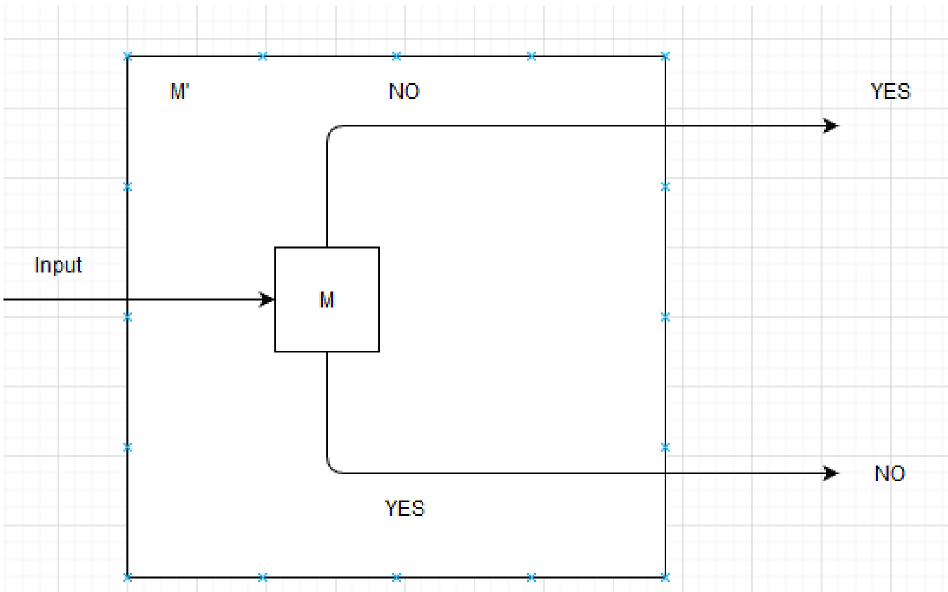
(1) Due to undecidability in the 'no' case, r.e. machines could continuously run forever in a loop meaning that they are not Turing decidable. Recursive machines are Turing decidable as they always respond with a 'yes' or 'no'. This means that *L* and *L'* can be put together to form one machine *M*:



This new machine *M* will always produce an output of either yes or no, making it recursive. Thus making *L* recursive.

(2) If *L* and *L'* are both semi-decidable, then it must follow that they are both decidable under their compliment. Therefore, if an r.e. is closed against its complement and it is recursive, then it is avd r.e. However, because of the halting problem example it is known that recursive $\not\subset$ r.e., so that cannot be true. Thus, r.e. Languages are not closed under their complement.

2. A language *L* is recursive if there is a TM *M* that when given any input x, *M* will halt on it. It will either halt and accept, or halt and reject, but it will never go on in an infinite loop. Suppose we have the complement of *L, L'*. The TM for *L', M'* can be shown as:



As we can see in *M'*, all inputs that *M* rejected are now accepted by *M'* and all inputs that *M* accepted are rejected by *M'*. This proves that *M'* is Turing decidable and recursive languages are closed under their complement.

3. The string determined by M_i is the $w_i m$ word in the dictionary ordering. If a string w_i must be admitted by TM M_i, the state to recognize **W+TF** should be there in M_i. To recognize the strings in a given dictionary, the TM has to be lower compatible, so the language determined by M_i can also be accepted by *M*. If the language accepted by two TMs are the same, we can say that $L(M) = L(M_i)$ since the TMs accept the same language the TMs are the same.

4. We will construct a TM *M* that recognizes *L* and *R* instructions. We will list all transition functions of *M*. We will count the number of transition functions moving towards *L* and the number of transition functions moving towards *R*. If the number of transition functions moving towards *L* and *R* are the same, then the TM will be in the language. Therefore, a polynomial time algorithm that checks such a number can exist, so the problem is decidable.

5. We will construct a TM *M* that recognizes *L* and *R* instructions.
M = "on input <*M,M'*>"
(1) Repeat the following for i = 1, 2, 3, …, n
(2) Simulate *M* for i steps, record any output *L*-*R*-instructions on the tape.
(3) Simulate *M'* for i steps, storing any output of *M'* on the tape.
(4) Compare the output of *M* and *M'*. If the two output the same value, then it is decidable and $L(M) = L(M')$