Problem 1.

Prove that the class of Turing-acceptable languages is closed under union, intersection, and reversal. For each property, give a detailed sketch of the proof, by saying how you would build a Turing machine that accepts the resulting language, given the Turing machine(s) that accept the original language(s).

Solution

Union

Proof. Let L_1, L_2 be Turing-acceptable languages, then define their corresponding Turing machines as M_1, M_2 respectively.
We will then define a new machine M_U which operates on the union of Q, Σ for M_1, M_2. We will distinguish each state by the source machine, thus $Q_1 \cap Q_2 = \emptyset$. The initial state of M_U will be the initial state of M_1.
On the input of a string w, M_U will behave exactly as M_1. If we reach a halting and accepting state of M_1, then M_U will also halt and accept. Otherwise, rather than halting and accepting, M_U will then run M_2 on w, once M_2 halts, M_U accepts if M_2 does, and rejects otherwise. Note that M_I will halt provided that both M_1 and M_2 each halt.
Now assume $w \in L_1 \cup L_2$, then either $w \in L_1$ or $w \in L_2$, if $w \in L_1$ w will be accepted by M_U after the initial run of M_1 (as M_1 accepts w). Otherwise if $w \in L_2$, then after M_U runs M_1 where M_1 rejects w, M_U will continue with M_2 which accepts w and thus so does M_U.
If $w \notin L_1 \cup L_2$ then $w \notin L_1$ thus M_U continues onto M_2, but as $w \notin L_2$, L_2 will reject w and thus so will M_U.

□

Intersection

Proof. Let L_1, L_2 be Turing-acceptable languages, then define their corresponding Turing machines as M_1, M_2 respectively.
We will then define a new machine M_I which operates on the union of Q, Σ for M_1, M_2. We will distinguish each state by the source machine, thus $Q_1 \cap Q_2 = \emptyset$. The initial state of M_U will be the initial state of M_1.
On the input of a string w, M_I will behave exactly as M_1. If we reach a halting and rejecting state of M_1, then M_I will halt and reject, if M_1 halts and accepts, M_I will continue on and run M_2 on w. If M_2 accepts w, then so will M_I otherwise w is rejected by M_I. Note that M_I will halt provided that both M_1 and M_2 each halt.

Now assume $w \in L_1 \cap L_2$, then $w \in L_1$ and $w \in L_2$, because of this, w will run through both stages of M_I and pass each, thus M_I clearly accepts w. However, if $w \notin L_1 \cap L_2$, then $w \notin L_1$ (in which case w is rejected in the first stage of M_I) or $w \notin L_2$ (in which case w is rejected by the second stage of M_I). In either case, w will be rejected by M_I, therefore M_I represents $L_1 \cap L_2$. $\square$

Reversal

Proof. Let L be a Turing-acceptable language, then define a corresponding Turing machine M which. Define a new Turing machine M_R using the same alphabet, and set of states as M. However let the initial state of M_R be the final accepting state of M, similarly, let the final state of M_R be the initial state of M.

We can then run M backwards on w (note we are starting on the final state), if we can then reach the initial state of M we have successfully read the reverse of a string $w \in L$, thus the initial state of M will be the final accepting state of M_R. If running M backwards halts but does not accept, then M_R should reject the input. In this sense, the δ function of M_R is the inverse of the δ function for M.

To show this accepts reversal, assume $w \in L$ then M accepts w and $w^R \in L^R$. Running M_R on w^R is equivalent to running w on M, which accepts w thus M_R accepts w^R. If $w \notin L$ then M does not accept w, thus $w^R \notin L^R$. Running M^R on w^R is again equivalent to running w on M, however as $w \notin L$, we know that M will not reach an accepting state, therefore running δ backwards from the accepting state cannot possibly reach the initial state (the final accepting state of M^R) therefore M^R will not accept w^R. $\square$

Problem 2.

Prove or disprove that the set of Turing-acceptable languages is closed under concatenation.

Solution

Proof. □

Problem 3.

Consider a new type of *deterministic* machine, having one read-only input tape and two stacks. The tap is read-only, it cannot be written, but the head can move left, right, or do nothing. Each stack operates, independently of the other, as in a deterministic pushdown automaton:

$$M = (K, \Sigma, \Gamma_1, \Gamma_2, z_1, z_2, \delta, s)$$

where K is a finite set of states, Σ is a finite input alphabet, Γ_1 and Γ_2 are two finite stack alphabets, $z_1 \in \Gamma_1$ and $z_2 \in \Gamma_2$ are the initial symbols for the two stacks, $s \in K$ is the initial state. h is a special halting state not in K, just like a Turing machine.

(a) Give an appropriate definition for the transition function δ, for a configuration of this machine, for the "yields in one step" operator, and for the language accepted by this machine.

(b) These machines can accept the same languages as a class of automata you already know: deterministic pushdown automata, pushdown automata, or Turing machines? Prove your answer formally.

Solution