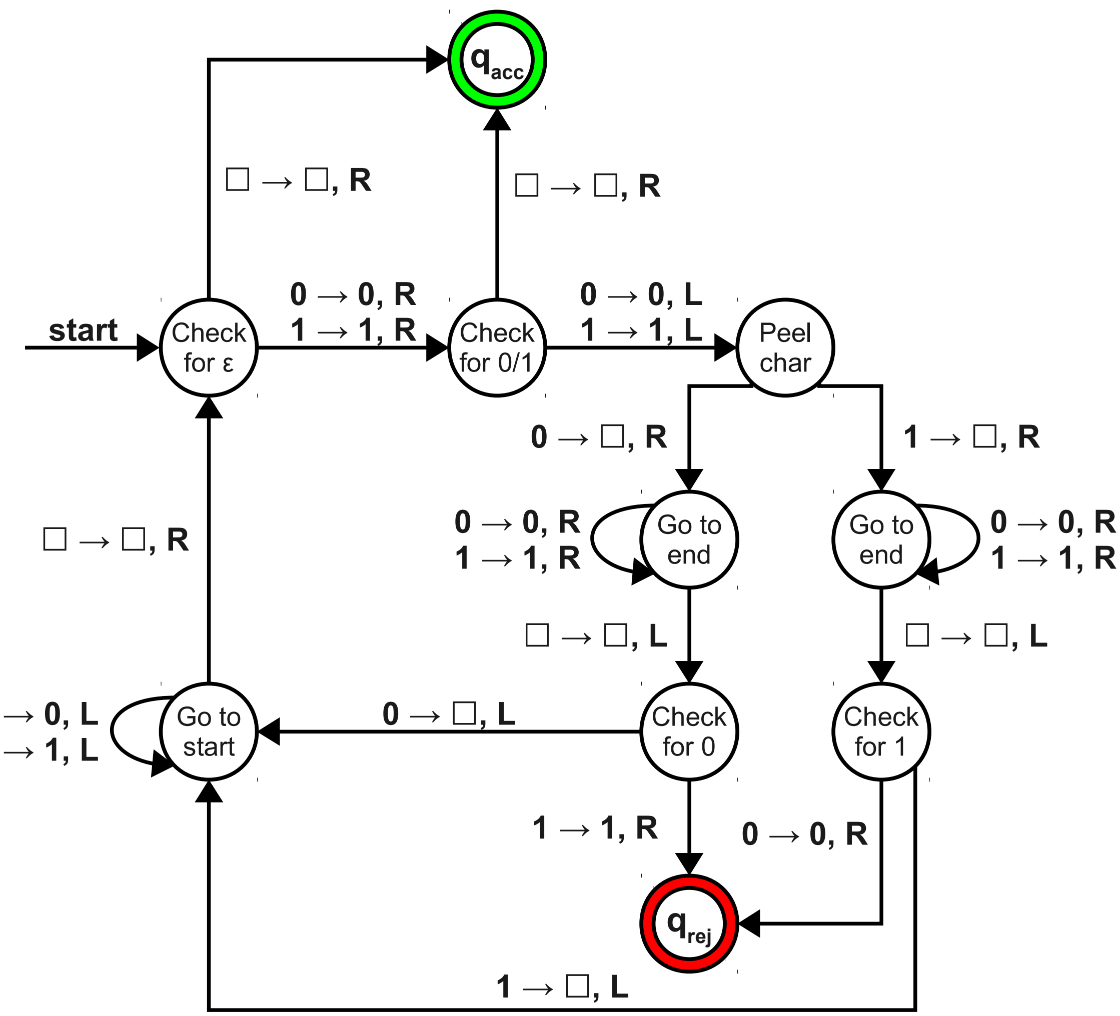
Discussion Solutions 6

Problem One: Designing Turing Machines

Here is one possible option:



This TM is based on the following recursive observation:

- The strings ε, `0`, and `1` are palindromes.
- Any longer string is a palindrome iff it starts and ends with the same character and the string formed by removing those characters is a palindrome.

Notice how the TM uses constant storage to remember what the first character of the string is.

Problem Two: Nondeterministic Algorithms

Prove that if $L \in \mathbf{RE}$ and f is a computable function, then $f(L) \in \mathbf{RE}$. As the title of this problem suggests, you might want to build a nondeterministic Turing machine for $f(L)$.

Theorem: If $L \in \mathbf{RE}$ and $f : \Sigma^* \to \Sigma^*$ is a computable function, then $f(L) \in \mathbf{RE}$.

Proof: Consider any $L \in \mathbf{RE}$ and any computable function f. This means that there is a recognizer R for L and a TM F that computes f.

Consider the following NTM N:

N = "On input $w \in \Sigma^*$:
Nondeterministically guess a string $x \in \Sigma^*$.
Using F as a subroutine, compute $f(x)$.
If $f(x) \neq w$, reject.
Deterministically run R on x.
If R accepts x, accept.
If R rejects x, reject."

We prove that $\mathscr{L}(N) = f(L)$, from which we have that $f(L) \in \mathbf{RE}$. To see this, we will prove that $w \in f(L)$ iff there is some series of choices such that N accepts w. To see this, note that there is some choice of $x \in \Sigma^*$ such that N accepts w iff $f(x) = w$ and R accepts x. Since R accepts x iff $x \in \mathscr{L}(R)$ and $\mathscr{L}(R) = L$, this means that N accepts w iff there is some choice of x such that $f(x) = w$ and $x \in L$. Finally, note that there is some choice of x such that $f(x) = w$ and $x \in L$ iff $w \in f(L)$. Thus N accepts w iff $w \in f(L)$, so $\mathscr{L}(N) = f(L)$, as required. ■

Problem Three: Unsolvable Problems

Consider the language $L = \{ \langle M, w, q \rangle \mid$ TM M does not enter state q when run on string $w \}$. Prove that $L \notin \mathbf{RE}$ by showing if $L \in \mathbf{RE}$, then $L_D \in \mathbf{RE}$.

Theorem: $L \notin \mathbf{RE}$.

Proof: By contradiction; assume $L \in \mathbf{RE}$. Let R be a recognizer for L and consider the following TM H:

H = "On input $\langle M \rangle$:
Run R on $\langle M, \langle M \rangle, q_{acc} \rangle$, where q_{acc} is M's accepting state.
If R accepts $\langle M, \langle M \rangle, q_{acc} \rangle$, accept.
If R rejects $\langle M, \langle M \rangle, q_{acc} \rangle$, reject."

We claim that $\mathscr{L}(\mathrm{H}) = L_D$. To see this, note that H accepts $\langle M, w \rangle$ iff R accepts $\langle M, \langle M \rangle, q_{acc} \rangle$. R accepts $\langle M, \langle M \rangle, q_{acc} \rangle$ iff M does not enter state q_{acc} when run on $\langle M \rangle$, and M does not enter state q_{acc} when run on $\langle M \rangle$ iff M does not accept $\langle M \rangle$. Finally, M does not accept $\langle M \rangle$ iff $\langle M \rangle \in L_D$. Thus H accepts $\langle M \rangle$ iff $\langle M \rangle \in L_D$, so $\mathscr{L}(H) = L_D$.

Since H is a recognizer for L_D, we have $L_D \in \mathbf{RE}$. But this is impossible, since $L_D \notin \mathbf{RE}$. We have reached a contradiction, so our assumption was wrong. Thus $L \notin \mathbf{RE}$. ■