CPSC 351 — Tutorial Exercise #10
Additional Practice Problems

About These Problems

These problems will not be discussed during the tutorial, and solutions for these problems will not be made available. They can be used as "practice" problems that can help you practice skills considered in the lecture presentation for Lecture #10, or in Tutorial Exercise #10.

Practice Problems

Once again, let $\Sigma = \{\mathtt{a}, \mathtt{b}\}$. A string

$$\omega = \alpha_1 \alpha_2 \ldots \alpha_n$$

(with length $n \geq 0$, so that $\alpha_1, \alpha_2, \ldots, \alpha_n \in \Sigma$) is a ***palindrome*** if $\omega = \alpha_n \alpha_{n-1} \ldots \alpha_1$ as well — that is, you get the same string by reading the symbols from right to left instead of from left to right.[1]

Let

$$L = \{\omega \in \Sigma^\star \mid \omega \text{ is a palindrome}\}.$$

These practice problems are intended to help you to practice the skills (concerning Turing machines) that you have developed, so far, by developing a proof that a Turing machine, shown later, decides this language. The last problem is somewhat more challenging than the corresponding problems on Tutorial Exercise #10, because it asks you to discover the things that the Turing machine that you must establish, in order to prove its correctness — while these things were listed in the exercise's problems, for the Turing machine that was considered there.

1. Consider the "`isPal`" algorithm that is shown in Figure 1 on page 2. Prove that an execution of this algorithm on an input string $\omega \in \Sigma^\star$ after a finite number of steps. Furthermore, the algorithm returns `true` if $\omega \in L$ and it returns `false` if $\omega \notin L$.

[1] Another way to say this is that the string is its own ***reversal***.

```
boolean isPal (ω : Σ*) {
 1.  if (ω == λ) {
 2.    return true
 3.  } else if (ω begins with "a") {
 4.    if (|ω| == 1) {
 5.      return true
 6.    } else if (ω == a · μ · a for some string μ ∈ Σ*) {
 7.      return isPal(μ)
       } else {
 8.      return false
       }
     } else {
 9.    if (ω == 1) {
10.      return true
11.    } else if (ω == b · μ · μ for some string μ ∈ Σ*) {
12.      return isPal(μ)
       } else {
13,      return false
       }
     }
}
```

Figure 1: An Algorithm to Decide Membership in the Language L

2. Consider the Turing machine

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{accept}}, q_{\text{reject}})$$

shown in Figure 2 on page 3.

For this Turing machine, $Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_{\text{accept}}, q_{\text{reject}})$ — so neither the accepting state nor the rejecting state are shown in this picture.

Three transitions, that are not shown, go to the accepting state:

$$\delta(q_0, \sqcup) = (q_{\text{accept}}, \sqcup, \mathtt{R}),\ \delta(q_1, \sqcup) = (q_{\text{accept}}, \sqcup, \mathtt{R}, \text{ and } \delta(q_5, \sqcup) = (q_{\text{accept}}, \sqcup, \mathtt{R}).$$

On the other hand, four other transitions that are not shown, go to the rejecting state:

$$\delta(q_3, \mathtt{b}) = (q_{\text{reject}}, \mathtt{b}, \mathtt{R}),\ \delta(q_7, \mathtt{a}) = (q_{\text{reject}}, \mathtt{a}, \mathtt{R}),$$

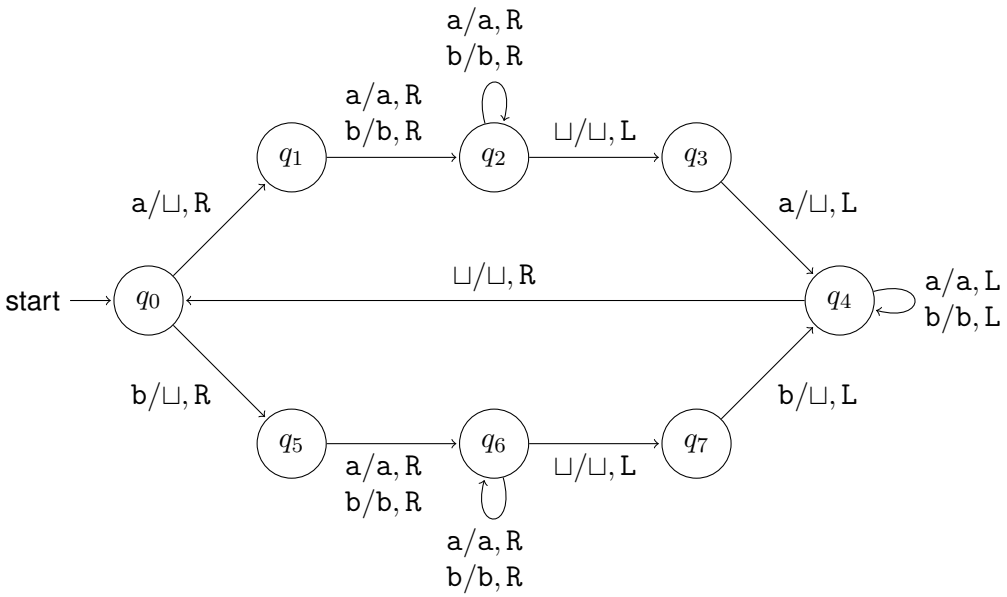

Figure 2: A Turing Machine that Decides the Language L

and $\delta(q, \sqcup) = (q_{\text{reject}}, \sqcup, \mathtt{R})$ when $q = q_3$ and when $q = q_7$.

It can be argued that this Turing machine implements the "`isPal`" algorithm that is given in Figure 1.

Use this information to prove the following.

Claim. *Let h be a non-negative integer, let $\omega \in \Sigma^\star$, and suppose that M is in configuration*

$$\sqcup^h q_0\, \omega$$

If $\omega \in L$ then M moves to an accepting configuration after a finite number of steps. On the other hand, if $\omega \notin L$ then M moves to a rejecting configuration after a finite number of steps.

Note that this claim implies that M decides the language L, as desired.