CPSC 351 — Tutorial Exercise #12

Multi-Tape Turing Machines

About This Exercise

This exercise is intended to give you practice working with textbf*multi-tape Turing machines*. Thus it gives you practice applying material introduced in Lecture #12.

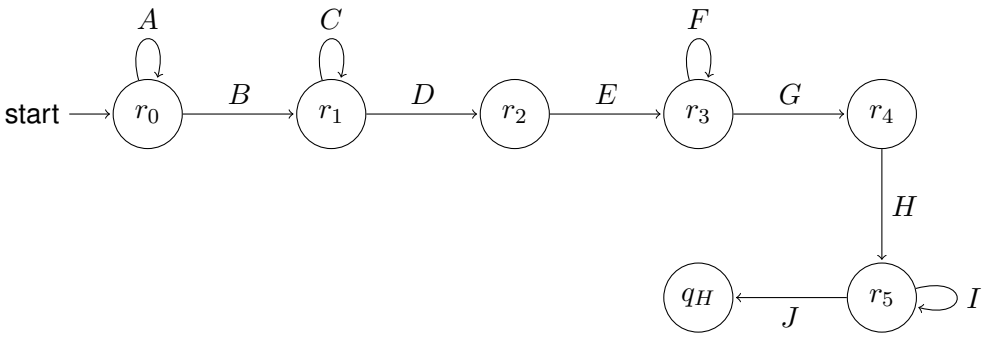
Getting Started

Initial questions concern a ***component*** of a 3-tape Turing machine, with input alphabet $\Sigma_{\text{pair}} = \{0, 1, \#\}$, tape alphabet $\Gamma = \{0, 1, \#, \dot{0}, \dot{1}, \dot{\#}, \sqcup\}$, and, and with transitions as shown in Figure 1 pn page 2. In order to keep the picture simpler, the halt state is shown in the diagram as "q_H" instead of q_{halt}. Transitions out of r_0, r_1, r_2, r_3, r_4 or r_5 that are not shown will never be used (for the computation being considered) but should go to the halting state, without changing the contents of tapes or positions of tape heads.

1. To begin, suppose consider an execution of this component that begins with the Turing machine in state r_0, with $\dot{0}1$ on the first tape, with the tape head pointing to the copy of "1", with $\dot{0}$ on the second tape, with the tape point pointing to the copy of "$\sqcup$" immediately to the right of the "$\dot{0}$" on the tape, and with third tape head filled copies of "$\sqcup$" and the tape head pointing to the leftmost cell — so that the Turing machine is in the configuration represented by the string

   $$\dot{0}\, r_0\, 1\, \sharp\, \dot{0}\, r_0\, \sharp\, r_0$$

   Show that, after a finite number of steps, the execution halts, with all tape heads filled with copies of "$\sqcup$" and with all tape heads pointing to the leftmost cells of the tapes.



A : For all $\sigma \in \Sigma_{\text{pair}}$, $\delta(r_0, \sigma, \sqcup, \sqcup) = (r_0, (\sigma, \mathtt{R}), (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}))$
B: $\delta(r_0, \sqcup, \sqcup, \sqcup) = (r_1, (\sqcup, \mathtt{L}), (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}))$
C: For all $\sigma \in \Sigma_{\text{pair}}$, $\delta(r_1, \sigma, \sqcup, \sqcup) = (r_1, (\sqcup, \mathtt{L}), (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}))$
D: For all $\sigma \in \{\dot{0}, \dot{1}, \dot{\#}\}$, $\delta(r_1, \sigma, \sqcup, \sqcup) = (r_2, (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}))$
E: $\delta(r_2, \sqcup, \sqcup, \sqcup) = (r_3, (\sqcup, \mathtt{S}), (\sqcup, \mathtt{L}), (\sqcup, \mathtt{S}))$
F: For all $\sigma \in \Sigma_{\text{pair}}$, $\delta(r_3, \sqcup, \sigma, \sqcup) = (r_3, (\sqcup, \mathtt{S}), (\sqcup, \mathtt{L}), (\sqcup, \mathtt{S}))$
G: For all $\sigma \in \{\dot{0}, \dot{1}, \sqcup\}$, $\delta(r_3, \sqcup, \sigma, \sqcup) = (r_4, (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}))$
H: $\delta(r_4, \sqcup, \sqcup, \sqcup) = (r_5, (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}), (\sqcup, \mathtt{L}))$
I: For all $\sigma \in \Sigma_{\text{pair}}$, $\delta(r_5, \sqcup, \sqcup, \sigma) = (r_5, (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}), (\sqcup, \mathtt{L}))$
J: For all $\sigma \in \{\dot{0}, \dot{1}, \sqcup\}$, $\delta(r_5, \sqcup, \sqcup, \sigma) = (q_{\text{halt}}, (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S}), (\sqcup, \mathtt{S})))$

Figure 1: Component of a 3-Tape Turing Machine

Problems Discussed in the Tutorial

Analyzing the Component in Figure 1

2. Consider an execution of component in Figure 1 that begins with the Turing machine in a configuration represented by a string

$$\mu_L \, r_0 \, \mu_R \, \sharp \, \mu_2 \, r_0 \, \sharp \, \mu_3 \, r_0$$

where μ_L begins with either $\dot{0}$ or $\dot{1}$ and all other symbols in μ_L and μ_R are in Σ_{pair}, either μ_2 begins with one of $\dot{0}$ or $\dot{1}$ and all the other symbols in μ_2 are in $\{0, 1\}$, or $\mu_2 = \lambda$, and either μ_3 begins with one of $\dot{0}$ or $\dot{1}$ and all the other symbols in μ_3 are in $\{0, 1\}$, or $\mu_3 = \lambda$.

   (a) ***Briefly*** explain why $\mu_L \, r_0 \, \mu_R \, \sharp \, \mu_2 \, r_0 \, \sharp \, \mu_3 \, r_0 \vdash^\star \mu_L \, \mu_R \, r_0 \, \sharp \, \mu_2 \, r_0 \, \sharp \, \mu_3 \, r_0$.

   (b) ***Briefly*** explain why $\mu_L \, \mu_R \, r_0 \, \sharp \, \mu_2 \, r_0 \, \sharp \, \mu_3 \, r_0 \vdash^\star r_2 \, \sharp \, \mu_2 \, r_2 \mid \mu_3 \, r_2$.

   (c) ***Briefly*** explain why $r_2 \, \sharp \, \mu_2 \, r_2 \, \sharp \, \mu_3 \, r_2 \vdash^\star r_4 \, \sharp \, r_4 \, \sharp \, \mu_3 \, r_4$.

(d) ***Briefly*** explain why $r_4 \sharp r_4 \sharp \mu_3\, r_4 \vdash^\star q_{\text{halt}} \sharp q_{\text{halt}} \sharp q_{\text{halt}}$.

It follows from this that this component can be used to continue from a configuration

$$\mu_L\, r_0\, \mu_R \sharp \mu_2\, r_0 \sharp \mu_3\, r_0$$

— with μ_L, μ_R, μ_2 and μ_3 as described here — to the end of a computation in which the empty string is returned as output.

This *begins* the design and analysis of a component, of a 3-tape Turing machine, that implements step 1 of the algorithm, for the addition of binary numbers, discussed in the presentation for Lecture #12. The additional practice problems, for this tutorial exercise, complete the design and analysis of such a component — by solving more problems like the above ones.

Subtraction: Implementing an Algorithm for the Binary "Monus" Operation

Let n and m be non-negative integers. Then $\boldsymbol{n}$ ***monus*** $\boldsymbol{m}$, written "$n \dot{-} m$", is as follows:

$$n \dot{-} m = \begin{cases} n - m & \text{if } \geq m, \\ 0 & \text{if } n < m. \end{cases}$$

Once again, let $\Sigma_{\text{binary}} = \{0, 1\}$ and let $\Sigma_{\text{pair}} = \{0, 1, \#\}$. Consider a function

$$f_{\text{monus}} : \Sigma^\star_{\text{pair}} \to \Sigma^\star_{\text{binary}}$$

such that the following properties are satisfied for every string $\omega \in \Sigma^\star_{\text{pair}}$:

- If $\omega = \mu \# \nu$, where $\mu, \nu \in \Sigma^\star_{\text{binary}}$ are the unpadded binary representations of a pair of non-negative integers n and m, respectively, then $f_{\text{monus}}(\omega)$ is the unpadded binary representation of $n \dot{-} m$.
- Otherwise $f_{\text{monus}}(\omega) = \lambda$, the empty string.

Now consider the algorithm shown in Figure on page 4.

3. Prove that if this algorithm is executed with a string $\omega \in \Sigma^\star_{\text{pair}}$ as input then the execution of the algorithm halts, with $f_{\text{monus}}(\omega)$ returned as output.
4. As noted above, problems on the practice exercise complete the design of a component of a 3-tape Turing machine that implements step #1 of the algorithm, for the addition of binary numbers, discussed in the presentation for Lecture #12. You may assume — when solving this problem — that this component can be modified, so that implements step #1 of the algorithm shown here, in Figure #1, instead.

On input $\omega \in \Sigma_{\text{pair}}$:

1. If ω has the form $\mu \# \nu$, where μ and ν are the unpadded binary representations of non-negative integers then write μ onto Tape #3 and write ν onto Tape #2, erasing Tape #1 and moving all tape heads to their leftmost positions. That is, go from the configuration

$$q_0 \, \omega \, \sharp \, q_0 \, \sharp \, q_0 \;\; = \;\; q_0 \, \mu \# \nu \, \sharp \, q_0 \, \sharp \, q_0$$

to a configuration

$$q_1 \, \sharp \, q_1 \, \nu \, \sharp \, q_1 \, \mu$$

and continue to the next step. On the other hand, if ω does not have this form then erase the first tape, moving the tape head back to its leftmost position, and halt — that is, go from the configuration $q_0 \, \omega \, \sharp \, q_0 \, \sharp \, q_0$ to the configuration

$$q_{\text{halt}} \, \sharp \, q_{\text{halt}} \, \sharp \, q_{\text{halt}}$$

Let n and m be the non-negative integers whose representations are on Tapes #3 and #2, respectively.

2. `while` $((n > 0)$ `and` $(m > 0))$ `{`
3. $\quad n := n - 1$ (updating Tape #3 to make this change)
4. $\quad m := m - 1$ (updating Tape #2 to make this change)
   `}`
5. Erase the non-blank string that is now on Tape #2 — so that the tape head for this points to its leftmost cell — and go to state q_{halt}, in order to return the unpadded binary representation of n.

Figure 2: An Algorithm to Compute the Binary "Monus" Function

Describe how to modify the Turing machine to compute f_{add}, discussed in the presentation for Lecture #12, to obtain a 3-tape Turing machine to compute the function f_{monus}, and explain, briefly, why your Turing machine is correct.