Theory of Computer Science

G. Röger
Spring Term 2019

University of Basel
Computer Science

Exercise meeting 6 — Solutions

Exercise 6.1

Consider a Turing machine with $E = \{q_E\}$, which encodes the function $f(\mathtt{a}^n) = (\mathtt{ab})^n$ for $n \in \mathbb{N}_0$. Will the Turing machine stop on the following input? If so, what does the final configuration look like?

(i) `a`

Solution:

The TM stops with final configuration $\langle \square \ldots \square, q_E, \mathtt{ab}\square \ldots \square \rangle$

(ii) `aa`

Solution:

The TM stops with final configuration $\langle \square \ldots \square, q_E, \mathtt{abab}\square \ldots \square \rangle$

(iii) ε

Solution:

The TM stops with final configuration $\langle \square \ldots \square, q_E, \square\square \ldots \square \rangle$

(iv) `ab`

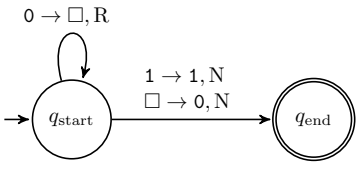
Solution:

The input is invalid. Hence, there are two possibilities: (1) The TM remains in an infinite loop, or (2), the TM stops in an invalid configuration, (Head not at the beginning of the output, $\square$ in the middle of the output or symbols from $\Gamma \setminus \Sigma$ on the tape.)

Exercise 6.2

(a) Specify a Turing machine which removes a prefix of zeros from an input over $\Sigma = \{0, 1\}$. Let the result be 0 if the input consists of zeros only or is ε.

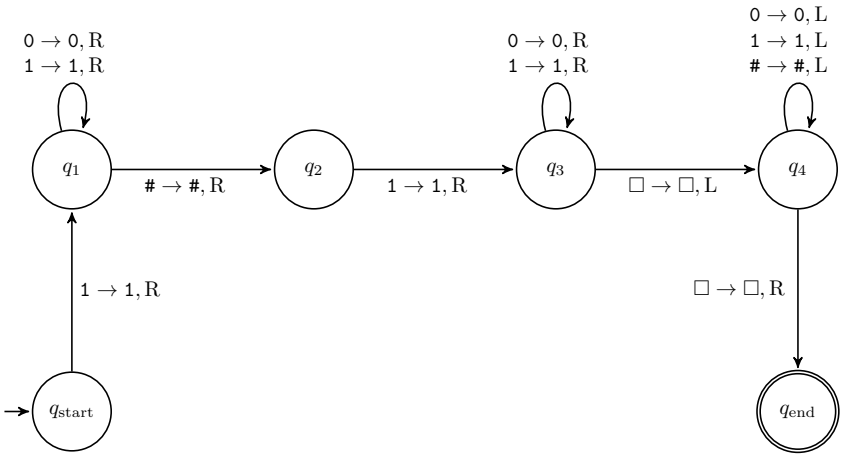
Solution:



(b) Specify a Turing machine which accepts an input over $\Sigma = \{0, 1, \#\}$ if and only if the input encodes two positive binary numbers, separated by a single #-sign.

Solution:

All missing transitions lead to an error state.



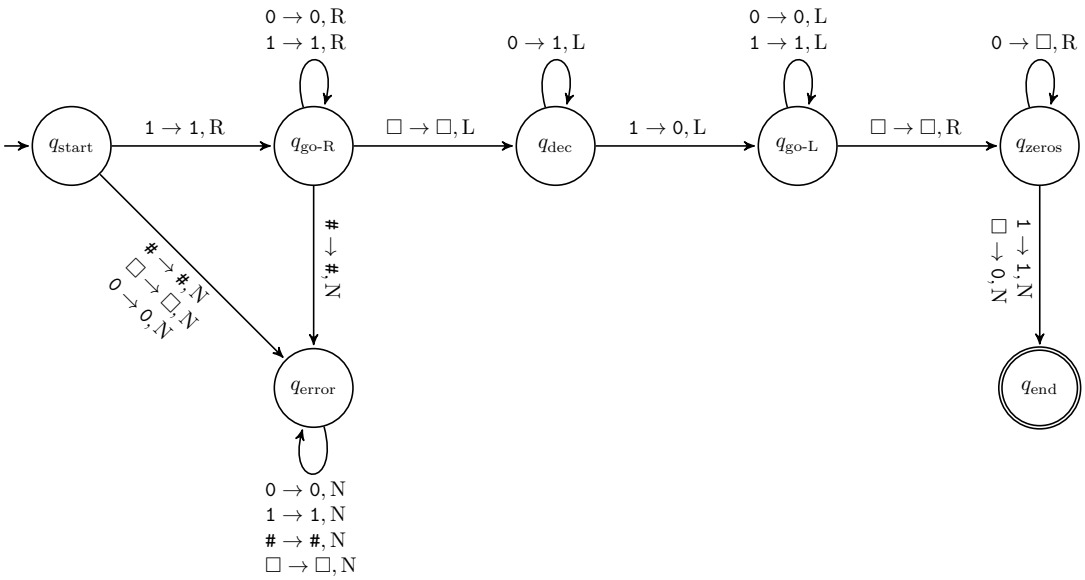
Exercise 6.3

Specify the transition diagram of a Turing machine which computes the *predecessor function* $pred_2 : \mathbb{N}_0 \to \mathbb{N}_0$ over the natural numbers:

$$pred_2(n) = \begin{cases} n-1 & \text{if } n \geq 1 \\ \text{undefined} & \text{if } n = 0 \end{cases}$$

Solution:

According to the definition of computing a numerical function with a DTM we need to specify a DTM M which computes the *encoding* $pred_2^{\text{code}} : \{\texttt{0}, \texttt{1}, \texttt{\#}\}^* \to \{\texttt{0}, \texttt{1}, \texttt{\#}\}^*$ of $pred_2$; this means it represents the function which computes $bin(pred_2(n))$ given input $bin(n)$.
Note: According to the definition of computability of functions in $\Sigma^* \to \Sigma^*$, this means, strictly speaking, that we also need to ensure that when an *invalid* input is given (for example input words such as `#011##`, `11#10`, ε or `011`), M must not stop in a valid configuration. We ensure this by leading M into an error state q_{error} if an invalid input is given, in which it will always stay.
We use the input alphabet $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{\#}\}$ as required by the definition of Turing-computability for numerical functions. As the tape alphabet we use $\Gamma = \Sigma \cup \{\square\}$, adding no additional symbols except the blank symbol. q_{start} is our start state and the only end state is q_{end}. The full definition of our DTM is thus $M = \langle Q, \Sigma, \Gamma, \delta, q_{\text{start}}, \square, \{q_{\text{end}}\}\rangle$ with $Q = \{q_{\text{start}}, z_{\text{error}}, q_{\text{go-R}}, q_{\text{dec}}, q_{\text{go-L}}, q_{\text{zeros}}, q_{\text{end}}\}$ and the transition function δ which is defined in the following state diagram. We omit transitions which cannot occur (like reading `#` in state $q_{\text{go-L}}$). These transitions can be chosen arbitrarily.



Our DTM works as follows:

- We start in start state q_{start} and test the first input symbol. There are two possibilities:
  - If the input word starts with `0`, `#` or $\square$ it is invalid. In these cases we move into the error state.
  - If the input word starts with `1`, further processing is needed, which starts in state $q_{\text{go-R}}$.
- State $q_{\text{go-R}}$ should move to the right end of the input (for inputs of type `1`...) and test whether the input is valid. If it is not valid (which is the case iff a `#` occurs), we move into the error state. Else we move into the state q_{dec} once we arrived at the right end of the input.
- Wen arriving q_{dec} the DTM checks the last digit of the input word. This state should decrement the number (reduce it by 1). To this end we replace `0` with `1` and move to the left until we reach a `1`. This digit gets replaced by `0`, finishing the decrementing.
- Afterwards we move to state $q_{\text{go-L}}$ and move the reading head to the left until we reach the start of the tape.
- Finally we need to remove potential leading zeroes and stop. The states q_{zeroes} and q_{end} do exactly this.