Turing Machine Definitions

Adam Yedidia

April 26, 2016

This document explains the formal definitions of Turing Machines as generated by this project. Note that most of this document also appears in *A Relatively Small Turing Machine Whose Behavior Is Independent of Set Theory*, which can be found at:

`parsimony/tex/busybeaver/busybeaver.pdf`

1 2-Symbol Turing Machines

This section explores how the 2-symbol Turing machines that are generated by this project are defined.

There are many slightly different definitions of Turing machines. For example, some definitions allow the machine to have multiple tapes; others only allow it to have one; some allow an arbitrarily large alphabet, while others allow only two symbols, and so on. In most research regarding Turing machines, mathematicians don't concern themselves with which of these models to use, because any one can simulate the others (usually efficiently). However, because this work is concerned with upper-bounding the exact number of states required to perform certain tasks, it's important to define the model precisely. The model we choose here is traditional for the Busy Beaver function.

Formally, a k-state Turing machine is a 7-tuple $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$, where:

Q is the set of k *states* $\{q_0, q_1, \ldots, q_{k-2}, q_{k-1}\}$
$\Gamma = \{a, b\}$ is the set of *tape alphabet symbols*
$\mathtt{a}$ is the *blank symbol*
$\Sigma =$ is the set of *input symbols*
$\delta = Q \times \Gamma \to (Q \cup F) \times \Gamma \times \{L, R\}$ is the *transition function*

q_0 is the *start state*
$F = \{\text{HALT}, \text{ERROR}\}$ is the set of *halting transitions.*

A Turing machine's *states* make up the Turing machine's easily-accessible, finite memory. The Turing machine's state is initialized to q_0.

The *tape alphabet symbols* correspond to the symbols that can be written on the Turing machine's infinite tape.

In this work, all Turing machines are run on the all-`a` input.

The *transition function* encodes the Turing machine's behavior. It takes two inputs: the current state of the Turing machine (an element of Q) and the symbol read off the tape (an element of Γ). It outputs three instructions: what state to enter (an element of Q), what symbol to write onto the tape (an element of Γ) and what direction to move the head in (an element of $\{L, R\}$). A transition function specifies the entire behavior of the Turing machine in all cases.

The *start state* is the state that the Turing machine is in at initialization.

A *halting transition* is a transition that causes the Turing machine to halt.

2 4-Symbol Turing Machines

This section explores how the 4-symbol Turing machines that are generated by this project are defined.

a k-state, 4-symbol Turing machine is a 7-tuple $M = (Q, \Gamma, b, \Sigma, \delta, q_0, F)$, where:

Q is the set of k *states* $\{q_0, q_1, \ldots, q_{k-2}, q_{k-1}\}$
$\Gamma = \{\texttt{\_}, \texttt{1}, \texttt{H}, \texttt{E}\}$ is the set of *tape alphabet symbols*
_ is the *blank symbol*
$\Sigma =$ is the set of *input symbols*
$\delta = Q \times \Gamma \to (Q \cup F) \times \Gamma \times \{L, -, R\}$ is the *transition function*
q_0 is the *start state*
$F = \{\text{HALT}, \text{ERROR}\}$ is the set of *halting transitions.*