# Equivalent models to TMs

a. Let a multitape Turing Machine be like a normal TM except that it has multiple tapes: all but the first tape start blank; at each computation step it reads one symbol from each tape, writes one symbol to each tape, and moves all the read/write heads (not necessarily in the same direction).

   i What is the formal type signature for the transition function $\delta$ of a $k$-tape TM?

   ii Describe how to simulate a multitape TM with a normal TM. (Give some implementation detail but not a full formal construction.)

b. A Nondeterministic Turing Machine (NTM) is defined basically as you'd expect; it accepts if any nondeterministic computation branch accepts. (However it is only considered to be a *decider* if *every* branch always halts.)

   We can think of the computation branches of an NTM on a given input as forming a tree, and attempt to simulate the NTM by deterministically searching the tree for an accepting configuration.

   i Why would a depth-first search not be a viable strategy?

ii Given an NTM $(Q, \Sigma, \Gamma, \delta, q_0, q_{accept}, q_{reject})$, what is the maximum number of children one node in its computation tree can have?

iii Describe how to simulate an NTM with a normal TM. (Give some implementation detail but not a full formal construction.) *Hint: instead of working with a normal TM, you can use any model that we've shown to be equivalent (since the final step in your construction could always be "Finally, use our standard construction to simulate this other model with a normal TM."). Your solution can be* extremely *'inefficient' in terms of runtime.*

c. Here are a few very open-ended prompts without exact answers.

   i Are there any ways real computers seem more powerful than TMs, and do you see any way to simulate those features using a TM?

   ii Are there any ways a TM is more powerful than a real computer?

   iii Soon we will see some undecidable languages, so we might wish for a more powerful model than the TM. Is it possible to come up with a mathematical model of computation that is more powerful than Turing Machines? If so, is it possible to come up with a real-world physical computer that acts like this model?

d. We will explore one extremely different-looking (and silly) model of computation as evidence that our model of computation is 'robust'. First for background, a "counter machine" is like a TM except that its tape is read-only, and instead it has one or more non-negative integer registers which it can increment, decrement, and test for being zero. (You can think of each register like a stack which has a $ at the bottom and then can only push and pop 1s above that.) Additionally, for our purposes we will ignore the tape by assuming that the input is encoded as an integer and then directly placed in one of the counters to begin with rather than on the tape. You may assume without proof that such machines with two counters are equivalent to TMs.

   A FRACTRAN program is a finite list of fractions. To run it on an input of value $n$, take $5 * 7^n$, and then repeatedly multiply by the first fraction in the list that produces an integer. For example, the program $[\frac{3}{2}, \frac{1}{2}, \frac{55}{9}, \frac{4}{7}]$ runs on input 1 as follows: $5 * 7^1 = 35$, $35 * \frac{4}{7} = 20$, $20 * \frac{3}{2} = 30$, $30 * \frac{3}{2} = 45$, $45 * \frac{55}{9} = 2475$, so then it halts with output 2475. (Note that the $\frac{1}{2}$ is 'dead code' since there is no circumstance where it would be used rather than $\frac{3}{2}$.) We will consider an output of 2 to be an accept and anything else to be a reject. Describe at a high level how to simulate a two-counter machine using a FRACTRAN program.