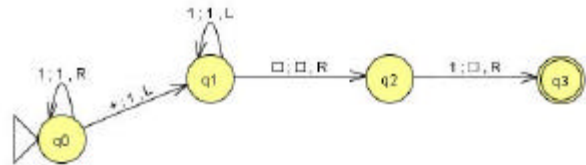


Exercise A4, 2/1/05

Part 1: Test-Drive Turing Machines

Once your problem is represented as a set (precisely and discretely), you can analyze various computational properties in a formal manner. The standard tool to analyze “computability” (e.g., whether an algorithm exists for the problem) is the Turing machine (TM). By understanding this tool, we can access the rich literature and also make a lot of connections with other forms of analytical tools. So, in this exercise, you test-drive them, using a simulator called JFLAP (<http://www.cs.duke.edu/~rodger/tools/jflap/>; links are clickable on-line; also linked from the on-line syllabus). A copy of the program (JFLAP.jar) and some examples (*.TM, *.TM.jff) are available locally (<http://www.tcnj.edu/~komagata/csc460/05s/JFLAP/>).

Task 1: Using JFLAP, design a TM that would loop regardless of the input. Choose one of the following options to show your work: (1) attach/copy/draw (the screen shot of) the transition diagram of your TM, or (2) include the formal definition of your TM. In addition, concisely explain how your TM would work.



TM $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, F)$

- Q : finite set of states
- Σ : finite set of input symbols, $\square \notin \Sigma$
- Γ : finite set of tape symbols, $\square \in \Gamma, \Sigma \subseteq \Gamma$
- δ : transition function $Q \times \Gamma \rightarrow Q \times \Gamma \times \{Left, Right\}$
- q_0 : the start state $q_0 \in Q$
- \square : blank symbol (B in the text)
- F : set of accept states $F \subseteq Q$

Task 2: Consider a problem represented as $\{x+y=z \mid \text{when } x, y, \text{ and } z \text{ are interpreted as non-negative binary numbers in some standard representation, the arithmetic relation } x + y = z \text{ holds}\}$. In this representation, you may consider “ $x+y=z$ ” as a variant form of (x, y, z) with special delimiters. Repeat the process as in **Task 1**. Note that this task will be challenging, tedious, or both. You can stop when you feel you spent enough time.

Task 3: Define an interesting, non-trivial problem of your own *as a set*. Then, repeat the process as in the earlier **Tasks**. Again, you can stop when you feel you spent enough time.

Part 2 (optional): TM Variations

Note: Depending on your time and interest, you decide whether to do this part.

If you look up references, you will occasionally find different versions of TMs. Initially, it might be confusing. However, if you think carefully, it is possible to see those variations as different ways of representing the same notion of “computation.” But what does it mean to be the “same?” We will discuss this point more in detail in Module B. This part of the exercise is a preview and can be considered as an extended exercise on TMs.

Task 1: In another Theory of Computation textbook (Sipser, 1997), Turing machines are defined differently from our version on the following points:

1. Use of a semi-infinite tape. The input string must be placed at the left end of the tape.

2. There are exactly one accept state *and* exactly one reject state.

Despite the difference, we can still say that this version is “equivalent” to our version. Try to explain how to establish such equivalence.

Task 2: Most “decent” extensions of the standard TM (cf. Text Sec. 8.4) are known to be still equivalent to the standard version (i.e., our version). Can you think of an extension of the standard TM that could do *more* than what the standard TM can do [cf. my response to your Unit A3 Summary Questions Q6/A6]? Even if you cannot come up with such an extension, can you still describe how to analyze whether a certain given TM can do more than the standard TM?

Survey: Time spent between classes: _____

// End