Lec-4-HW-1-TM-basics

PART I, Readings. I tried to put the shortest and simplest first. The notation for TMs is not standardized: you will encounter slight variations from the notation we used in class. Below are mostly web links, but a couple refer to .pdf files in CourseDocuments/120-Readings/. See which of these seem most worth reading.

What is a Turing Machine? The simple idea of a TM, with a simple example. Part of course materials from University of Cambridge. Subsequent pages refer to building a TM on a breadboard:
http://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/turing-machine/one.html

Michel, P. ***Turing Machines: an Introduction.*** A very brief intro to TMs, but with a longer computational example, and the mathematical formalisms. Is intended as an intro to the busy beaver problem:
http://www.logique.jussieu.fr/~michel/tmi.html

Sedgewick, R. ***Turing Machines***. A short intro to TMs for the CS-1 course at Princeton. Uses a slightly different syntax. Shows several finite-state diagrams and steps through a computation:
http://introcs.cs.princeton.edu/java/74turing/

Lodder, J. ***An Introduction to Turing Machines.*** A short historical introduction, which moves into reading excerpts of Turing's paper "On Computable Numbers, with an Application to the Entscheidungsproblem". The second part is more complex. Both follow closely Turing's original formulations, which have some abbreviated syntax. The third part describes the pedagogical project.
http://www.math.nmsu.edu/hist_projects/turingI.pdf
http://www.math.nmsu.edu/hist_projects/turingII.pdf
http://www.math.nmsu.edu/hist_projects/j13.html

Schmidt, FormalModels**.** A long and detailed explanation of FSMs, Turing Machines, including general automata theory and languages with lots of examples. Uses mathematical notation from the very start, which might not make it the best for a first look.
http://www.rci.rutgers.edu/~cfs/472_html/TM/FormalModelsToc.html

Hodges-AlanTuringTheLogicalAndPhysicalBasisOfComputing-2004.pdf. Historically oriented discussion of Turing's motivations and subsequent developments in defining computation, mind, and other topics.

Kondo-ReactionDiffusionModelAsFrameworkForUnderstandingBiologicalPatternFormation-2010.pdf. Turing Machines applied to biological processes.

The Alan Turing Internet Scrapbook. Lots of interesting bits and pieces about TMs and Turing, including the mathematical/logical/historical context and many links for further reading:
http://www.turing.org.uk/turing/scrapbook/machine.html

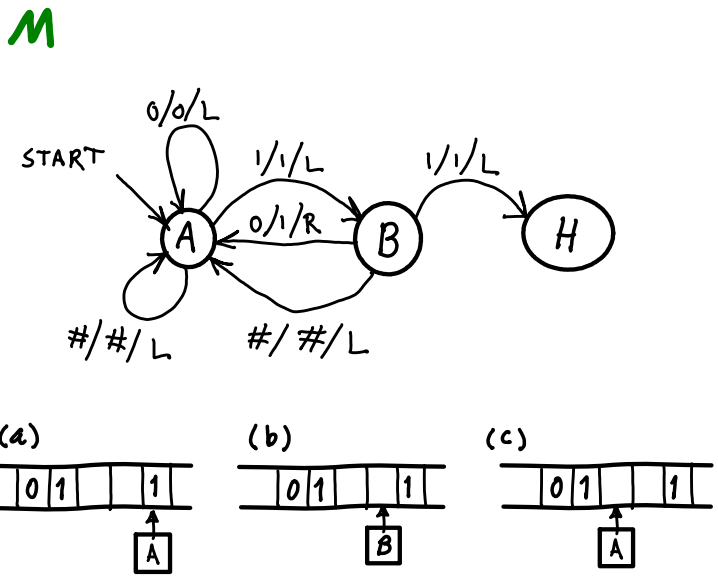
The Alan Turing Archive for the History of Computing. Original publications, including previously classified material on code breaking.
http://www.alanturing.net

Introduction to Finite Automata. A long and mathematically complete explanation with Java applets for animation. Starts with simple vending machine examples, but quickly turns to mathematical treatment. Lots of examples of finite-state machines. Goes through DFAs, NFAs, regular languages, grammars, CFGs, Turing Machines, the Halting Problem, time complexity, P and NP, and NP completeness.
http://www.cs.odu.edu/~toida/nerzic/390teched/regular/fa/delta-star.html

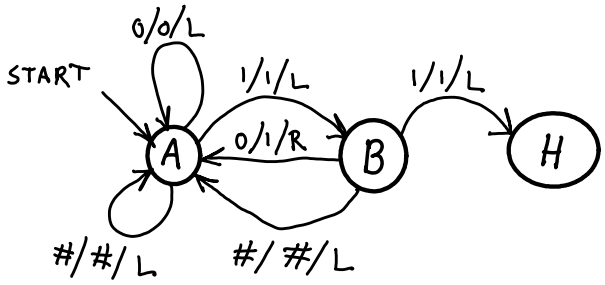
The FSM for a Turing Machine (TM), M, is shown at right. Its symbol set is { #, 0, 1}. "#" stands for a blank cell. M initially starts in state A. A state that has no next-state transitions is a halting state. If M ever transitions to state H, it halts.

Shown are three of M's machine configurations:
(a) shows M's initial tape, the initial position of its R/W head, and its initial state;
(b) shows M's configuration after the 1st state transition;
(c) is after the 2nd state transition.

Q. Continuing in the same fashion, show M's configurations step-by-step until M halts.



Q. M's FSM is shown below. Alter it so that H is no longer a halting state: make H a looping state. That is, add state-transition arcs so that if M ever enters state H, it always transitions back to state H regardless of what input M is currently reading.

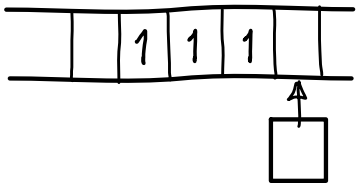


Q.
Design Tdec, a TM, that does unary decrement by one. A legal, initial tape consists of a contiguous set of cells, each containing a "1", surrounded by blank tape to both left and right. Assume the R/W head is initially positioned on the first blank cell to the right of the string of 1s (see example configuration below). Have Tdec decrement from the left end of the string, and halt with its R/W head back in its initial position.

Show a state-transition diagram for Tdec with state-transition arcs labeled with "input/output/move" notation. Legal input is any **positive** integer encoded using this unary encoding,

```
unary code     value (in decimal notation)
   1              0
   11             1
   111            2
   ...   (and so on)
```

Blank tape cells are indicated w/ "#". Your symbol set can be any set of symbol, so long as it includes



Here is a table of rules for a TM, R:

state	input	output	move	next-state
X	A	A	L	X
X	B	A	R	Y
Y	A	B	R	Y
Y	B	B	L	X

R starts in state X; its set of states is {X, Y}; its symbol set is {A, B}.

Q. Draw R's state-transition diagram.

Q. Encode R's rule table in unary: For each state pick a unary code and replace the state with its code; for each symbol pick a unary code and replace the symbol with the code; finally, for the moves, pick encodings and replace them as well.

state	input	output	move	next-state
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—
—	—	—	—	—

Q. What does R do when started reading a tape full of As? Bs?

Below is shown a UTM's tape with an encoding of some TM, M. The UTM's symbol set is {0, 1, #}. Note that exactly one of these symbols is in each UTM tape cell. M's symbol set is {A, B, #}, encoded as {1, 11, 111}; M's states are {X, Y, Z}, encoded as {1, 11, 111}. "#" means a blank cell.

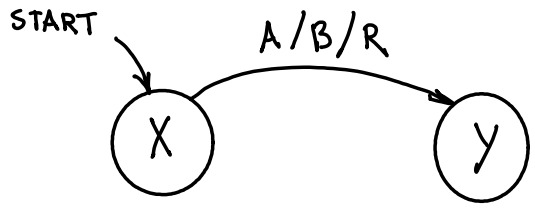
100#01#00100001#1#11#11#1101#11#1#1#101#111#1#1#1011#1#1#11#11011#11#1#1#1011#111#1#1#1110001

Here's the interpretation of the input tape shown above (left-to-right):

100 --- left end of M's simulated tape
--- marks left side of a cell of M's tape
0 --- indicates M's **R/W** head is reading this cell
1 --- encoding of A in the cell being read
--- tape cell's right-side marker
00 --- right end of the encoding of M's tape
1 --- encoding of M's current state, X
00 --- separator for beginning of M's rules
01#1#11#11#110 --- The first of M's rules: { state=X, input=A, output=B, move=R, next-state=Y }
...
001 --- right end of M's rules

The UTM expands M's simulated tape as needed by moving existing encoded tape cells to create space for new ones. If M gets into a state for which no rules are in the table, the UTM stops the simulation. Such a state is a halting state for M.

Q. Draw M's state-transition diagram. That is, draw and label M's states, and draw arrows showing state transitions. Label the arrows "input/output/move". E.g., as shown below, the first rule corresponds to an arrow from state X to state Y labeled "A/B/R", which stands for "input=A, output=B, move=R".



Q. Is there any limit on how large a TM is? That is, is there a largest TM, in the sense that it has the maximum number of states, or the maximum number of symbols, or the maximum combined number of states and symbols? Is there any limit on the largest TM that can be described using the encoding scheme in the previous problem? Given a UTM, is there a largest TM it can simulate?