6.004 Computation Structures
Spring 2009

Models of computation

Problem 1. In lecture, we saw an enumeration of FSMs having the property that every FSM that can be built is equivalent to some FSM in that enumeration.

A. We didn't deal with FSMs having different numbers of inputs and outputs. Where will we find a 5-input, 3-output FSM in our enumeration?

B. Can we also enumerate finite combinational logic functions? If so, describe such an enumeration; if not, explain your reasoning.

C. Why do 6-3s think they own this enumeration trick? Can we come up with a scheme for enumerating functions of continuous variables, e.g. an enumeration that will include things like sin(x), op amps, etc?

Problem 2. We saw that certain functions, such as parentheses checking, cannot be performed by any finite state machine. Which of the following can be performed by an FSM? Assume, in each case, that the device is to take a series of 0s and 1s that represent the digits of a binary number entered left-to-right. The device is to have a single output, which is 1 only under the specified conditions:

A. ★ When the last 277 digits entered have been alternate 1s and 0s.

B. ★ When more 0s than 1s have been entered.

C. ★ When the number entered thus far is divisible by 3.

D. ★ When an odd number of 1s and and even number of 0s have been entered.

E. ★ When the number entered corresponds to a year in which the Red Sox win the world series.

Problem 3. Recall that we refer to a Turing machine's tape configuration as bounded if all 1s recorded on the tape are within some finite distance from the initial head position. We saw in lecture that every bounded tape configuration can be viewed as an encoding of a binary number, and that a TM can be said

to compute the integer function f if, for every n, starting that TM with a tape encoding n will result in its halting with the tape encoding f(n).

A. Given TMs that compute f(x) and g(x), respectively, describe how to construct a TM that computes f(g(x)).

Problem 4.

A. ★ Ben Bitdiddle's proposed Ph.D. thesis involves writing a program to compute a function f(x) on a Cray supercomputer. Ben's advisor points out that f cannot be computed on any Turing machine. Should Ben care? Why?

B. ★ Discouraged by your answer to the last question, Ben has turned his attention to an alternative thesis topic. He now proposes to invent the universal FSM, which will be to FSMs what a universal Turing machine is to Turing machines. Ben's idea is to build an FSM that can fed a sequence of inputs describing any other FSM and the inputs to that FSM. The universal FSM would then emulate the behavior of the described FSM on the specified inputs. Is Ben's idea workable? Why or why not?

Problem 5. We saw in lecture that the function Halts(k, j) which determines whether TM k halts with the argument j is uncomputable. For each of the following functions, describe whether that function is computable or not and explain your reasoning.

A. ★ HaltsBefore(k, j, s) = 1 if TM k halts with argument j within s steps, else 0.

B. ★ HZero(k) which determines whether TM k halts with the argument zero. Hence HZero(k) returns 1 iff TM k (0) halts, else 0. [HINT: this is tricky].

C. ★ H12345(x) which determines whether TM 12345 halts with the argument 12345.

D. ★ Dow(y) = the final value of the Dow Jones average on the last trading day of the year 2000+y, for y < 100 (and zero for y >= 100).

Problem 6. In the following problems consider a Turing machines with the following specifications.

Each Turing machine has n states labeled {S1, S2, ..., Sn}, the Turing machine begins in state S1 and halts by transitioning to the special state S0, each cell of the Turing machine's tape can contain either a "1" or a "0", and each move of the Turing machine based solely on its current state and the value of the tape cell under the tape head. A move consists of first modifying the contents of the current tape cell under the head of the Turing machine, moving the tape either left, L, or right, R, followed by a transition to the next state.

The following truth table defines the behavior of a Turing machine. Note that in this FSM the outputs are a function of both the current state and the tape value.

Current State	Tape Value	Write Tape	Move	Next State
S1	0	1	L	S2
S1	1	1	R	S3
S2	0	1	R	S1
S2	1	1	L	S2
S3	0	1	R	S2
S3	1	1	R	S0

A. How large a ROM is required to implement an n-state Turing machine that adheres to the given specification (give the number of words and the bits-per-word)?

B. Given an infinite tape (in both directions), with "0"s in every cell. What will the Turing machine described by the truth table above leave on the tape when it halts? Show the status of the tape in the region around the head after each move.

C. Design a 2-state Turing Machine that writes as many "1"s as possible onto an all zero (in both directions) tape and then halts. Hint: you can write four "1"s.

D. Suppose that we choose to ignore the value of the cell under the read head, thus turning our Turing machine into a finite-state machine. In this case, how many "1"s can we write if the FSM has n states?

Problem 7. **"Extra Credit" -- hard problem.**

Given the uncomputability of the halting problem for arbitrary TMs, lets consider the halting problem for FSMs.

Assume that Fi(x) is the ith FSM in our FSM enumeration, where the first input is given the successive binary digits of x from low-to-high order and zeros thereafter. Thus x represents an input sequence

containing a bounded number of 1s. All other inputs to the FSM are tied to 0.

We interpret the first output of the FSM as the "Halt" signal -- if it ever becomes 1, the computation stops and Fi(x) is said to halt.

A. Consider the function FSMHalts(x, y) which returns 1 if Fx(y) halts, else 0. Is FSMHalts a computable function?