**474 HW 15 problems (highlighted problems are the ones to turn in)**

**The main assignment sheet has several Q and A from previous term's Piazza forums**

17.11
(#1) 6-3

17.12
(#2) 6-6-6-6

Note on 17.12a:

L = { [<x>, <f(x)>], x∈ℕ}, where <x> means "the binary encoding of x" and <f(x)> means "the binary encoding of f(x)"

17.12b,c: Do these constructions for a general function-computing TM, not specifically for the successor function.

(c) You might find the concept of "dovetailing" helpful for this problem. If you have not seen that technique before, this reference will probably help:

http://lambda-the-ultimate.org/node/322

**11.** Prove rigorously that the set of regular languages is a *proper* subset of D.

**12.** In this question, we explore the equivalence between function computation and language recognition as performed by Turing machines. For simplicity, we will consider only functions from the nonnegative integers to the nonnegative integers (both encoded in binary). But the ideas of these questions apply to any computable function. We'll start with the following definition:

- Define the *graph* of a function $f$ to be the set of all strings of the form $[x, f(x)]$, where $x$ is the binary encoding of a nonnegative integer, and $f(x)$ is the binary encoding of the result of applying $f$ to $x$.

For example, the graph of the function *succ* is the set $\{[0,1],[1,10],[10,11],\ldots\}$.

**a.** Describe in clear English an algorithm that, given a Turing machine $M$ that computes $f$, constructs a Turing machine $M'$ that decides the language $L$ that contains exactly the graph of $f$.

**b.** Describe in clear English an algorithm that, given a Turing machine $M$ that decides the language $L$ that contains the graph of some function $f$, constructs a Turing machine $M'$ that computes $f$.

**c.** A function is said to be partial if it may be undefined for some arguments. If we extend the ideas of this exercise to partial functions, then we do not require that the Turing machine that computes $f$ halt if it is given some input $x$ for which $f(x)$ is undefined. Then $L$ (the graph language for $f$), will contain entries of the form $[x, f(x)]$ for only those values of $x$ for which $f$ is defined. In that case, it may not be possible to decide $L$, but it will be possible to semidecide it. Do your constructions for parts (a) and (b) work if the function $f$ is partial? If not, explain how you could modify them so they will work correctly. By "work", we mean:

- For part (a): Given a Turing machine that computes $f(x)$ for all values on which $f$ is defined, build a Turing machine that semidecides the language $L$ that contains exactly the graph of $f$;
- For part (b): Given a Turing machine that semidecides the graph language of $f$ (and thus accepts all strings of the form$[x, f(x)]$ when $f(x)$ is defined), build a Turing machine that computes $f$.

17.13
(#3) 3

**13.** What is the minimum number of tapes required to implement a universal Turing machine?

18.1a
(#4) 9

**1.** Church's Thesis makes the claim that all reasonable formal models of computation are equivalent. And we showed in, Section 17.4, a construction that proved that a simple accumulator/register machine can be implemented as a Turing machine. By extending that construction, we can show that any computer can be implemented as a Turing machine. So the existence of a decision procedure (stated in any notation that makes the algorithm clear) to answer a question means that the question is decidable by a Turing machine.

Now suppose that we take an arbitrary question for which a decision procedure exists. If the question can be reformulated as a language, then the language will be in D iff there exists a decision procedure to answer the question. For each of the following problems, your answers should be a precise description of an algorithm. It need not be the description of a Turing Machine:

**a.** Let $L = \{<M> : M$ is a DFSM that doesn't accept any string containing an odd number of 1's$\}$. Show that $L$ is in D.

18.1b
(#5) 9

**b.** Let $L = \{<E> : E$ is a regular expression that describes a language that contains at least one string $w$ that contains 111 as a substring$\}$. Show that $L$ is in D.

Problem #6 A TM M has tape alphabet { □, a, b} (this is the order used in the encoding <M>).

<M> = (q00,a00,q00,a00, →), (q00,a10,q01,a10,→), (q00,a01,y10,a10, ←), (q01,a01,q00,a10,→), (q01,a10,n11,a01, ←)

(a) (6) Provide a transition diagram or a transition table for the TM M.
(b) (3) For each of the following outcomes of running M, provide a short string of a's and b's that is
accepted by M,
rejected by M,
neither.