COS487 Problem Set 3

Spyridon Triantafyllis

December 8, 1999

Problem 2

(i)

Class 1. This is just the regular language $0|1(0|1)^*$, if we demand that a binary representation of a nonzero integer begins with 1, and the universal language otherwise.

(ii)

Class 1. All the powers of 2 have a binary representation of the form $10 \ldots 0$. Thus, this language is equivalent to the regular language 10^*.

(iii)

Class 3. Obviously we can build a Turing Machine which checks if the input is in the form $10\ldots0$, then counts the zeros and decides if the number of zeros is prime. However, it is obvious that primality cannot be tested by a DFA or a PDA.

(iv)

Class 2. This language can be equivalently defined as:

$$\{10\} \cup \{10^{n-1}10^n \mid n \geq 1\}$$

which is context - free but not regular.

(v)

Class 1. If M accepts no strings, this language is the empty language, which is regular. If on the other hand M accepts some strings, let k denote the length of the smallest string accepted by M. Then the language becomes:

$$\{[n]_2 \mid n \geq k\}$$

which is regular.

Problem 3

This language is decidable. An algorithm for this language can be built by refining the algorithm that decides whether a CFG is empty. The algorithm will mark symbols of the language with two different markers; e for symbols that can produce a string of even length, and o for symbols that can produce a string of odd length (a symbol can have both markers). First of all the algorithm marks all terminals with o. Then it goes into a loop. During each pass of the loop the algorithm examines all (marked and unmarked) nonterminal symbols. If for a nonterminal N there is a rule $N \to \alpha$, where α consists only of marked symbols and an even number of symbols of α are marked with o, then N is marked with e; if an odd number of symbols in α are marked with o, then N is marked with o. The loop stops when the start symbol is marked with e, or when no new markings can be made. In the first case the algorithm outputs yes, and in the second case it outputs no.

Problem 4

(a)

This is obvious; just take the identity function as the mapping function.

(b)

If f is a computable mapping function such that $w \in A \Leftrightarrow f(w) \in B$ and g is a computable mapping function such that $w \in B \Leftrightarrow g(w) \in C$, then $f \circ g$ is a computable mapping function such that $w \in A \Leftrightarrow f(g(w)) \in C$.

(c)

Since $A \leq_m B$, there is a computable function f such that $w \in A$ iff $f(w) \in B$. But this is equivalent to saying that $w \notin A$ iff $f(w) \notin B$. Thus, $\overline{A} \leq_m \overline{B}$.

(d)

According to (c), it must also be the case that $\overline{A} \leq_m A$, which means that $\overline{A}$ is enumerable. Since both A and $\overline{A}$ are enumerable, A is decidable.

(e)

Let f be a function such that $f(w) = 01$ if $w \in A$ and $f(w) = 10$ if $w \notin A$. The function f is decidable, since A is decidable, and obviously $w \in A$ iff $f(w) \in 0^*1^*$.

(f)

Let M be an enumerator for A. We can built a TM N which accepts some string w as input, then simulates the enumerator M and halts if/when w appears in the output stream of M. Clearly, N on input w halts iff $w \in A$. Thus, the decidable function $f(w) = \langle N, w \rangle$ has the property that $w \in A$ iff $f(w) = \langle N, w \rangle \in A_{TM}$. Thus $A \leq_m A_{TM}$.

Problem 5

It is obviously decidable whether $u < v$ for any strings u, v. Also, it is obvious that we can effectively enumerate all binary strings in increasing order.

Suppose that the language A is decidable. Then there is a TM M that decides it. Built an enumerator N for A as follows: Produce all strings w in increasing order, and for each w simulate M on input w, and output w if the simulation returns "yes". Obviously, N enumerates all strings in A in increasing order.

Suppose now that an enumerator N enumerates all strings of A in increasing order. Build a TM M as follows: M on input w simulates the enumerator N until it encounters a string $v \geq w$. This is going to happen sooner or later, since the outputs of N must be increasing. Thus, M is guaranteed to stop. Now, M can output "yes" if $v = w$ and "no" if $v > w$. Obviously, M decides A.

Problem 6

Since B is recognizable, we can build an enumerator N for it. Consider now a TM $M'_i, i \in \mathbb{N}$, which simulates N, discards N's first $i-1$ outputs, and then simulates the i'th output of N on its input. We can fix the implementation of the M'_i's in such a way that it can be mechanically verified if a given TM is an M'_i for some i. Then, the language $C = \{M'_i \mid i \in \mathbb{N}\}$ will be decidable, and each TM $M \in C$ will have an equivalent TM in B and vice versa.

Problem 7

We can calculate the representations of two TMs N_1 and N_2 with the following properties: Both machines run all three machines M_1, M_2 and M_3 with corresponding inputs x_1, x_2 and x_3 in parallel. N_1 stops when one of these machines stops, and outputs the number of the machine that stopped. N_2 stops when two of these machines stop, and outputs the numbers of the machines that stopped.

We can now build a TM N which does the following. First, it calculates the representations of N_1 and N_2. Then it asks the oracle if N_2 is in $\mathrm{HALT}_{\mathrm{TM}}$. If it is, then it runs N_2 (which outputs the machines that stopped). Then it asks the oracle if the remaining machine with the corresponding input stops. If it does, it writes the number of this machine in its output.

If N_2 is not in $\mathrm{HALT}_{\mathrm{TM}}$, then we know that either one or none of the machines stop. The next step is to ask the oracle if N_1 is in $\mathrm{HALT}_{\mathrm{TM}}$. If it is, then N simulates N_1 (which outputs the number of the machine that stops). If it is not, then none of the machines stop.

Problem 8

It is more convenient to show first that $\mathcal{R}$ has the same multiplicity as the interval $(0,1)$. This is easy to show; consider the function $f(x) = \frac{1}{1+\exp(-x)}$. This function is a 1-1 and onto mapping from $\mathcal{R}$ to $(0,1)$.

a

We will prove this in two steps; first we will create a 1-1 mapping from $(0,1)$ to $2^{\mathbb{N}}$, and then we will create a 1-1 mapping from $2^{\mathbb{N}}$ to $\mathcal{R}$.

Each number in the interval $(0,1)$ can be represented as an infinite string of binary digits, of the form $0.a_1a_2\ldots a_n\ldots$. There is some ambiguity here, between numbers ending with an infinite string of zeros and numbers ending with an infinite string of 1's. For example, 0.1 can be represented either as $0.1000\ldots$, or as $0.0111\ldots$. Let us make the convention that we will always prefer the representations ending with an infinite string of 1's in such cases. Now, consider a function that maps the number $0.a_1a_2\ldots a_n\ldots$ to the set of integers $\{2^{a_1}, 3^{a_2}, \ldots, p_n^{a_n}, \ldots,$ where p_n is the n'th prime number. This mapping is obviously 1-1.

On the other hand, consider any set of integers S. If $S = \emptyset$, we map it to the number 0. If it is not empty, we order the elements of S in order of increasing size. We thus get a sequence (a_n). If the set has only n elements, we can set $a_m = a_n, m > n$. Now we can map this set to the continued fraction

$$a_1 + \frac{1}{a_2 + \frac{1}{a_3 + \cdots}}$$

Obviously, this mapping is a 1-1 mapping from $2^{\mathbb{N}}$ to $\mathcal{R}$.

b

It suffices to show that $(0,1)^2$ has the same size as $(0,1)$. A 1-1 and onto mapping from $(0,1)^2$ to $(0,1)$ can be achieved by interleaving the digits; a pair of numbers $(0.a_1a_2\ldots a_n\ldots, 0.b_1b_2\ldots b_n\ldots)$ is mapped to the number $0.a_1b_1a_2b_2\ldots a_nb_n\ldots$.