COSC 2001(A) 3.0—Fall 2000

Date: Nov 7, 2000
Due: Nov 28, 2000

Problem Set No. 3—The last one!

Papers ***must*** **be typed or word-processed (the "*must*" does not apply to diagrams)**, and deposited in the course drop-box(es) on the due date.

▶ **Due time:** Any time on November 28, 2000. **Boxes will be cleared the following morning. Location of the drop-boxes:** There are **two** boxes—labelled 2001A and 2001B—on the first floor of CCB, in the corridor that leads to the Ariel Lab.◀

In this Problem Set it is allowed—but not required!—to submit **ONE joint paper that has a total of TWO co-authors from the same section**. The same mark, as assigned to such a joint paper, will be given to **each** of its two authors.

▶ **IFF** you are submitting Problem Set #3 ***with*** a partner, then you ***must*** notify me as described below, **Prtnr1.–Prtnr4.:**

Prtnr1. Make a file called "partner" (no quotes). [Please do ***not*** call it "Partner" or "PARTNER" or "a3partner" or anything other than "partner"].

Prtnr2. Put in it your name and "ariel" login, ***and*** the name and ariel login of your partner as well.

Prtnr3. Give the following command on ariel

"**submit 2001 a3 partner**"

NOT later than 5:00pm, Nov. 21, 2000.

Prtnr4. Only ***one*** submission (**Prtnr3.**, above) ***per pair*** please! ◀

If you do NOT plan to work with a partner please do NOT submit any co-author information!

(1) This teamwork is **strictly for "declared" pairs**.

(2) Any strong similarity between different papers will be seriously frowned upon. (To learn more about this issue please follow the link "**Senate Policies**" found on the URL: http://www.cs.yorku.ca/~gt/courses/)

General Remark. Each solution must contain *adequate explanation(s)* of *why* it answers the relevant question. While examples can help one to understand your point of view, *they are NOT substitutes* for a logical argument that establishes your solution's validity ***in general***.

1. For the grammar G of problem 3 in Problem Set #2 ("the MATH1090 connection") produce a PDA M using the procedure in Sipser, or that from class (the two are almost identical), so that $L(G) = L(M)$.

2. "***Universal DFA do not exist***". OK, let us substantiate the preceding statement.

My part: We can easily see that *all* possible DFA with tape alphabet $\{0, 1\}$ can be coded as strings over a fixed alphabet (I am describing this for your benefit in the following few lines—nothing for you to prove here):

Indeed, let us code a DFA M: Each "instruction" $\delta(q_i, a) = q_j$ is represented by the string $q\tilde{i} * a * q\tilde{j}$, where $a \in \{0, 1\}$ and $\tilde{i}$ is the decimal representation of the number i. Thus, adding ";" as a ***new*** symbol, we represent the automaton by "gluing" the instruction-representations, one after the other, using ";" as inter-instruction glue, and appending at the end the sequence the string "$; q\tilde{m}; q\tilde{n}; \ldots ; q\tilde{k};$" which indicates that q_m is the *initial* state and $q_n, \ldots, q_k$ are the *final* states.

Any automaton, such as M, has more than one string representation (due to the fact that permutations of states and/or instructions are possible) over the alphabet $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, q, *, ;\}$.

$R(M)$ will denote ***all*** the representations of M.

End of the description of how to code a DFA.

Your part now: Prove that the language

$$L = \big\{x; y : (\exists M)\big(M \text{ is a DFA } \ \& \ y \in R(M) \ \& \ x \in L(M)\big)\big\}$$

is ***not*** regular, that is, in plain English, "there is ***NO*** DFA U which can faithfully simulate an ***arbitrary*** DFA M (coded as y) on ***arbitrary*** input x".

3. Prove that CFLs are closed under concatenation, reversal, and Kleene star.

By "closure under reversal" we mean this: For any language L define

$$L^R \stackrel{\text{def}}{=} \{x^R : x \in L\}$$

Recall that for a ***string*** x, "x^R" is the string obtained by reading x from right to left. Thus, "closure under reversal" means that if L is a CFL, then so is L^R.

Recall that ϕ_x denotes the function ***of one argument*** computed by the Turing Machine, M_x, found in position x of the "standard" (lexicographic) enumeration of all Turing Machines (over a fixed input alphabet Σ).

4. Prove that $\{x : \mathrm{dom}(\phi_x) = \emptyset\}$ is not decidable.

Note. By "dom" we mean "domain", that is, for any function f of one argument

$$\mathrm{dom}(f) = \{x : (\exists z) f(x) = z\}$$

5. **YES/NO**, but ***with proof*** please!

(a) The problem $\varepsilon \in L(G)$ is decidable for any CFG G.

(b) The problem $0 \in \mathrm{dom}(\phi_x)$ is decidable.

(c) The problem $0 \in \mathrm{dom}(\phi_x)$ is semi-decidable (i.e., the set $\{x : 0 \in \mathrm{dom}(\phi_x)\}$ is recognizable).

6. **For extra credit (10 points) and some amusement. You can still get full marks without doing this ADDITIONAL problem.**

Fix a TM (input) alphabet as always. In what follows, "output" means numerical (in $\mathbb{N}$) output, as always.

Now, define a function of one variable, $b(n)$, by

$b(0) = 1$ and, for all $n > 0$,

$$b(n) \stackrel{\mathrm{def}}{=} \max\{\text{output of an } n\text{-state TM that starts with a blank tape and eventually halts}\}$$

Prove

(1) $b(n)$ is total.

(2) For any total computable function $f(n)$, there is a number n_f (the subscript suggests that "n_f" depends on f) such that

$$n > n_f \text{ implies } f(n) < b(n)$$

(3) $b(n)$ is not computable.

Notes and Hints.

(I) Of the 10 extra points, (1) gets **1**, (2) gets **6**, (3) gets **3**.

(II) (3) uses (2). You can still earn your full points in (3) even if you could not do (2) (take (2)'s statement without proof).

(III) Hints for (2): Take an arbitrary computable total $f(n)$. Say, "Without loss of generality, $f(n)$ is increasing", and ***show why indeed*** there is no loss of generality in assuming so. Then say "Let the TM M of, say, s states compute $f(2n+2)+1$." Wait a minute! Can you say that? Why?

Now take $n_f = 2s$ and take $n > 2s$ (you want to prove $b(n) > f(n)$ for all $n > n_f$, right?). Next, build a TM, N, of exactly n states, that uses $n - s$ of those states to build the number $n - s - 1$ on tape, starting with a blank tape.

Next N uses M as a subprogram. You do the rest and fill in all the details that I left out.