Final Exam

Francisco J. Díaz Riollano
Student ID: 802-15-2172

December 11, 2018

1) (25 pts.) The argument to prove that the complement of a regular language is also a regular language is:

"Since L is regular, there is Deterministic Finite State Automaton D that recognizes L. But then, the Deterministic Finite State Automaton $\bar{D}$ obtained interchanging accept and non-accept states of D, recognizes $\bar{L}$, the complement of L."

Is the argument valid when Deterministic FSA is replaced with Non-deterministic FSA and no transformation from Non-deterministic to Deterministic FSA is invoked?

Proof:

Let A be a non-deterministic finite state machine $A = (Q, \Sigma, \delta, q_0, F)$ and $L = L(A)$ be the language that the finite state machine recognizes. We want to prove the existance of another non-deterministic finite state machine that recognizes $\bar{L}$. The idea is to build a machine A' which accepts when A rejects. Let $A' = (Q, \Sigma, \delta, q_0, Q - F)$, where $Q - F$ is the set of states that are in Q but not in F. Since F contains the set of accepting states, then any input string w that does not land on one these states, the machine A, is said to reject and by the construction above A' will thus accept. Thus A' will accept anything input string of the form $w \in \bar{L}$, where $\bar{L} = \Sigma^* - L$. This proves that non-deterministic machines can recognize the complements of regular languages and that the complement of regular languages are also regular languages.

2) (25 pts). Let $L = \{< a, b, c, p >$: a,b,c and p are integers $a^b \equiv c \mod p\}$. Demonstrate that L is in P.

By defintion $a^b \equiv c \mod p$ is the same as $p | a^b - c$, which read p divides $a^b - c$. As a direct consequence of the above $a^b \equiv c \mod p$ iff $a^b \mod p = c \mod p$. Roughly speaking, congruence $\iff$ same remainder. We could design a Turing Machine M that accepts if and only if a^b and c have the same remainder when taken the modulus p

// E is a subroutine

E = " On input $< x, y >$ where x, y are integers in binary
1) Repeat until $x < y$
    Assign $x = \lceil x/y \rceil$
2) Output x"

M = “ On input $< a, b, c, p >$, where a, b, c, p are integers in binary
1) Run E on $< a^b, p >$
    Assign the output to x
2) Run E on $< c, p >$
    Assign the output to y
3) If $x == y$, *accept*
4) $reject$

The complexity of the division in subroutine E is $O(n^2)$, where n is an n-digit number. Since subroutine E will do at most k iterations. In algorithm M we perform two of these operations, $2O(n^2k)$ and a comparison, $O(1)$ so the algorithm is of the order $O(n^2k)$, we can guarantee that k will be no longer than n itself (this is due to asymptotic nature of division) therefore:

$L = \{< M, a, b, c, p > |M$ is a TM that checks $a^p \equiv c \mod p\}$ belongs to the class P.

3) (25 pts) Consider the problem $L = \{< T, w >$: T is a Turing Machine and w a fixed string such that T enters each of its states on input w $\}$. Is L undecidable? Provide a formal answer.

We will show that L is undecidable by a reduction of A_{TM} is reducible to L, where $A_{TM} = \{< M, w > |M$ is a TM and M accepts $w\}$.
Suppose that L is decidable and that TM R decides it. Since R solves L, we can use R to check if w visits all of the states to decide A_{TM}. Below, I will construct a TM S that “decides” A_{TM} by using the decider R for L as a subroutine.

S = “ On input $< T, w >$, where T is a TM.
1) Run TM R on input $< T, w >$
2) If R accepts, then *accept*, If R rejects, $reject$.”

However, since we know A_{TM} is undecidable, there cannot exist a TM that decides L.

4) (25 pts) Consider the problem $L = \{< G, w >: G$ is a context free grammar and w a string, such that $w \in L(G)\}$. Is L decidable? Provide a formal answer.

Any context-free language is decidable.

Proof:

Let L be a context-free language and G be the context-free grammar that generates L, then for each $w \in \Sigma^*, w \in L$ if and only if $S(< G, w >)$ accepts, thus L is decided by $S(< G, >)$, where S is the following algorithm:

procedure S (On input $< G, w >$)

1) Transform G into Chomsky normal form
2) Compute $n \leftarrow |w|$
3) Enumerate all $2n - 1$ steps generated by G
4) for each $2n - 1$ step derivations.
   if w is generated,
      accept
5) *reject*