CSC520 Q232 Final Exam

***Part A***

Due by the end of this class

Total: /50

Name (last, first): \_Ferdousi, Jannatul\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*“If you wish to understand you must …”*

# [20 points] Draw a state diagram for defining a Turning Machine to copy a given string. The given string can only contain ‘A’ or ‘B’, i.e. Z = {‘A’, ‘B’}. There need to be one (and only one) blank \_ between the given string and its copy. The result of the copied string can be located to the right or left of the given string.

# e.g. given in tape: AABBBA (the read write head is at the leftmost A, i.e. rwh = 0) (The input string can be of any length. If \_ is given, \_ will result.) Need to have AABBBA\_ AABBBA in the tape after the TM halts. (There may be any blank \_ before or after)

# (Your part B question 1 needs to be based on this state diagram.)

Diagram, letter

Description automatically generated

# [15 points] Unsolvable: (a) In general, what make a problem unsolvable? (b) How many problems are definable on natural number N? How many problems of those definable problems are solvable?

### **Unsolvable Problems:**

Algorithmic problem **undolvability may** occur only in the case where **infinitely** many cases (arguments of a function, strings of a language) are to be considered.

* e.g. One machine may not predicts the behavior of *any* other machines.

When a problem *P* is represented as a function *f*, then:

*P* is unsolvable if *f* is nonrecursive

When a problem *P* is represented as a set *Sp* then:

*P* is unsolvable if *Sp* in nonrecursively enumerable

### **Uncomputable function**

A vast majority of function on N cannot be computed.

Since all computable function *f*: N ⇒ N can be enumerated, the cardinality of the class of **all computable** functions is at most

*N*0

However, the cardinality of the class *P* of **all definable function** on N, where *P* = {*f*:(*f*: N⇒N)} is

2*N*0

### **Non-recursively enumerable set**

A set S is **not recursively enumerable** if

((i *S*) ⟹ (*fi* is total)) ∧ (∀*f, f* is total and computable ∃*i* ∈ *S* (*f* == *fi*))

* Total computable function are not recursively enumerable

(*S*tot = {*i*: *fi* is total}) is not recursively enumerable

### **Rice's Theorem (non-solvable case)**

Let F be any set of computable function.

The set *S* = {*x*: *fx* ∈ *F*} is **undecidable** ⇔ ((F != ⦰) ∧ (F != {the set of all computable functions}))

* For any non-trivial property of partial function, the question of whether a given algorithm computes a partial function with this property is undecidable.
* Total computable function are not recursively enumerable

b) the cardinality of the class *P* of **all definable function** on N, where *P* = {*f*:(*f*: N⇒N)} is

2*N*0

Since all computable function *f*: N ⇒ N can be enumerated, the cardinality of the class of **all computable** functions is at most

*N*0

# [15 points] *Complexity*: (a) Describe how to determine a problem is “less complex” than another problem. (b) Define “complexity degree”.

### **Complexity Theory**

Each problem has inherent complexity.

There is some minimum amount of work required to solve it.

### **Reducibility Ordering**

Reducibility expresses that a set A of problems is less complex than a set B of problems, by transforming each instance of A into a instance of B.

A B

A set A is reducible under F to a set B, given A and B be subsets N, and a set of functions F: N⇒N which is closed under composition, if and only there exists a *f* ∈ F such that for all *x* ∈ N, *x* ∈ A if and only if f(*x*) ∈ B

(A F B) ⇔ (∃*f*∈F ∀*x*∈N((*x* ∈ A) ⇔ (*f*(*x*) ∈ *B*)))

If F is a set of polynomial function, then the reduction is called polynomial reduction.

To define reduction in term of language, the domain 𝛴\* is used instead of N, and F is a set of computable functions.

A set *S* ⊆ *P*(N) is closed under reduction

(S is closed under ) ⇔ (∀*A∈P*(N)∀*s*∈*S*((*A*⇐ *s*) ⇔ (*A* ∈ *S*))

A ⊆ H of N is hard for *S* ⊆ *P*(N)

(H is hard for *S*) ⇔ (∀s∈S(*s* H))

A subset *C* of N is complete for *S* ⊆ *P*(N) if *C* is hard for *S* and *C* is in *S*

(*C* is complete for *S*) ⇔ ((∀*s*∈*S*(*s* *C*) ∧ (*C* ∈ *S*))

The set *C* is the **most complex** (maximal) problem in the class *S*

Any solution to C can, in combination with the reductions, be used to solve every problem in the class S.

**Properties of reduction**

The reduction relation is Reflexive and Transitive

(A B) 🡪 (( ))

(A B) ∧ (B *is recursive*)) 🡪 (*A is recursive*)

(A B) ∧ (B *is recursively enumerable*)) 🡪 (A *is recursively enumerable*)

(b) **Complexity degree**

A set A of problem has the **same complexity** under reduction r as a set B of problems if and only if A is reducible to B and B is reducible to A

(*A* =r *B*) ⇔ ((*A* r *B*) ∧ (*B* r *A*))

A complexity degree is an equivalence class under =r