CIS 306 Quiz 5 Discrete Structures II

Name: ___Demetrius Johnson__2/23/2021_____________

No calculators are allowed. Show your work. The points for each problem are indicated. Problems with incomplete work may receive partial, or no credit.

COs: [No direct COs]

Points: ______________/ 20

1. [5 pts] Let the alphabet Σ = {a, b, c}. Given the regular expression r = b(a + c)*b, assume a language L(r) is the language defined by the regular expression r. Explain in English what accepted strings in the language are like.

Accepted strings for the language L(r) is very simple given r, a regular expression, r = b(a+c)*b; the Kleene closure of a + c (a or c) means that there could be any combinations of a or c to produce a substring including the empty string: λ.
Thus, (a+c)* means $(\lambda + a+c) + (a+c)^2 + \dots + (a+c)^n$.
Having explained the Kleene closure of a+c, now we address the "b" that is attached to the start and end of r; this means that all strings for language L(r) must start and end with "b". Since (a+c)* includes lambda, the smallest string of L(r) possible is "bb". No other b's allowed other than start b and end b.

In Summary: L(r) = The language with Σ = {a, b, c}, where all strings must start with "b", and end with "b"; Any combination of a or c, including λ, are allowed between the start and ending "b". No other "b" allowed.

2. [10 pts] Give the configuration after applying the ***appropriate*** transition function from those listed, using the symbols a, b, c. Only apply the transition function **once**. PICK THE CORRECT TRANSITION FUNCTION, and apply it, giving your final configuration

Assume your **original configuration** is: **abbq₁aabac**

state 1

Reading "a"

Transition functions available:

- **δ(q_1, a) = (q_2, a, L)**
- ~~**δ(q_1, b) = (q_3, c, R)**~~ reading "b"
- ~~**δ(q_1, c) = (q_1, c, L)**~~ Reading "c"
- ~~**δ(q_2, a) = (q_3, c, R)**~~ state 2

abb $\underline{q_1}$ $\boxed{a}$ abac : go to q_2 after writing "a" and moving Left.

State reading

~~abq b aa~~

"move Left)

"a" written over "a"

$\boxed{\text{ab } \underline{q_2} \boxed{b} \text{aabac}}$

state 2

Now reading "b"

Solution.

3. [5 pts] One of the biggest open (that is, unsolved) theoretical computer science questions, is "Is P = NP". We do know that P ⊆ NP. In your own words, what does *that* mean?

Using Dr. John P. Baugh's video on Classes P and NP as a reference: In Computer science, Boolean Algebra – manifested as electrical circuits with circuit gates – is why the field really exists. Boolean Algebra is CS's best friend; it's why Computers are able to solve complex problems: the problems are broken down into the most simple, smaller problems to which a boolean algorithm can solve it.

The one caviot to boolean is that the more complex a problem becomes, the more it must be broken down, which results in more and more smaller problems to be solved. But because of the speed of electricity, and using it to calculate billions of simple boolean expressions that as a whole make up the larger, complex problem, computers are able to run such algorithms; however, here in lies the limitation: class P is a group of problems that can solve large inputs in a reasonable time frame; in the form of big O time complexity which is what polynomial time refers to. Class NP problems are not always decideable because there is no way to break down certain complex problems in a reasonable level to which a computer can compute. We know P is a subset of NP; P ⊆ NP, but we don't know if they are a proper subset: we don't know if all elements of problems of type P – which can be solved with boolean logic for large inputs – are exactly the same as all problem types in NP.

If we discovered it is true that P=NP, that would mean that many complex problems previously thought that computers cannot digest for large inputs would suddenly be able to be solved as long as we find the correct representation of the NP-type problem to match a P problem. If we found P=NP, it would electrify the Comp. Sci. world because many scientists would then know that for many complex problems, there is a reasonable boolean representation that could be developed so that computers would have more ability in computational power without having to increase processing power. Scientist would have more confidence that there is a solution to many complex problems that can be solved in order to make computers efficient at solving these problems. In fact, in my estimation, they would directly be able to use P=NP to solve numerous ~~mathematical~~ mathematical inquires and to test other theories and models. Even if it is discovered that $P \neq NP$, it could be used for many of the same purposes at least in testing other mathematical principles or developing current or new theories in mathematics and Computer Science.