COT 4210 Quiz #3 Part A: Decidability, Coountability 3/25/2021 Solutions

1) (6 pts) Prove that the following language, L_1, is a decidable language:

L_1 = { (<D>, k, m) | <D> is the encoding of a DFA which accepts at least k strings that are length m or less}

Solution

Here is a high level algorithm to decide membership in this language:

1. Set a counter x = 0
2. For each string s of length m or less:
    a. Simulate reading s into D and see if s gets accepted by D.
    b. If s was accepted by D, then add one to x.
3. Return true if x is greater than or equal k, and false otherwise.

This terminates because there are a finite # of strings of length m or less, and reading a string into a DFA is guaranteed to terminate as well. It works because it exhaustively tries all valid strings and counts exactly the number of those strings accepted by D.

Grading criteria: 1 pt for mentioning some sort of counter, 2 pts to mention looping through all strings in the "sample space", 2 pts for adding 1 to the counter if a string is accepted, 1 pt for returning true or false appropriately.

2) (9 pts) Prove that the following language L_2, is a decidable language:

L_1 = { (<D>, k, m) | <D> is the encoding of a DFA which accepts at least k strings that have a length greater than m}

Solution

This problem is harder because you can't simulate running all strings of length k or greater. But, from previous proofs, we know there's a way to detect if there are an infinite number of strings in a language, and we can exploit the reasoning behind this to come up with an algorithm that decides this problem. Here is a possible solution:

0. Set a counter v = 0.
1. Count the number of states in D. Let this be s.
2. For each string x, with a length in between s and 2s:
    a. Simulate running x on D and see if x gets accepted by D.
    b. If x was accepted by D return true
3. If m ≥ s-1 and step 2 didn't return true, return false.
3. For each string, x, of length m+1 through s-1, inclusive: (Guaranteed m < s-1…)
    a. Simulate running x on D and see if x gets accepted by D.
    b. If x was accepted, add 1 to v.
4. If v is greater than or equal to k, return true. Otherwise, return false.

We know if a DFA accepts any string of length in between s and 2s, by the pigeonhole principle, the acceptance path contains a loop and infers that an infinite number of longer strings will be accepted by the DFA (longer than any given input integer m). (This is the crucial fact upon which the pumping lemma for regular languages is based.)

If none of these strings are accepted, then we know that there exists no string that is accepted with a looping path, and no strings of length s or greater. Thus, if m is s-1 or greater, this means that there are 0 strings that satisfy the criteria and we can return false immediately.

Now, the only case remaining is if m is less than s-1. In this case, since we already know that no strings of length s or greater are accepted, we can simulate all "potential" strings and count how many of those strings are accepted.

Note: None of the explanation provided above needs to be provided to earn full credit. Any algorithm that correctly decides the problem should get full credit, including ones different than the one above. Here is rough sketch of how points should be allocated:

4 pts for some type of "loop detection" where we try to see if there exists some loop in the path of any accepted string, that terminates in a finite amount of time.

2 pts for returning true if a loop is detected in some accepting path of a string.

3 pts for handling the case where there is no string that gets accepted with a loop in its path, but where the answer is yes anyway.

3) (10 pts) Prove that the set of ordered pairs (x, y) where x and y are both positive integers with x < y is countable. To make sure you just don't copy a similar proof shown in class, in order to get full credit for this one, you need to do the following:

(a) Give an English description of how you would order the ordered pairs, making sure that each one gets listed exactly once. (So, based on your description, I should be able to design a computer program that starts running and spits out ordered pairs listed above without ever repeating one and such that it would eventually spit out any specified ordered pair.)

(b) Given the ordering you've described, what is the rank of the ordered pair (100, 200) in your ordering. **Please leave your answer as an arithmetic expression containing only integers and usual mathematical operations or functions (+, -, *, /, combos and powers).**

(c) Explain why this ordering: (1, 2), (1, 3), (1, 4), … , (2, 3), (2, 4), (2, 5), …, (3, 4), (3, 5), (3, 6), … is insufficient to show that the ordered pairs described are countable.

Solution

(a) Sort the list by ending value in numerical order, listing all ordered pairs that end in 2, the all ordered pairs that end in 3, then 4, etc,

For all the pairs that end in a specified value, v, there will be v-1 corresponding pairs. List these in order from smallest to largest of the first item in the pair, so (1, v), (2, v), …, (v-1, v).

Here is the first few ordered pairs that this algorithm would produce:

1. (1, 2) 2. (1, 3) 3. (2, 3) 4. (1, 4) 5. (2, 4) 6. (3, 4)

(b) The ordered pair (100, 200) has all ordered pairs of the form (x, y) before it where y < 200. There are $\sum_{i=1}^{198} i = \frac{198\times199}{2}$ such ordered pairs. Then, of the ordered pairs of the form (x, 200), (100, 200) is the 100th one. Thus, it's 1-based rank on the list is $\frac{198\times199}{2} + 100 = 19801$.

To verify, I wrote a small python program that generated the first 19901 values of the list:

```
cnt = 1
for y in range(2, 201):
    for x in range(1, y):
        print(cnt,". (",x,",",y,")", sep="")
        cnt += 1
```

Indeed, the 19801th line of output produced by this program is:

```
19801. (100,200)
```

(c) The ordered pairs starting with 1 would never end, and we could not give a fixed position in the list where the ordered pair (2, 1) appears. ($\infty + 1$ is not an allowable answer!)

Grading: 2 pts if the algorithm doesn't repeat anything, 2 pts if the algorithm hits all items listed, 4 pts for their computation…unfortunately, you'll have to verify this for each different valid ordering, sorry… 2 pts for the last one, use your judgement in awarding partial credit.