5.5/5

Lamess Kharfan
Student ID: 10150607
June 19, 2016

CPSC 313 – Assignment 3

1) The problem solving strategy outlined in "How to Solve It" by George Polya is a 4 step process which consists of the following steps:

   a) Understand the Problem: We cannot attempt to solve problems before we understand what they are asking. We can try to understand the problem by:
      - Stating the problem in our own words.
      - State the principal parts of the problem. Ask what is the unknown? What are the data? What is the condition?
      - Try drawing a figure to better understand the problem.
      - Begin to introduce notation.
      - Ask is it possible to satisfy the condition?

   b) Devise a plan: We devise a plan to create an outline of the calculations, computations, or constructions we need to do to solve the problem. We can do this by:
      - Guessing and checking
      - Forming ideas based on formerly acquired knowledge related to the problem
      - Try solving similar and related problems first
      - Considering all of the given data
      - Trying to work backwards

   c) Carry out the plan: The plan we devised gives a good general outline, and now we must try to make all of the details fit into the outline. This takes patience as we must check each step as we go. Ask, can I see that each step is correct? Can I prove that each step is correct? If the problem isn't working and the steps are not correct, discard the plan and attempt to devise a new one.

   d) Looking Back: Looking backwards and reconsidering the solution we came up with can help develop knowledge of the problem and knowledge of problem solving overall. Errors are always possible, so verification is desirable. Check if the result makes sense at first glance, if not, check where it may have gone wrong. Then ask, can we use this strategy to solve similar problems?

Lamess Kharfan
Student ID: 10150607
June 19, 2016

2)

a) For the alphabet Σ = {a, b, c}, write a context-free grammar for the language L1 = {w ∈ Σ * | w does not contain both a and b}:

S → S1 | S2 | ε
S1 → bS1 | cS1 | ε
S2 → aS2 | cS2 | ε

(1) Derivations:

- caac: S → S2 → cS2 → caS2 → caaS2 → caacS2 → caacε → caac
- bbb: S → S1 → bS1 → bbS1 → bbbS1 → bbbε → bbb

b) For the alphabet Σ = {a, b}, write a context-free grammar for the language L2 = {w ∈ Σ * | |w| ≡ 0 mod 3 and all the characters in the first third of w are the same}.

S → S1 | S2 | ε
S1 → aS1aa | aS1ab | aS1ba | aS1bb | ε
S2 → bS2bb | bS2ba | bS2ab | bS2aa | ε

(1)

Derivation for bbbaaaaaa:
S → S2 → bS2aa → bbS2aaaa → bbbS2aaaaaa → bbbεaaaaaa → bbbaaaaaa

c) For the alphabet Σ = {a, b, c, d}, write a context-free grammar for the language L3 = {a i b j c k d l | i + l = j + k}.

1/2 S → S1S2 | ε
S1 → aS1b | aS1c | ε
S2 → bS2d | cS2d | ε

S → S1S2 → aS1cS2 → aaS1ccS2 → aaccbS2d → aaccbd ∉ w

abbS2d
abbcS2dd
abbccS2ddd

3) Constructive proof that for any regular language L, the language STUTTER(L) = {stutter(w)|w ∈ L} is also regular:

(1) Assume L is regular. Then there exists a NFA M = (Q, ∑, d, qS, F) such that L is accepted by M. Now, our goal is to create an NFA M' such that L(M') = STUTTER(L).The state qS is the start state. According to the definition of the STUTTER(L), for every symbol in a string $w \in L$ we duplicate it and move onto the next symbol until we've done that for every symbol of w. To do this, every transition function, d(qi, x) = qj must become two transition functions, and we need to add two new non-accepting states q0 and q1. The transition functions now become d(qi, 0) = q0, and d(q0, 0) = qj for when x = 0, and d(qi, 1) = q1 and

d(q1, 1) = qj for when x =1. We accept if we have reached the end of the string and are in a state that is in F, after "stuttering" each symbol of w.

More formally, the NFA that accepts STUTTER(L) is M' = (Q' , ∑, d', qS, F):

Q' = Q U {q0, q1}
∑ = {0 , 1}
d' = d(qi, 0) = q0
d(q0, 0) = qj
d(qi, 1) = q1
d(q1, 1) = qj
qS = same start state as M
M = same final states as M

Thus, since the language L has a DFA that accepts it, and we were able to manipulate the DFA M into the DFA M' to accept stutter(L), stutter(L) is regular.

4)

a) Show that the grammar G is ambiguous

We can derive:
if condition then if condition then a = 1 else a = 1 in two ways:

- <STMT> → <IF-THEN> → if condition then <STMT> → if condition then <IF-THEN-ELSE> → if condition then if condition then <STMT> else <STMT> → if condition then if condition then <ASSIGN> else <STMT> → if condition then if condition then a = 1 else <STMT> → if condition then if condition then a = 1 else <ASSIGN> → **if condition then if condition then a = 1 else a = 1**

- <STMT> → <IF-THEN-ELSE> → if condition then <STMT> else <STMT> → if condition then <IF-THEN> else <STMT> → if condition then if condition <STMT> then else <STMT> → if condition then if condition then <ASSIGN> else <STMT> → if condition then if condition then a = 1 else <STMT> → if condition then if condition then a = 1 else <ASSIGN> → **if condition then if condition then a = 1 else a = 1**

1/2

Lamess Kharfan
Student ID: 10150607
June 19, 2016

b) Give an unambiguous grammar that generates the same language as G.

Let the grammar G = (V, Σ,<STMT>, R) be as follows.
Σ = {if, condition, then, else, a = 1}
V = {<STMT>, <ASSIGN>, <IF-THEN>, <IF-THEN-ELSE>, <NEXT>}

The set of rules R is

<STMT> → <ASSIGN> | <IF-THEN> | <IF-THEN-ELSE>
<NEXT> → <ASSIGN> | <IF-THEN-ELSE>
<IF-THEN> → if condition then <STMT>
<IF-THEN-ELSE> → if condition then <NEXT> else <STMT>
<ASSIGN> → a = 1