Ryan Darras, CS 5070 - HW06 2.(5(b,c,e,f), 44, 47)

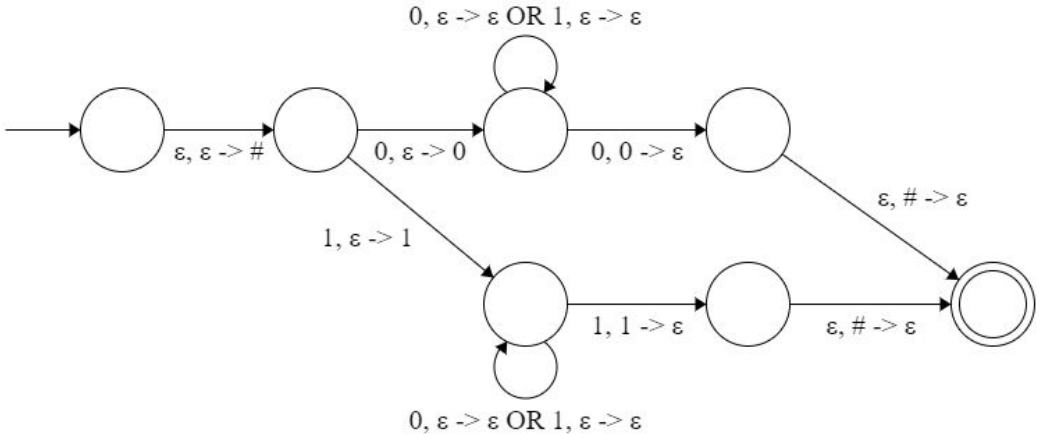
*Used a combination of https://www.cefns.nau.edu/~edo/Classes/CS315_WWW/Tools/fsm.html and paint.net to create images

Problem 2.5

Give informal descriptions and state diagrams of pushdown automata for the languages in exercise 2.4.

Problem 2.5 B Answer

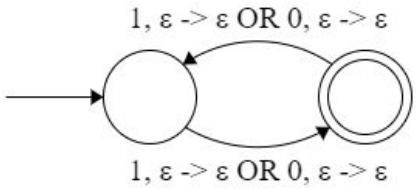
{w | w starts and ends with the same symbol}



Informal Description: Start by pushing the end symbol (#) to the stack. Then accept 0 or 1 and push it to the stack. Continue through every element in the string, but if we find our first value, pop it off of the stack and then pop the end symbol off the stack. If our string is done at this point, then the string is in the language.

Problem 2.5 C Answer

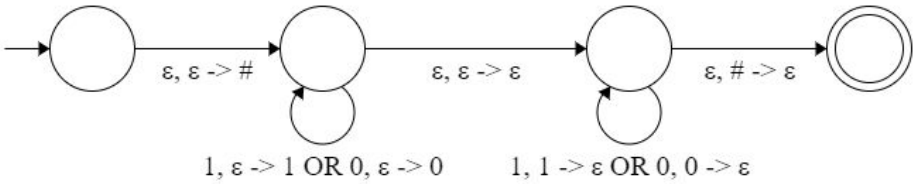
{w | the length of w is odd}



Informal Description: We don't even need to use the tools for a pushdown automata here. We can just bounce back and forth between an "odd" and "even" length node. If we land on the odd node then we accept.

Problem 2.5 E Answer

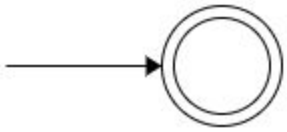
{w | w = w^R, that is, w is a palindrome}



Informal Description: Start by pushing the stack end symbol. Then start pushing all inputs to the stack. At every point, spawn off another machine by an epsilon transition that starts popping every input off the stack if it matches the input. If every element pushed onto the stack is popped, we finish by popping the stack end symbol. If the string is done at this point, then this string is in the language.

Problem 2.5 F Answer

The empty set



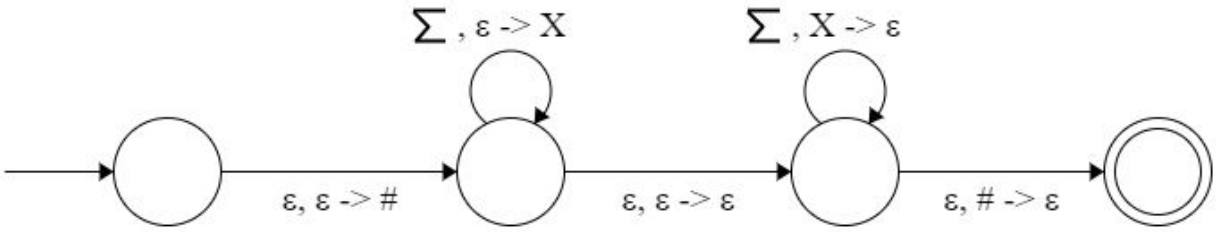
Informal Description: As long as we don't have any input after the start state, we accept the string as it is in the language. IE only accept the empty string.

Problem 2.44

If A and B are languages, define A ◊ B = {xy| x ∈ A and y ∈ B and |x| = |y|}. Show that if A and B are regular languages, then A ◊ B is a CFL.

Problem 2.44 Answer

PDA is defined as state diagram and description below.



From the start state, push the stack floor symbol to the stack. Then we progressively push any character (X in this example) to the stack on every step of the string for every letter in the alphabet of A & B. We then non deterministically (guess) to epsilon transition where we then progressively pop X off the stack for the remainder of the input string. If we reach the stack floor symbol and are at the end of the string, we recognize A ◊ B.

Proof:

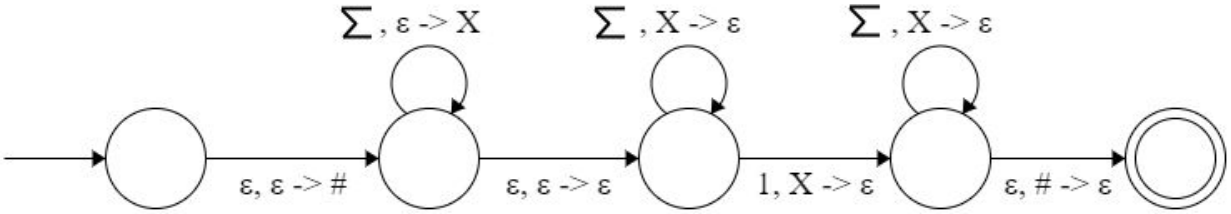
If w is not contained in A ◊ B, then either xy is not contained in AB, or for all x, y with x ∈ A, y ∈ B and w = xy, |x| != |y|. If |x| > |y|, we will get to the end of the machine and still have X's in the stack. If |x| < |y| then we will get to the end of the stack while our input string still has elements that we need to handle. In these cases if w is not contained in A ◊ B then w is not accepted by our machine. Thus A ◊ B is a context free language.

Problem 2.47

Let Σ={0,1} and let B be the collection of strings that contain at least one 1 in their second half. In other words, B = {uv| u ∈ Σ*, v ∈ Σ*1Σ* and |u|≥|v|}.

Problem 2.47 A Answer

Give a PDA that recognizes B.



Similar to the answer in 2.44, we will non deterministically guess the halfway point of the string. We start by putting the stack floor symbol on the stack (#). We will then go through each element in the string, pushing a counting character (X) to the string and guessing where the middle of the string is by taking the epsilon transition. We will then begin to pop the counting character off the stack for each element of the input string until we find a 1, at which point we will transition to a very similar loop, but this one has access to the accept state node.

Problem 2.47 B Answer

Give a CFG that generates B.

S -> UV

U -> Σ | ε | UU

V -> U1U

Σ -> 0 | 1

*Note: I know this isn't correct. I just can't figure out how to get the CFG to force U to be larger than V.