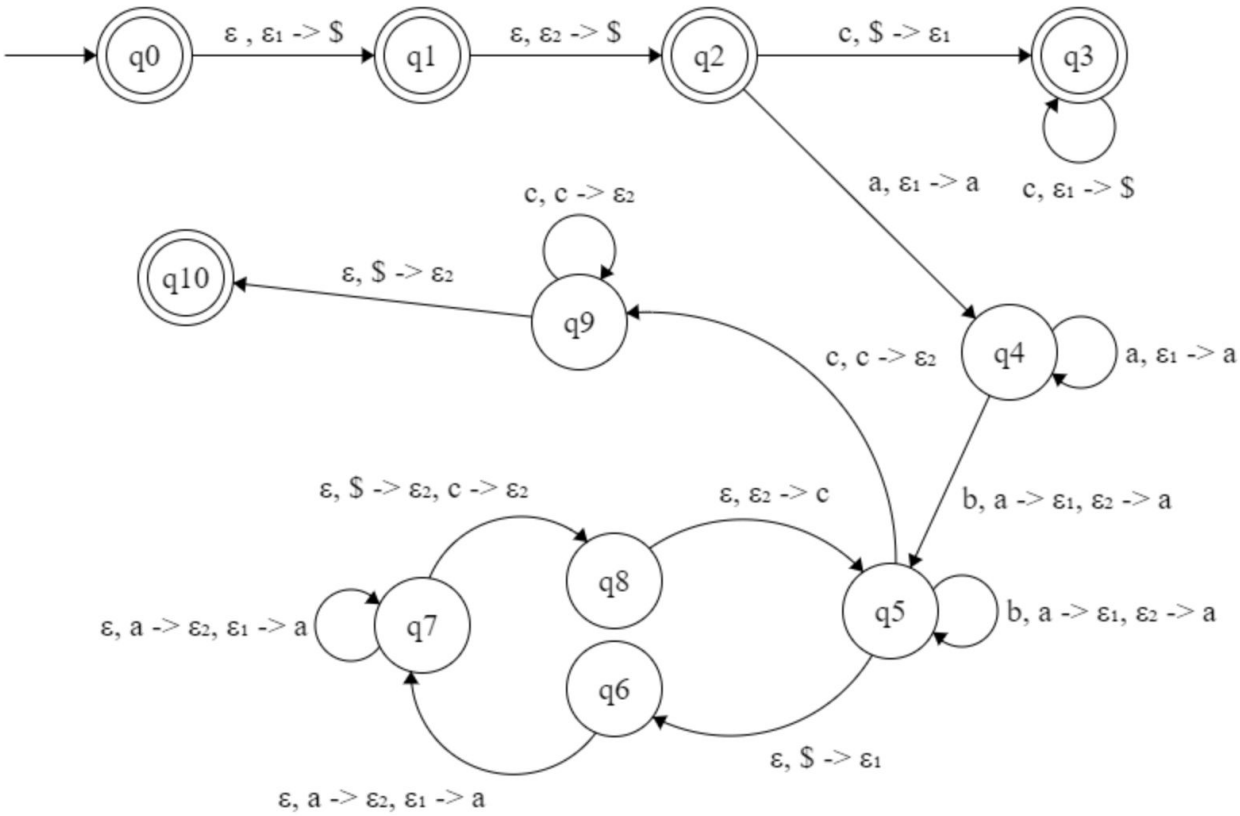
Names: Omar Abdelmotaleb and Benjamin Singleton
Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Omar Abdelmotaleb
Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Benjamin Singleton

Problem 1. (5 points) A 2-stack PDA (2PDA for short) is a pushdown automaton with two stacks. In one step each stack can be popped or pushed independently. For example, the 2PDA may read an input symbol, pop the first stack, ignore the second stack and, based on the values seen, change state, push different symbols on the two stacks. The input tape is read only.
Describe a 2-stack PDA that accepts the language $L = \{a^i b^k c^i d^k \ : i, k \geq 0\}$. You do not need to give a PDA diagram – a high-level, but complete, description of how your PDA works will suffice.

To create a 2PDA that accepts the language L = {$a^i b^k c^i d^k$: i, k >= 0} you would create 7 states arranged in a linear fashion. The first leftmost state and the last rightmost state would each be accept states. The transition from the start state to the second state would be an epsilon transition and push a $ onto each of the two stacks. The second state would transition to the third state with the symbol “a”. The third state would have a self loop on “a” which would push the symbol “a” onto stack 1. The third state would transition to the fourth on the symbol “b” and push the symbol “b” onto stack 2. The fourth state would have a self loop on “b” which would push the symbol “b” onto stack 2. The fourth state would transition to the fifth state on the symbol “c” and pop the symbol “a" off stack 1. The fifth state would have a self loop on “c” which would pop “a" off stack 1. The fifth state would transition to the sixth state on the symbol “d” and pop the symbol “b” off stack 2. The sixth state state would have a self loop on “d” which would pop “b” off stack 2. Finally, the sixth state would have an epsilon transition to the seventh and final state which would pop $ off both stack 1 and 2. If the two stacks are empty after putting a string through the 2PDA, is it in the language.

Names: Omar Abdelmotaleb and Benjamin Singleton
Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Omar Abdelmotaleb
Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Benjamin Singleton

Problem 2. (10 points) In Problem Set 6 you showed that the language $L_{mult} = \{a^i b^{ij} c^j : i, j \geq 0\}$ is not context free. Either show that this language can be recognized by a 2PDA or explain why that is not possible.



Visual 2PDA

For the language L_{mult}, there exists a 2PDA of 11 states, from q0, q1, …, q10. q0 is the start state and q1, q2, q3, and q10 are accept states, where everything else is a reject state.

From q0 to q1, we add a symbol $ to the first stack, S1, and don't read anything. From q1 to q2, we add a symbol $ to the second stack, S2. From q2 we have two outgoing states. The first outgoing state reads the symbol and checks if it is a "c" that is read. If the symbol read is a "c" and we can pop the top of S1 to get a $, meaning the stack is

Names: Omar Abdelmotaleb and Benjamin Singleton
Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Omar Abdelmotaleb
Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Benjamin Singleton
empty, this implies that the input was all c's, which loops for every c that is read in the accept state q3. q3 will loop, reading "c" symbols and adding $ symbols to S1.

The other possible outgoing state for q2 is to q4 when the symbol read is "a". If it was a "b", it would stop on the spot as we cannot read any b's at the start without having any a's, as the language implies that having no a's, meaning i = 0, that i * j would also have to be 0, so there couldn't be any b's. This outgoing state from q2 to q4 read's an "a" and pushes it to S1. In state q4, we will loop onto q4 for every "a" that is read, pushing "a" to S1 every time.

From q4 to q5, we transition if the symbol read is a "b". If it had been a "c", this would reject, as having any number of a's and c's implies that we would've needed some number of b's to be accepted. If the symbol read is a "b", then we start popping the a's S1 and push them onto S2.

From q5, we loop onto q5 for symbol "b" that is read, popping a's from S1 and pushing a's to S2 until S1 is empty. If S1 is empty, where popping S1 would reveals a $, we transition to q6. From q6, we will transition to q7, operating on S2 to pop all of the a's from there and pushing a's to S1. From q7, we keep looping onto q7 until it is empty or we hit a "c" on S2, constantly popping a's from S2 and pushing a's onto S1.

NOTE: After later explaining this, I realized q6 is unnecessary and we can transition directly from q5 to q7, but I'm just leaving this here to make easier to follow the diagram.

From q7, we transition to q8 if we hit a "c" or a $ on S2. Once in q8, we transition back to q5 while performing on S2, pushing a "c" onto S2.

We have an outgoing transition for q5 to q9 when the next symbol read is a "c". Otherwise, keep going through q5 to q8 as described in previous steps. This outgoing transition to q9 upon reading a "c" pops a "c" from S2. In q9, we will continue loop for each symbol "c" we read and we can keep popping c's from S2. If we can't pop a "c" from S2 but we are still reading "c"s, the PDA will fail. If we run out of c's to read and S2 is empty, meaning we pop a $, then it transitions to q10 where q10 is the accept state and we are done.

Names: Omar Abdelmotaleb and Benjamin Singleton
Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Omar Abdelmotaleb
Pledge: I pledge my honor that I have abided by the Stevens Honor System. -Benjamin Singleton

Problem 3. (10 points) Prove that the intersection of a CFL and a Regular language is always context free.

The intersection of a CFL and a regular language is always context free. A CFL can be expressed as a PDA and a regular language with a DFA. A PDA is more powerful than DFA because a PDA has a stack. As such, there are languages which can be expressed by a PDA that can't be expressed by a DFA. Thus, when a context free language and regular language intersect the product is a PDA and therefore context free. This can be proved with a proof by construction, which follows:

Proof

Let M_1 be a PDA that recognizes A_1, where $M_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_1, F_1)$, and
M_2 be a DFA that recognizes A_2, where $M_2 = (Q_2, \Sigma_2, \delta_2, q_2, F_2)$.

Construct M to recognize $A_1 \cap A_2$, where M = (Q, Σ, Γ, δ, q, F)

1. $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$.
   This set is the cartesian product of sets Q_1 and Q_2 and is written $Q_1 \times Q_2$. It is the set of all pairs of states, the first from Q_1 and the second from Q_2.
2. Σ, the alphabet, is the same as in M_1 and M_2.
3. Γ, the stack alphabet, is the same as in M_1.
4. δ, the transition function, is defined as follows. For each $(r_1, r_2) \in Q$, $s \in \Gamma$, and $a \in Z$, let
   $$\delta((r_1, r_2), s, a) = (\delta_1(r_1, s, a), \delta_2(r_2, s, a)).$$
   Hence δ gets a state of M, together with an input symbol and stack symbol, and returns M's next state.
5. q_0 is the pair (q_1, q_2).
6. F is the set of pairs in which both members are an accept state of M_1 or M_2. We can write it as
   $$F = \{(r_1, r_2) \mid r_1 \in F_1 \text{ and } r_2 \in F_2\}.$$
   This is the cartesian product of F_1 and F_2, written as $F_1 \times F_2$.