Exercise 3.9: Pushdown Automata with Multiple Stacks

Part (a): Show that 2-PDAs are more powerful than 1-PDAs

Theorem: 2-PDAs are strictly more powerful than 1-PDAs.

Proof: Consider the language L = {a^n b^n c^n | n ≥ 0}, which consists of strings with equal numbers of a's, b's, and c's in that order.

1. **Claim 1**: L cannot be recognized by any 1-PDA.
   - By the pumping lemma for context-free languages, L is not context-free
   - If L were context-free, for any string a^p b^p c^p where p exceeds the pumping length, we could decompose it as uvwxy where |vx| > 0, |vwx| ≤ p, and uv^i wx^i y ∈ L for all i ≥ 0
   - However, pumping would always disrupt the equality between the counts of a's, b's, and c's
   - Therefore, L is not context-free
   - Since 1-PDAs recognize exactly the context-free languages, no 1-PDA can recognize L
2. **Claim 2**: L can be recognized by a 2-PDA. Formal construction of a 2-PDA M for L:

   - M = (Q, Σ, Γ, δ, q_0, F) where:
     - Q = {q_0, q_1, q_2, q_3, q_4, q_accept, q_reject}
     - Σ = {a, b, c}
     - Γ = {X, Z_0} (X is the stack symbol, Z_0 is the initial stack marker)
     - q_0 is the start state
     - F = {q_accept}
   - Stack operation:
     - Initial configuration: (q_0, w, Z_0, Z_0) where w is the input
     - Phase 1 (q_0): Read a's, push X onto stack 1 for each a
     - Phase 2 (q_1): Read b's, for each b: pop X from stack 1 AND push X onto stack 2
     - Phase 3 (q_2): Read c's, pop X from stack 2 for each c
     - Phase 4 (q_3): After input is consumed, check if both stacks contain only Z_0
     - Accept if both stacks contain only Z_0; otherwise reject

   This 2-PDA correctly recognizes L because:
   - The number of X's pushed to stack 1 equals the number of a's
   - For each b, exactly one X is popped from stack 1 and pushed to stack 2
   - For each c, exactly one X is popped from stack 2
   - If all three counts are equal, both stacks will contain only Z_0 at the end

Since L is not recognizable by any 1-PDA but is recognizable by a 2-PDA, 2-PDAs are strictly more powerful than 1-PDAs.

Part (b): Show that 3-PDAs are not more powerful than 2-PDAs

Theorem: 3-PDAs and 2-PDAs have equivalent computational power.

Proof: I'll show that a 2-PDA can simulate a Turing machine, which implies it has the same power as any k-PDA for k ≥ 3, since all would be equivalent to Turing machines.

Formal construction: Using 2 stacks to simulate a Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, q\_accept, q\_reject)$:

Let $P = (Q', \Sigma, \Gamma', \delta', q'_0, F')$ be a 2-PDA where:

- Q' includes Q and additional states needed for simulation
- $\Gamma' = \Gamma \cup \{Z_0\}$ where Z_0 is the initial stack marker

Initially:

- Stack 1 contains Z_0 followed by the input (in reverse order)
- Stack 2 contains only Z_0

Simulation steps:

1. To read the current symbol under the TM head:
   - Peek at the top symbol X of stack 1 (excluding Z_0)
2. To simulate a TM transition $\delta(q, X) = (q', Y, D)$:
   - Pop X from stack 1
   - Based on direction D:
     - If D = R (move right):
       - Push Y onto stack 1
       - If stack 2 is empty (contains only Z_0), push blank onto stack 1
       - Otherwise, pop top symbol from stack 2 and push onto stack 1
     - If D = L (move left):
       - If stack 1 contains only Z_0, push a blank onto stack 2
       - Otherwise, pop top symbol from stack 1 and push onto stack 2
       - Push Y onto stack 1
     - If D = S (stay):
       - Push Y onto stack 1
3. Accept if the TM state is q_accept, reject if it's q_reject

This simulation faithfully recreates the behavior of a TM:

- Stack 1 represents the portion of tape from the current position leftward (in reverse)

- Stack 2 represents the portion of tape rightward from the current position
- Each configuration of the 2-PDA uniquely represents a configuration of the TM

Since a 2-PDA can simulate a TM, it can recognize all recursively enumerable languages. A 3-PDA cannot be more powerful than a TM, so 3-PDAs are not more powerful than 2-PDAs.

Conclusion: The hierarchy of k-PDAs collapses at k=2, meaning:

- 0-PDAs (NFAs) recognize regular languages
- 1-PDAs recognize context-free languages
- 2-PDAs (and any k-PDA for k ≥ 2) recognize recursively enumerable languages

Exercise 3: Decidable Languages and Standard String Ordering

Theorem: A language L is decidable if and only if there exists an enumerator that enumerates L following the standard ordering of strings.

Direction 1: If L is decidable, then L can be enumerated in standard order

Proof: Let M be a Turing machine that decides L. We construct an enumerator E that enumerates L in standard order as follows:

1. E generates all strings in Σ* in standard lexicographic order: ε, 0, 1, 00, 01, 10, 11, 000, ...
2. For each string s:
   - E runs M on s
   - If M accepts s, E outputs s
   - E proceeds to the next string in the standard ordering

Since M is a decider for L, it always halts and correctly determines whether s ∈ L. Therefore, E will systematically enumerate exactly the strings in L, following the standard ordering.

Direction 2: If L can be enumerated in standard order, then L is decidable

Proof: Let E be an enumerator that enumerates L following the standard ordering of strings. We construct a TM M that decides L:

1. On input s, M simulates E and tracks its output:
   - If E outputs s, M accepts
   - If E outputs a string lexicographically greater than s, M rejects

- If E has enumerated all strings lexicographically less than s without outputting s, M rejects
- If E halts without outputting s (possible if L is finite), M rejects

2. M is guaranteed to halt and correctly decide L because:
   - If $s \in L$, E must eventually output s before any lexicographically greater string
   - If $s \notin L$, then either:
     - E will output a string lexicographically greater than s (allowing M to reject)
     - E will enumerate all strings in L without outputting s (allowing M to reject)
     - E will halt after enumerating all strings in L (if L is finite)

The critical property is that the standard ordering provides a definitive point at which M can determine $s \notin L$: when E outputs a lexicographically greater string or completes enumeration of all strings less than or equal to s.

Conclusion: A language L is decidable if and only if there exists an enumerator that enumerates L following the standard ordering of strings.