PS06-04

February 12, 2018

To prove this, I will posit two machines: one a 2-PDA that can emulate a Turing Machine and the other Turing machine that can emulate a 2-PDA.

a. A 2-PDA can emulate a Turing machine by shuffling items between its two stacks. Let M be a 2-PDA with stacks X and Y. *General idea:*
When given a string, M will "read" the input onto the "tape" and then use epsilon transitions to move between states until it reaches an accept state.
In order to put the input onto the "tape", the characters from the input string are all pushed onto X. In order to move right, pop Y onto X. If Y has the $\vdash$(is empty), then push a $\textvisiblespace$ onto X in order to simulate infiniteness. In order to move left, pop X onto Y. In this way, a 2-PDA can pop a character, push the character onto the other stack, and transition states accordingly until it reaches a an accept state(or forever, if the given language is r.e.)

b. A Turing Machine can emulate a 2-PDA using 3 tracks. The first track is the input track, where the input string goes. The second and third represent the two stacks. The Turing machine could write to either, but only to the blank adjacent to the rightmost written character, and if it read a character, it would have to write a blank where said character was, thus preserving the stacklike behavior. As long as the Turing machine only wrote to either of the stack tracks, it would accurately emulate a 2-PDA.