CS 1511 Homework 8

Mathew Varughese, Justin Kramer, Zach Smith

Friday, Feb 15

15. a) We can store one number that represents the count. We pass through the input and every time we see a "(" we add a 1. Every time we see a ")" we subtract a 1. If the count goes below 0 we rejected. At the end of the input if the count is not 0 reject. Otherwise accept. The trick to making this a log-space Turing machine is that this count number can be stored in $\log_2 n$ bits, with n being the count of parentheses. Thus, the turing machine only has to use a logarithmic amount of bits to check an input of n parentheses.

15. b) When you begin the Turing machine, while only encountering "(" or ")", follow the lead of the previous problem and add one or subtract one from the Turing machine's count of n parentheses stored in $\log_2 n$ bits. Then, when a "[" is encountered, restart the parentheses counter. Store the index of where this "[" occurred in a value n stored in $\log_2 n$ bits. Then, when a "]" is finally encountered, check to see if the new parentheses count is equal to 0 and reject if it is not. If no "]" is encountered, reject. Once a "]" is encountered, mark its end index with a value stored in $\log_2 n$ bits. Then, loop back to the beginning of the tape. Begin the counting of the parentheses again, and when your "start" index is encountered, skip to the "end" index and continue counting the parentheses. When another "[" is encountered, set the parentheses count to reset again until the TM rejects or a "]" is encountered. If there is still no rejection, mark the "]" symbol of this bracket sequence as the new "end" index. Now loop back to the beginning. When the "start" index is found, skip to the "end" index each time. Continue to move the "end" index accordingly. If a "[" appears inside of a "[", then this "[" index becomes the new start index and the first "(" in the outer "[" becomes your new loop point. Once everything is clear within the outer "[", make it your new "start" index (if there isn't one already before it) and make your outer "]" bracket the new "end" index. If in between your "start" and "end" index there are "(" or ")" parentheses that appear outside of brackets, and their count is not set to zero when you create a new "end" index, add them to a running count of parentheses that appears when you loop back to the beginning. For example, "([()()])[()]" would start by counting to 1 in parentheses count, then set start index at the "[", then get 0 for the count inside the brackets, set the "end" index at the "]". From here, it will loop back to the start. Then, the indices will be skipped, and the next parenthesis will subtract the current running count to 0 and placed onto a new stack as -1. Then the "[" would appear and the inside will be checked. The "]" will create a new "end" index and the loop will begin from the start. The ")" that's inside of the "start" and "end" index will be put as a -1 on the new intro stack.

From the beginning, the count will start again, beginning as -1. Now it will encounter "(", increment by one, and be 0. Then it will skip the "start" and "end" indices and end in an accept state.