PS08-03

February 26, 2018

a. Union: In machine M, run x on M_1. If it rejects, reject. If it accepts, accept. If it accepts, accept. Since both languages run in polynomial time, so does their sum of steps.

b. intersection: In machine M, run x on M_1. If it rejects, reject. If it accepts, run x on M_2. If it accepts, accept.

c. concatenation. Given input x, we need to split x into two substrings, x_1 and x_2, then test if each is in each of the component languages. We start at $x_1 = \varepsilon$, and add chars to x_1 until $|x_1| = |x|$, so we end up runnning M_1 on x_1 and M_2 on x_2 $|x|$ times, resulting in an $O(n^{k+1})$ time.

d. complement. Assuming that the language in question is decidable, machine M_cmp is identical to M, except that that the accept and reject states are switched. This runs in the same time.

e. star. Construct machine M. on input w:
if $w = \varepsilon$, accept.
The strategy here is to check each substring to see if it belongs to A. This seems like it can only be done non-deterministically, but by keeping track of the substrings and their composition, this is possible to do this deterministically in polynomial time. Starting at the beginning of the string, we examine substrings. If we check both against the entire string up to our current index, as well as the substring starting from the ending of the previous valid substring, we can build substrings and, if necessary, disregard trivial substrings. If the final character receives a true marking, it is because either all of the previous previous characters make up valid substrings, all in A, or because the string w itself is in A. Accept. If the last character in w is marked as invalid, reject. This runs in polynomial time because each component runs in polynomial time, and the whole algorithm is nested in 3 loops.