Problem Set 5: Regular Expressions

Noah Buchanan
CS 4043: Formal Languages
University of Arkansas - Fort Smith
Fall 2021

October 27, 2021

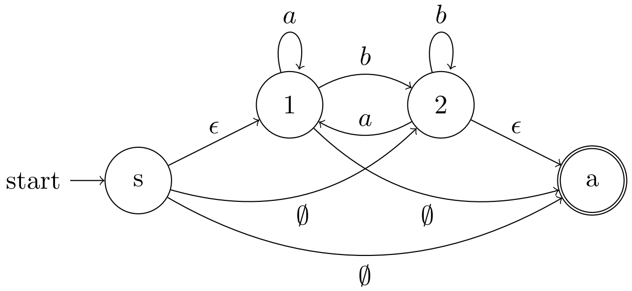
1. (a) $aa^*b(aa^*b \cup b)^*$

   (b) $(((aaa^*b \cup (ab \cup ba)a)b \cup ((ab \cup ba) \cup bba)a)a^*b \cup (((ab \cup ba) \cup bba)b \cup bbba)a)a^* \cup (((aaa^*b \cup (ab \cup ba)a)b \cup ((ab \cup ba) \cup bba)a)a^* \cup ((aaa^*b \cup (ab \cup ba)a)a^* \cup aaa^*))$

   (c) $(aa)^*aba \cup (aa)^*b$

   (d) $(aa^*bb \cup b)^*aa^*(b \cup \epsilon) \cup \epsilon$

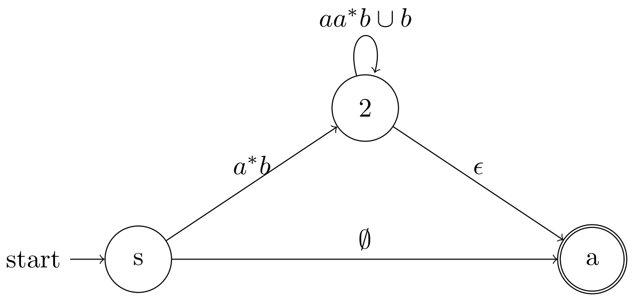
2. (a) Add new start and accept states and necessary transitions.



   (b) Determine new labels with state 1 removed:

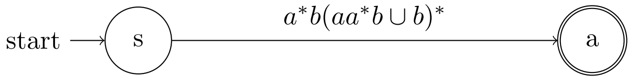
   - $s \to 1 \to 2 : (\epsilon) \circ (a)^* \circ (b) \cup (\emptyset) = a^*b$
   - $s \to 1 \to a : (\epsilon) \circ (a)^* \circ (\emptyset) \cup (\emptyset) = \emptyset$
   - $2 \to 1 \to 2 : (a) \circ (a)^* \circ (b) \cup (b) = aa^*b \cup b$
   - $2 \to 1 \to a : (a) \circ (a)^* \circ (\emptyset) \cup (\epsilon) = \epsilon$

   (c) Create new GNFA with one less state:



(d) Determine new labels with state 2 removed:
$s \to 2 \to a : (a^*b) \circ (aa^*b \cup b)^* \circ (\epsilon) \cup (\emptyset) = a^*b(aa^*b \cup b)^*$

(e) Create new GNFA with one less state:



Thus we have our final regular expression to represent the original DFA.

3. The purpose of the paper was to find a technique for automatically identifying vulnerable regular expressions. As well as crafting an "attack automaton" to create strings that will exploit the weaknesses found in vulnerable programs.

   Its actually very relevant to anyone that uses regular expressions in a production environment. If we are unaware of the possible exploitation's of the regular expressions that we push to a production environment, we run a high risk of unreasonably awful running times for our regular expressions. Whether these long running times are exploited maliciously, or purely by coincidence, would cause problems nonetheless. In the paper they claim that the web pages would become unresponsive for at least 10 minutes when exploited. Put simply, it is a security flaw that can be resolved with just a little bit of reading into how they detect weak regular expressions.

   Regular expression analysis is first performed for each regular expression in the program in which they are trying to detect a weakness in. Their NFA complexity analysis finds instances of linear, super-linear, and exponential complexity. Each regular expression's lower bound on the length of any possible attack string is determined using dynamic analysis. Attack automatons are constructed that recognize all strings that cause worse-case behavior in super-linear and exponential regular expressions. Program analysis is then performed. Merely the presence of a vulnerable regular expression does not mean the program is exploitable. Program analysis is what determines whether this is the case or not. Static analysis is performed at the source code level to determine if the program is actually vulnerable. Things such as checks to make sure input to regular expressions does not exceed a character limit, or input not coming from user input can both be scenarios where a program would contain a vulnerable regular expression, but the program itself is not.

Their overview of the process they took was a notably good part of the paper. Their explanation at a high level of what they are doing is sufficiently short but still explains enough to understand what is happening.

There is too much terminology that could be interpreted multiple ways that they never lay out exactly what they mean in a sufficient form. Another complaint I have with the paper is their extremely repetitive notations. The same symbol is used for practically everything when you get to the attack automaton portion of the paper. The only differences are hyphens, subscripts, superscripts, and other things that can be very confusing to look at and understand when there are 30 of them in a paragraph. A diverse notation of a few more symbols perhaps would have benefited the paper greatly. The format of their paper could have been clarified. The points of certain topics are scattered throughout paragraphs when they could have been enumerated for easier readability and other examples like this.