# Week 7: Loony Automata Tune

Response by: **Benjamin (aqn9yv, dlb2ru, ht6xd, iad4de, jmn4fms, lw7jz)**

**Problem 1** *NFA Size*

**Theorem 1** *The language represented by a regular expression that is $n$ characters long can be computed by a non-deterministic finite automaton of $O(n)$ states.*

*Proof.* A regular expression consisting only of the empty string can be represented by a NFA of 1 state. The language represented by a regular expression consisting of one literal character can be computed by a NFA of 2 states.

Note that the burden of proof of the following results was achieved in lecture.

Let $R_1$ and $R_2$ be regular expressions such that $R_1$ is $n_1$ characters long and computed by an NFA with $n_1 + c_1$ states and $R_2$ is $n_2$ characters long and computed by an NFA with $n_2 + c_2$ states.

The NFA which computes $(R_1|R_2)$ requires one additional state with $\varepsilon$ transitions to each of the start states for $R_1$ and $R_2$. Thus $(R_1|R_2)$ requires $n_1 + n_2 + c_1 + c_2 + 1$ states, where $(R_1|R_2)$ is $n_1 + n_2 + 1$ characters long. This can be simplified by stating the Regex $(R_1|R_2)$ which is n bits long is computed in $n + c$ states where $n = n_1 + n_2$ and c is some constant.

The concatenation of $R_1$ and $R_2$, represented by $R_1R_2$, is formed by simply adding an epsilon transition from the final states of $R_1$ to the start states of $R_2$, and thus does not require any additional states besides those used to compute $R_1$ and $R_2$. Thus for the FSA which computes the expression $R_1R_2$ of length n requires n states, where $n = n_1 + n_2$.

The Kleene star * adds one character to the regular expression, $R_1^*$, and adds a new start state to the NFA for the empty string that has an $\varepsilon$ transition to the original start state. Thus $R_1^*$ will be $n_1 + 1$ characters long, and take $n_1 + c + 1$ states to compute, where $R_1$ took $n_1 + c$ states.

Any regular expression is a combination of the empty string, literal characters, unions or Kleene stars, and as shown above any regular expression of length n composed of these elements will be computed by an FSA with $n + c$ states, where c is some constant resulting from the operations of alternation and Kleene star.

Let $R$ be a regular expression with $n$ characters, and computed by an FSA with $n + c$ states. To show that the number of states of a regular expression, $R$, is in $O(n)$ where n is the number of characters in R, we need to show that for some constant $C$, that the number of states is at most $C * n$.

From our results above we have that the number of states is $n + c$ for some constant $c$, and thus it suffices to show that $n + c \leq C * n, \ \forall n \geq n_0$ for some $n_0 \in \mathbb{N}$.

Let $n_0 = c/(C - 1)$, then for $n \geq n_0$, $n \geq c/(C - 1)$
$\implies (C - 1) \cdot n \geq c$
$\implies C \cdot n - n \geq c$
$\implies C \cdot n \geq n + c \iff n + c \leq Cn$
$\implies n + c \in O(n)$
$\implies$ The number of states required to compute a regular expression is in $O(n)$. □