Properties of Regular Languages

Ihab M. E.

July 14, 2017

A very important fact about regular languages is called a "closure property". There are some properties that help us decide and answer important questions about automata. A central example is an algorithm for deciding whether two automata define the same language. As a consequence of our ability to decide that two automata are in fact equivalent helps us "minimize" automata, that is to get an automaton with as few states as possible.

1 Proving Languages not to be Regular

Regular languages, thus far, can be defined by DFA's, $\epsilon - NFA$'s, NFA's, and regular expressions.

1.1 The Pumping Lemma

First, take as an example the regular expression informally defined as follows:

$$L_{01} = \{0^n 1^n | n \geq 1\}$$

which is not a regular expression, thus it can't be represented by a DFA, or any of its sister automata, $\epsilon - NFA$ and NFA's, because a DFA simply can't remember.

Theorem 1.1.1: **(The Pumping Lemma).** Let L be a regular language. Then there exists a constant n (which is the number of states of the DFA representing the language) such that for each string w in L where $|w| \geq n$, we can break w into three strings, $w = xyz$ such that

1. $y \neq \epsilon$
2. $|xy| \leq n$.
3. For all $k \geq 0$, the string xy^kz is in L.

That is, we can always find a nonempty string y not too far from the beginning of w that can be "pumped"; that is, to be repeated any number fo times, or even deleting it (k=0), keeps the resulting string in the language L.

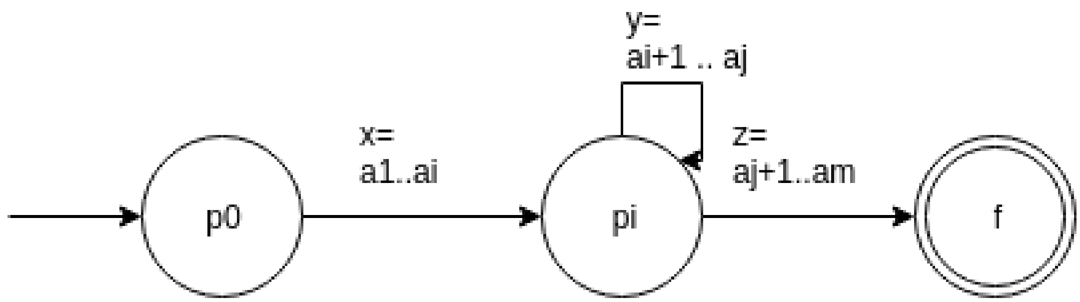
PROOF:

Suppose L is regular. Then L=L(A), for some DFA A. Suppose A has n states. Now, consider any string w of length n or more, say $w = a_1 a_2 \cdots a_m$ where $m \geq n$. For i $= 0, 1, \cdots, n$ define state p_i to be $\delta(q_0, a_1 a_2 \cdots a_i)$, where, of course δ is the transition function of the automaton A, and of course $p_0 = q_0$ (where $q_0$is the initial state of the automaton).

According to the pigeonhole principle, if we have n states, and an input of length m, then at least one of the states was visited more than once.

We can then break $w = xyz$ as follows:

1. $x = a_1 a_2 \cdots a_i$
2. $y = a_{i+1} a_{i+2} \cdots a_j$
3. $z = a_{j+1} a_{j+2} \cdots a_m$



Now, consider what happens if the automaton A receives $xy^k z$ for some $k \geq 0$. For k=0, the automaton A goes from $q_0$to p_i on input x, and then it goes from p_i to f on input z.

If k>0, then A goes from $q_0 (or\, p_0)$ to p_i on input x, and from $p_i$to itself on input y, and finally from p_i to f on input z.

And thus in both cases, our strings are actually in L.