Formal Language & Automata Theory

Prof. Sankhadeep Chatterjee

Properties of Regular Languages

- Which languages are not regular?
- Decision properties
  - A decision property for a class of languages is an algorithm that takes a formal description of a language (e.g., a DFA) and tells whether or not some property holds.
    - Is a regular language empty?
    - Is a string w belong to the language?
    - Do different descriptions languages describe same language?

Closure properties

- A *closure property* of a language class says that given languages in the class, an operator (e.g., union) produces another language in the same class.
  - The union of two regular languages is regular
  - The intersection of two regular languages is regular
  - The complement of a regular language is regular
  - The difference of two regular languages is regular
  - The reversal of a regular language is regular
  - The closure (star) of a regular language is regular
  - The concatenation of regular languages is regular
  - A homomorphism (substitution of strings for symbols) of a regular language is regular
  - The inverse homomorphism of a regular language is regular

Pumping Lemma for Regular Languages

For every regular language L

Number of states of DFA for L

There is an integer n, such that

For every string w in L of length $\geq$ n

We can write w = xyz such that:

Labels along first cycle on path labeled w

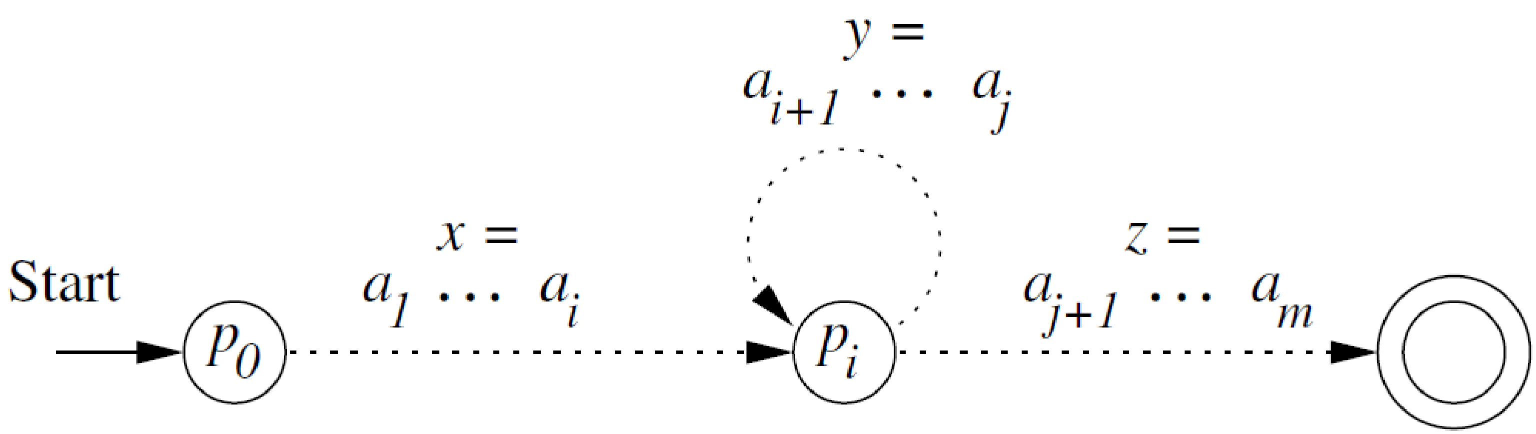
1. $|xy| \leq n$.
2. $|y| > 0$.
3. For all $i \geq 0$, xy^iz is in L.

Proof: Pumping Lemma for Regular Languages

- *Suppose L is regular. Then L = L(A) for some DFA A with* n states.
- Now, consider any string w of length n or more, say $w = a_l a_2 \ldots a_m$, where $m > n$ and each a_i is an input symbol
- *For $i = 0, 1, \ldots, n$ define state p_i to be $\hat{\delta}(q_O, a_l a_2 \ldots a_i)$, where $\hat{\delta}$ is the transition function of A, and q_O is the start state of A.*
- *That is, p_i is the state A is in after reading the first i symbols* of w.

Proof: Pumping Lemma for Regular Languages

- *By the pigeonhole principle, it is not possible for the n + 1 different p_i's for* i = 0, 1,... , n to be distinct, since there are only n different states.
- Thus, we can find two different integers i and j, with $0 \leq i < j \leq n$, such that $p_i = p_j$.
- *We can break w = xyz as follows:*
  - $x = a_1 a_2 \ldots a_i$.
  - $y = a_{i+1} a_{i+2} \ldots a_j$.
  - $z = a_{j+1} a_{j+2} \ldots a_m$



Proof: Pumping Lemma for Regular Languages

- *Now, consider input $xy^k z$ for any $k > 0$.*
- *If $k = 0$,*
  - *then the automaton goes from the start state q_O (which is also p_O) to p_i on input x.*
  - *Since p_i is also p_j, it must be that A goes from p_i to the accepting state on input z. Thus, A accepts xz.*
- *If $k > 0$,*
  - *then A goes from q_O to p_i on input x,*
  - *circles from p_i to p_i , k times* on input y^k,
  - then goes to the accepting state on input z.
- Thus, for any $k > 0$, $xy^k z$ is also accepted by A; that is, $xy^k z$ is in L.

Pumping lemma contd.

- We use pumping lemma to show that a given language is not regular

- If the conditions stated in pumping lemma is not satisfied it can not be concluded that the language is regular or not

- The trick is to find a suitable value of ‘k’ as mentioned in xy^kz and show that the string is not in L

Example: Pumping Lemma

- Show that L = {w | w is palindrome} over $\Sigma = \{0,1\}$ is not Regular
- Suppose, L is regular. Then there exists a constant 'n' such that it satisfies all conditions of Pumping Lemma.
- Let w = 0^n10^n (Note: $|w| \geq n$)
- Using pumping lemma w is broken as xyz such $y \neq \epsilon$ and $|xy| \leq n$
- Hence, both x and y contains only 0's
- Assume that, $x = 0^i$ and $y = 0^j$, thus z = 10^n
- Now, by pumping lemma, $xy^0z = 0^i\epsilon10^n = 0^{n-j}10^n \in L$
- However, $0^{n-j}10^n \notin L$
- Hence, our assumption was incorrect. L is not regular

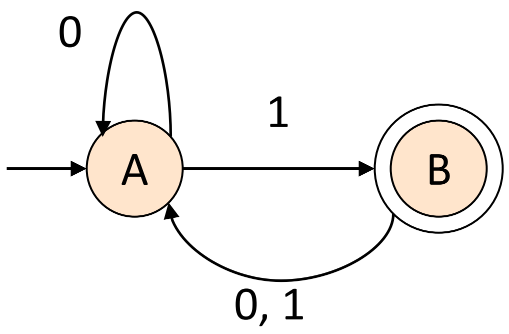
Is a regular language empty?

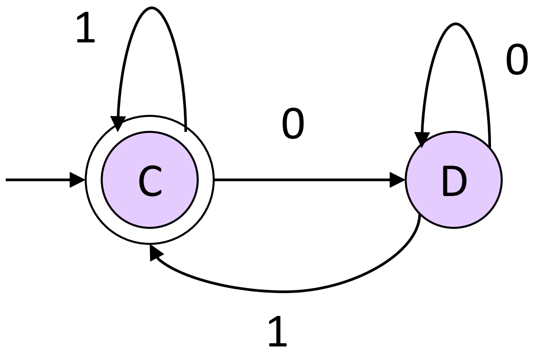
- Given a regular language, does the language contain any string at all.
- Assume representation is DFA.
- Construct the transition graph.
- Compute the set of states reachable from the start state.
- If any final state is reachable, then yes, else no.

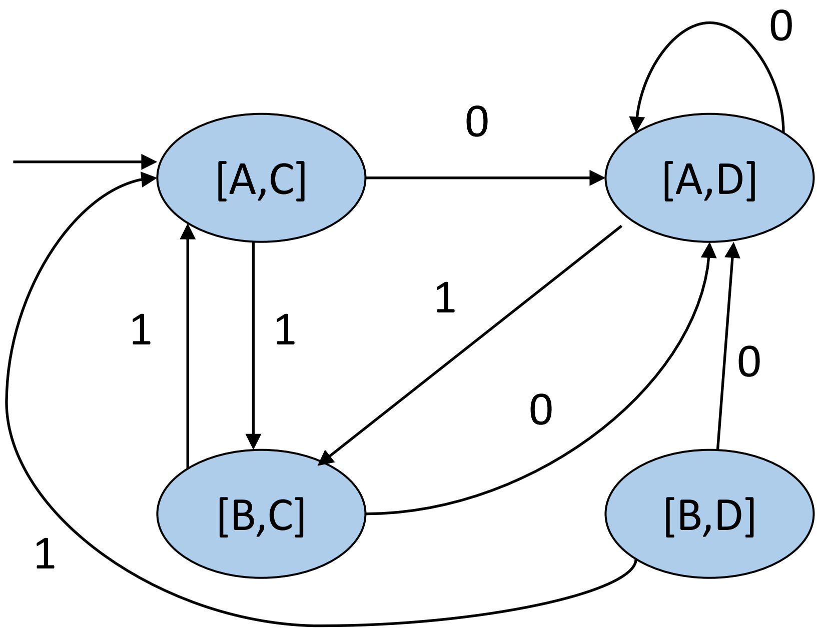
Given regular languages L and M, is L = M?

- Construct the *product DFA* from DFA's for L and M.
- Let these DFA's have sets of states Q and R, respectively.
- Product DFA has set of states $Q \times R$.
  - i.e., pairs [q, r] with q in Q, r in R.
- Start state = $[q_0, r_0]$ (the start states of the DFA's for L, M).
- Transitions: $\delta([q,r], a) = [\delta_L(q,a), \delta_M(r,a)]$
  - δ_L, δ_M are the transition functions for the DFA's of L, M.
  - That is, we simulate the two DFA's in the two state components of the product DFA.

Example: Product DFA







Given regular languages L and M, is L = M?

- Make the final states of the product DFA be those states [q, r] such that exactly one of q and r is a final state of its own DFA.
- Thus, the product accepts w iff w is in exactly one of L and M.
- The product DFA's language is empty iff L = M.
- Use Emptiness checking algorithm to test whether the language of a DFA is empty.