CS 208 : Automata Theory and Logic 20th Feb 2024

Lecture - 20

Topic: Operations on Automata

Scribed by: Lakshya Gadhwal (22b0995)
Checked and compiled by:

1 DFA definition

Any DFA can be completely represented by the tuple DFA $= (Q, \Sigma, q_0, \delta, F)$ where,

- Q is the set of all states
- Σ is the alphabet
- q_0 is the starting state
- $\delta : (Q \times \Sigma) \to Q$ is the transition function
- F is the set of final or accepting states

1.1 Combinations of DFAs

We can represent the combination of two DFAs defined on the same alphabet Σ, $\text{DFA}_1 = (Q, \Sigma, q_0, \delta, F)$ and $\text{DFA}_2 = (Q', \Sigma, q_0', \delta', F')$ by another DFA,

$$\text{DFA}_3 = \left(Q \times Q', \Sigma, \left(q_0, q_0'\right), \hat{\delta}, \hat{F}\right) \tag{1}$$

where $\hat{\delta}\left((q, r), a\right) = (\delta(q, a), \delta'(r, a))$ and $\hat{F}$ can be defined according to the required operation on the DFAs.

For example, if $Q = \{q_0, q_1\}$ and $Q' = \{r_0, r_1, r_2\}$ and $F = \{q_1\}$ and $F' = \{r_1, r_2\}$ then for the **intersection** of these Automata, $\hat{F} = \{(q_1, r_1), (q_1, r_2)\}$ and for the **union** of the Automata, $\hat{F} = \{(q_1, r_0), (q_1, r_1), (q_1, r_2), (q_0, r_1), (q_0, r_2)\}$. Similarly we can define the **complement** operation by taking $\hat{F} = Q - F$.

With these basic rules in place, we can go on to define more complicated combinations like $(DFA_1 \cap DFA_3) \cup (DFA_2 \cap DFA_3) - (DFA_1 \cap DFA_2)$

We now have another way to tell if a language is a subset of another language. To tell if $L_1 \subseteq L_2$, we have to show that $L_1 \cap L_2^{\complement} = \phi$. The problem thus reduces to showing that in the DFA defined by $DFA_1 \cap DFA_2^{\complement}$ there is no path from the start node to any accepting state.

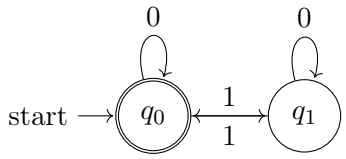


Figure 1: Two-State Automaton

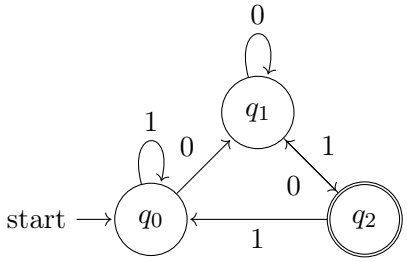


Figure 2: Three-State Automaton

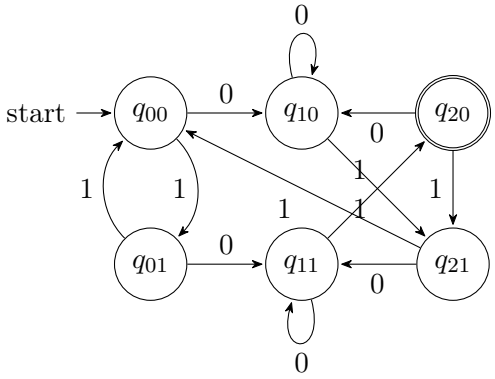


Figure 3: Intersection of the two automata

1.2 Combinations of NFAs

For two NFAs, the union and intersection is defined in a similar way with the only difference being the transition function since it now returns a set of states instead of a single state.

$$\hat{\delta}\left((q,r),a\right) = \delta(q,a) \times \delta'(r,a) \tag{2}$$

However, for NFAs, just flipping the accepting and non-accepting states won't give us the complement.
Thus to take the intersection or union of two NFAs, we can follow a similar approach to DFAs but for complementation, the NFA must first be converted to a DFA and then complemented to give the actual complement of the original NFA.

1.3 Closure Properties

Given L_1, L_2 are two regular languages over the alphabet $\Sigma = \{a, b\}$, $\overline{L_1}$, $L_1 \cup L_2$, $L_1 \cap L_2$ are also regular languages (because their corresponding DFAs can be constructed by the combination of the original DFAs).

2 Substitution

We will start with an example.

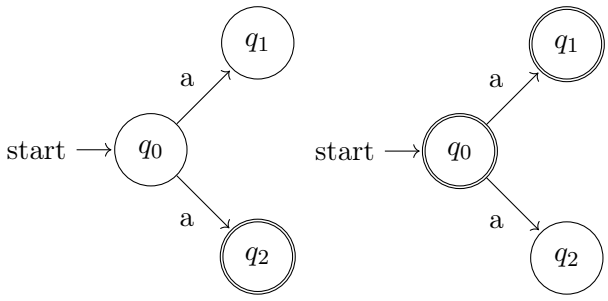


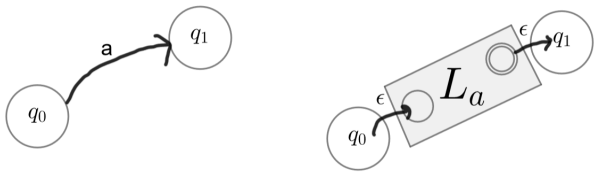
Figure 4: These NFAs are not complements of each other

Consider two alphabets $\Sigma_1 = \{a, b\}$ and $\Sigma_2 = \{0, 1, 2\}$ and languages $L_1 = a^*b^*$ defined on Σ_1^* and $L_a = 0^*(1+2)^*1^*$ and $L_b = 1^*(0+2)^*$ defined on Σ_2^*.

Now we define a set subst(L_1, L_a, L_b) as

$$\text{subst}(L_1, L_a, L_b) = \{w \in \Sigma_2^* \mid \exists u \in L_1 = \alpha_1\alpha_2 \dots \alpha_k \text{ such that } w \in L_{\alpha_1}L_{\alpha_2} \dots L_{\alpha_k}\} \quad (3)$$

Or equivalently, subst$(L_1, L_a, L_b) = \bigcup_{u=\alpha_1\alpha_2\dots\alpha_k \in L_1} L_{\alpha_1}L_{\alpha_2} \dots L_{\alpha_k}$
Intuitively, the substitution operation is to replace each letter by a language.
Diagrammatically, it is represented as replacing each edge by an entire automaton and connecting the initial and accepting states to the original states by ϵ edges.



Using this operation, we can easily prove that if L_1 and L_2 are regular languages then $L_1 \cdot L_2$ is also regular, since $L = \{a \cdot b\}$ is regular then subst(L,L_1,L_2) will also be regular.

2.1 Infinite Languages

If we have a DFA with n states and there exists some string of length greater than n which is accepted by the DFA, then by the pigeon-hole principle there must exist some state q in the DFA such that there is a cycle with q in it. Suppose that the accepting string is $u \cdot v \cdot w$, where v is the string that starts and ends at the same state, then the strings $u \cdot w$, $u \cdot v^2 \cdot w$, $u \cdot v^* \cdot w$ are also accepted in the DFA. Hence the language formed by the string is of infinite length.