CS 361: Lecture 3 Handout

Shikha Singh

1 Closure Properties of Regular Languages

We say that a set is closed under some unary operation if by applying this operation to elements of the given set, the result is an element also contained in that set. Closure under binary operations is defined analogously.

For example, consider the set of natural numbers $\mathbb{N}$, and consider the binary operations addition and division. The natural numbers are *closed under addition*, because if $a, b \in \mathbb{N}$, then $a + b \in \mathbb{N}$. On the other hand, natural numbers are NOT *closed under division* since if we take $1, 2 \in \mathbb{N}$, then $\frac{1}{2} \notin \mathbb{N}$.

When talking about regular languages, the same principle holds. We next discuss for which operations the set of regular languages are closed under.

Theorem 1. Regular languages are closed under complement.

Proof. First note that the complement of a language is the complement with respect to some alphabet. So if Σ is the alphabet of language L, then $\overline{L} = \Sigma^* - L$, i.e., all the strings over the alphabet Σ that are not in L.

Now let L be an arbitrary regular language. We want to show that $\overline{L}$ is also a regular language, that is, we need to show that there exists a finite automaton that recognizes $\overline{L}$.

By the definition of a regular language, there exists a finite automaton $M = (Q, \Sigma, q_0, \delta, F)$ such that $L = L(M)$. Consider the automaton defined by $\overline{M} = (Q, \Sigma, q_0, \delta, Q - F)$. The new automaton $\overline{M}$ is almost the same as M, differing only on the set of accept states.

We claim that $L(\overline{M}) = \overline{L}$. We prove this by showing that $\overline{L} \subseteq L(\overline{M})$ and $L(\overline{M}) \subseteq \overline{L}$.

Consider an input string $w \in \overline{L}$. As $w \in \overline{L}$, we know that $w \notin L$. Since $L(M) = L$, we know that M does not accept w. That is, there exists a computation of M starting in q_0 and ending in some state $q \in Q - F$. Since $\overline{M}$ has the same states and transition function as M, but with final states $Q - F$, we can conclude that $\overline{M}$ accepts w. Thus, $\overline{L} \subseteq L(\overline{M})$.

Now consider an input string $w \in L(\overline{M})$, that is, $\overline{M}$ accepts w. Then there exists a computation of $\overline{M}$ starting in q_0 and ending in some state $q \in Q - F$. Since M has the same states and transition function as $\overline{M}$, and $q \notin F$, we can conclude that M does not accept w, or $w \notin L(M) = L$ which means that $w \in \overline{L}$. Thus, $L(\overline{M}) \subseteq \overline{L}$. □

Theorem 2. Regular languages are closed under intersection and union.

Proof. We want to show that if L_1, L_2 are regular languages, then so are $L_1 \cap L_2$ and $L_1 \cup L_2$. By definition, we know there are finite automata $M_1 = (Q, \Sigma, \delta_1, q_0, F_1)$ and $M_2 = (S, \Sigma, \delta_2, s_0, F_2)$ such that $L_1 = L(M_1)$ and $L_2 = L(M_2)$. Note that we are assuming that

both languages have the same alphabet because if they were different, then we could consider Σ as their union.

To show that $L_1 \cap L_2$ is a regular language, we want to define an automaton that will simulate both M_1 and M_2 in parallel, and accept a string w if both automata accept w. To show that $L_1 \cup L_2$ is a regular language, we similarly want to define an automaton that will simulate both M_1 and M_2 in parallel and accept a string w if either automata accept w.

The states of the new machines Q will thus be pairs of states from M_1 and M_2, more precisely $Q = Q \times S$. The alphabet remains the same and we consider the start state as the pair (q_0, s_0), since both simulations must start in their respective start state.

The transition function δ of the new machines are defined considering what M_1 and M_2 would do. Given a pair of states (q, s) and a symbol $a \in \Sigma$, the transitions in the new machine should follow what M_1 does with a from state q (that is, $\delta_1(q, a)$), and what M_2 does with a and state s (that is, $\delta_2(s, a)$). Formally, $\delta((q, s), a) = (\delta_1(q, a), \delta_2(s, a))$.

Finally, the set of accept states of the new machines are defined by the pair of accept states in M_1 and M_2. We consider the final states for intersection and union separately.

Intersection. For the language $L_\cap = L_1 \cap L_2$, we define the finite automata that recognizes $L_\cap$ as $M_\cap = (Q \times S, \Sigma, \delta, (q_0, s_0), F_\cap)$, where $F_\cap = \{(q_f, s_f) \mid q_f \in F_1 \text{ and } s_f \in F_2\}$.

To prove that $L(M_\cap) = L_\cap$, we again have two cases.

First, consider a string $w \in L_\cap = L_1 \cap L_2$. Since $w \in L_1 = L(M_1)$ and $w \in L_2 = L(M_2)$, we know there exists computations in M_1 and M_2 that start in states q_0, s_0 respectively and end in accept states q_f, s_f respectively where $q_f \in F_1$ and $s_f \in F_2$. By construction of $M_\cap$, we know there is a corresponding computation of $M_\cap$ on w that starts in (q_0, s_0), follows the transitions defined by δ and ends in $(q_f, s_f) \in F_\cap$. This means that $M_\cap$ accepts w, that is, $w \in L(M_\cap)$. Thus, $L_\cap \subseteq L(M_\cap)$.

Second, consider a string w that is accepted by $M_\cap$. There must be a computation of $M_\cap$ that starts in (q_0, s_0) and ends in $(q_f, s_f) \in F_\cap$. This computation is a sequence of pairs of states that agree with δ, so if we consider a sequence formed by the first argument of each pair and another sequence formed by the second argument of each pair, we obtain two computations, one for M_1 and one for M_2. Since each computation ends in a accept state, we know that both M_1 and M_2 accept w, so $w \in L(M_1) \cap L(M_2) = L_1 \cap L_2$. Thus, $L(M_\cap) \subseteq L_\cap$.

Together we have proved that $L_\cap = L(M_\cap)$, that is, $M_\cap$ correctly recognizes the language $L_1 \cap L_2$ and thus it is a regular language.

Union. To show that $L_1 \cup L_2$ is also regular, we use almost the same construction as before, differing only on the set of final states. For the language $L_\cup = L_1 \cup L_2$, we define the finite automata that recognizes $L_\cup$ as $M_\cup = (Q \times S, \Sigma, \delta, (q_0, s_0), F_\cup)$, where $F_\cup = \{(q_f, s_f) \mid q_f \in F_1 \text{ or } s_f \in F_2\}$. The final states are defined using an "or" because for a string to be in the union we just need that at least one of the machines M_1 or M_2 accept it.

The proof that $L_\cup = L(M_\cup)$ is similar to the case of intersection. $\square$