ECS 120 Lesson 4 – Closure Properties of Regular Languages, Pt. 1

Oliver Kreylos

Friday, April 6th, 2001

1 Operations on Languages

We defined a (formal) language L over an alphabet Σ as a set of words: $L \subset \Sigma^*$. Today we introduce operations on languages, and how the class of regular languages behaves under such operations. Since languages are sets of words, the first operations we look at are the set operations complement, intersection and union.

- Let $A \subset \Sigma^*$ be a language. Then we define the *complement* $\bar{A}$ of A as $\bar{A} := \left\{ w \in \Sigma^* \;\middle|\; w \notin A \right\}$, or, equivalently, $\bar{A} = \Sigma^* \setminus A$.
- Let $A, B \subset \Sigma^*$ be languages over the same alphabet. Then we define the *intersection* $A \cap B$ of A and B as $A \cap B := \left\{ w \in \Sigma^* \;\middle|\; w \in A \wedge w \in B \right\}$, the usual set intersection; and we define the *union* $A \cup B$ of A and B as $A \cup B := \left\{ w \in \Sigma^* \;\middle|\; w \in A \vee w \in B \right\}$, the usual set union.

The next two operations are specific to sets of words: Concatenation and Kleene Star. Before we introduce them, we have to formally define the concatenation of two words. Let $x_1, x_2 \in \Sigma^*$ be two words over the same alphabet. Then we define the *concatenation* $x_1 x_2$ of x_1 and x_2 recursively as follows:

1. If $x_1 \in \Sigma^0$, i. e., $x_1 = \epsilon$, then we define $x_1 x_2 := x_2$.
2. If $x_1 \in \Sigma^* \setminus \{\epsilon\}$, i. e., x_1 is not the empty word, we can split x_1 into a character $a \in \Sigma$ and a word $x_1' \in \Sigma^*$: $x_1 = a x_1'$. Then we define $x_1 x_2 = (a x_1') x_2 := a(x_1' x_2)$.

Less formally, two words are concatenated by writing their characters as a single word: If $x_1 = a_1 a_2 \dots a_n$ and $x_2 = b_1 b_2 \dots b_m$, then $x_1 x_2 = a_1 a_2 \dots a_n b_1 b_2 \dots b_n$.

From the formal definition of concatenation, we can derive its following two properties:

- Associativity: If $a, b, c \in \Sigma^*$ are words over the same alphabet, then $a(bc) = (ab)c$. In other words, concatenating a with the result of concatenating b and c is equivalent to first concatenating a and b and concatenating the result with c.
- Identity element: If $a \in \Sigma^*$ is a word, then $a = \epsilon a = a\epsilon$. In other words, the empty word does not change any word it is concatenated with: It is the *identity element* of the concatenation operation.

Now we can define two more operations on languages:

- Let $A, B \subset \Sigma^*$ be languages over the same alphabet. Then we define the *concatenation* AB of A and B as $AB := \left\{ ab \;\middle|\; a \in A \wedge b \in B \right\}$. In other words, the set AB is formed by concatenating every word in A with every word in B.
- Let $A \subset \Sigma^*$ be a language. Then we define the sets A^n recursively for all $n \geq 0$:
  - $A^0 := \{\epsilon\}$
  - $A^{n+1} := A^n A$

  In other words, A^n is the set of all words formed by taking any sequence $a_1, a_2, \dots, a_n \in A$ of n words from A and concatenating them. Finally, we define the *Kleene Star* A^* of A as $A^* := \bigcup_{n \geq 0} A^n$.

2 Closure of Regular Languages

We will now ask the question what happens if we apply these operations to a regular language, i. e., a language accepted by a finite automaton. The claim is that applying any of these operations to a regular language creates another regular language; in other words, the class of regular languages is *closed* under these operations. To prove this claim, we have to start with one (or two) regular languages, apply one of the operation, and prove that

the resulting language is again regular. Since our definition is that a regular language is one accepted by a finite automaton, we can prove the resulting language being regular by designing an automaton that accepts it.

2.1 Complement

Let $A \subset \Sigma^*$ be a regular language, and $\bar{A} \subset \Sigma^*$ its complement. We will now construct an automaton that accepts $\bar{A}$. First we note that, since A is regular, there must be an automaton $M = (Q, \Sigma, \delta, q_0, F)$ that accepts A, i. e., $A = L(M)$. Since $\bar{A}$ contains exactly those words that are not in A, the new automaton $\bar{M} = (\bar{Q}, \Sigma, \bar{\delta}, \bar{q}_0, \bar{F})$ must reject all words in A and accept all words not in A. Let us recall the definition of acceptance: M accepts a word w, iff $\delta^*(q_0, w) \in F$. Since $\bar{M}$ must reject the same word w, we have $\bar{\delta}^*(\bar{q}_0, w) \notin \bar{F}$. The idea now is to define $\bar{M}$ to be identical to M, but having accepting and rejecting states reversed. Thus, we define $\bar{M} := (Q, \Sigma, \delta, q_0, \bar{F})$ where $\bar{F} := \left\{ q \in Q \;\middle|\; q \notin F \right\} = Q \setminus F$. Now let us prove that $\bar{M}$ accepts exactly $\bar{A}$ by looking at two cases:

1. Let $w \in A$. Then M accepts w, i. e., $\delta^*(q_0, w) \in F$. This means that $\delta^*(q_0, w) \notin \bar{F}$, which means that $\bar{M}$ correctly rejects w.

2. Let $w \notin A$. Then M rejects w, i. e., $\delta^*(q_0, w) \notin F$. This means that $\delta^*(q_0, w) \in \bar{F}$, which means that $\bar{M}$ correctly accepts w.

2.2 Intersection

This proof follows the same idea: Since A and B are regular languages, they are accepted by two machines $M_A = (Q_A, \Sigma, \delta_A, s_A, F_A)$ and $M_B = (Q_B, \Sigma, \delta_B, s_B, F_B)$, respectively. Based on those two, we construct a new machine M and prove that it accepts $A \cap B$. This proof is a little more complicated because it incorporates two machines. To decide whether a word w is in $A \cap B$, we have to run both machines in parallel, and accept w iff it is accepted by both at the same time. Here is the construction to run two automata in parallel: We define $M := (Q_A \times Q_B, \Sigma, \delta, (s_A, s_B), F_A \times F_B)$, where $\delta\big((q_A, q_B), a\big) := \big(\delta_A(q_A, a), \delta_B(q_B, a)\big)$. Here is an informal description:

- The set of states of M is the cartesian product of the sets of states of M_A and M_B, i. e., the set of all pairs where the first element is a state from M_A and the second element is a state from M_B.

- The transition function of M applies the transition functions of M_A and M_B independently to the two elements of each state. The idea is to simulate the computation of M_A in the first element of the states, and to simulate the computation of M_B in the second element.
- The start state of M is the pair consisting of the start states of M_A and M_B.
- The final states of M are all pairs where both elements are final states in their respective machines.

To prove that the construction for machine M works, i. e., that $L(M) = A \cap B$, we have to construct its language $L(M)$: Let $w \in \Sigma^*$ be any word. M accepts w iff $\delta^*\big((s_A, s_B), w\big) \in F_A \times F_B$. By definition of δ, we find that for any state $(q_A, q_B) \in Q_A \times Q_B$ and any word $w \in \Sigma^* : \delta^*\big((q_A, q_B), w\big) = \big(\delta_A^*(q_A, w), \delta_B^*(q_B, w)\big)$. Thus, $\delta^*\big((s_A, s_B), w\big) = \big(\delta_A^*(s_A, w), \delta_B^*(s_B, w)\big)$, and $\delta^*\big((s_A, s_B), w\big) \in F_A \times F_B$ iff $\delta_A^*(s_A, w) \in F_A$ and $\delta_B^*(s_B, w) \in F_B$. Therefore, M accepts a word w exactly if both M_A and M_B accept w, meaning that $w \in A \cap B$.

2.3 Union

The proof that $A \cup B$ is also a regular language works in exactly the same way as the proof for $A \cap B$. Again, we build a machine M that runs two other machines M_A and M_B in parallel, and accepts a word based on their acceptance. The only difference is the construction of the M's final states: For the intersection operation, a state $(q_A, q_B) \in Q_A \times Q_B$ was a final state iff both q_A and q_B were final states in their respective machines; making M accept a word if it is accepted by both machines M_A and M_B. Now, we define the final states of M to be the states $(q_A, q_B) \in Q_A \times Q_B$ where at least one of q_A and q_B is a final state in its respective machine: $F := \big\{ (q_A, q_B) \;\big|\; q_A \in F_A \vee q_B \in F_B \big\} = (F_A \times Q_B) \cup (Q_A \times F_B)$. This way, M will accept all those words which are accepted by M_A or M_B and are thus in the union of the languages A and B, $A \cup B$.