Lecture 6

Additional properties regular languages

In this lecture, we'll go through some further properties of regular languages and try to get a feel for how to prove things about regular languages.

6.1 Symmetric difference

The symmetric difference between two sets A and B is defined as the set of all elements that are in one of A or B, but not both. This is denoted by $A \triangle B$. For example, we have $\{a, b, c\} \triangle \{b, c, d\} = \{a, d\}$. The symmetric difference gives us the set of elements on which the two sets differ.

Are regular languages closed under symmetric difference? Well, we have $A \triangle B = (A \backslash B) \cup (B \backslash A)$, and since we've seen that regular languages are closed under set difference and under unions, it follows that regular languages are also closed under symmetric difference.

One cool property of the symmetric difference is that if $C = A \triangle B$, then $A = B \triangle C$. Let's prove this. Fix $x \in A$; we want to show that x is in $B \triangle C$. There are two options: if $x \in B$, then by the definition of $C = A \triangle B$, we know that $x \notin C$, since x is in both A and B; therefore, $x \in B \triangle C$. Alternatively, if $x \notin B$, then by definition we have $x \in C = A \triangle B$, so x is again in one of B and C, and therefore $x \in B \triangle C$. Conversely, fix $y \in B \triangle C$; we want to show that y is in A. There are two options: if y is in B but not C, then since $C = A \triangle B$, it must be the case that $y \in A$. Similarly, if y is in C but not B, then since $C = A \triangle B$, it must be the case that $y \in A$. We've now shown that all elements in A are in $B \triangle C$, and also that all elements in $B \triangle C$ are in A, so $A = B \triangle C$.

(It might help to think of the symmetric difference as similar to the XOR logical operation. Recall that the set $A \cap B$ is formed by taking all the elements in A AND B, while the set $A \cup B$ is formed by taking all the elements in A OR B. In a similar way, the symmetric difference $A \triangle B$ is formed by taking all the elements in A XOR B, where XOR is the exclusive-OR.)

Here is an interesting conclusion: if A is regular and $A \triangle B$ is regular, then B is regular. This is because we've seen that $B = A \triangle (A \triangle B)$, and regular languages are closed under $\triangle$.

Note that this property is not true for intersection and union: if A is regular and $A \cup B$ is regular, it might still be the case that B is irregular (for example, take $A = \{0, 1\}^*$ and $B = \{0^n 1^n : n \in \mathbb{N}\}$; in this case, $A \cup B = A$, which is regular, but B is not regular). Similarly, if A is regular and $A \cap B$ is regular, it might still be the case that B is irregular (for example, take $A = \varnothing$ and $B = \{0^n 1^n : n \in \mathbb{N}\}$; then $A \cap B = A$, which is regular, but B is not regular).

One particularly useful conclusion is that if A is regular and $A \triangle B$ is finite, then B is regular. In other words, if a language differs from a regular language on finitely many strings, it must still be regular. This follows from the fact that finite languages are regular, so if $A \triangle B$ is finite, it is regular. Similarly, if A is not regular and $A \triangle B$ is finite, then B is also not regular. (That's because if B were regular, we would conclude that A is regular by the aforementioned property.)

To see why this is useful, consider the following example. Suppose we wanted to show that the language $\{0^n1^n : n \geq 3\}$ is not regular. We could use the pumping lemma from scratch, of course. But we could also observe that this language differs from $\{0^n1^n : n \in \mathbb{N}\}$ on only finitely many strings: the symmetric difference between the two languages is finite. Since we've already shown the latter is irregular, the former must also be irregular.

6.2 Reverse

Next, let's show that the regular languages are closed under the reverse operation. Recall that the reverse of a language, denoted A^R, is the language $\{x^R : x \in A\}$, where x^R denotes the reverse of a string x (that is, if $x = x_1x_2 \dots x_n$ where $x_i \in \Sigma$ are symbols, then $x^R = x_nx_{n-1} \dots x_1$). We want to show that if A is regular, then A^R is also regular.

One way to do so will be to use the regular expression characterization of regular languages. Let S be any regular expression; we will define the reverse of S to be another regular expression, defined as follows.

1. If $S = \varnothing$, then $S^R = \varnothing$.
2. If $S = \epsilon$, then $S^R = \epsilon$.
3. If $S = c$ for some $c \in \Sigma$, then $S^R = c$.
4. If $S = (S_1 \cup S_2)$ for some regular expressions S_1 and S_2, then $S^R = (S_1^R \cup S_2^R)$.
5. If $S = (S_1S_2)$ for some regular expressions S_1 and S_2, then $S^R = (S_2^R S_1^R)$.
6. If $S = (S_1^*)$ for some regular expression S_1, then $S^R = ((S_1^R)^*)$.

Note that this definition covers all possible cases of what a regular expression S could be, given how we defined regular expressions. Thus for each regular expression S, the regular expression S^R is well-defined.

It is not hard to see that $L(S^R) = L(S)^R$. Now, if A is any regular language, then it must have some regular expression S such that $L(S) = A$; then $L(S^R) = A^R$, so S^R is a regular expression for A^R, and hence A^R is regular. This shows that the regular languages are closed under reverse.

Can we also prove this closure property using the DFA/NFA characterizations of regular languages? Given a regular language A, we want to show that A^R is regular. One useful approach is to assume we have a DFA for the given regular language (in this case A) and to construct an NFA for the target language (in this case A^R). Assuming we have a DFA for the given language is helpful as DFAs are easier to analyze than NFAs, and constructing an NFA for the target language is helpful because NFAs are easier to construct than DFAs.

In this case, if we are given a DFA M for A, how do we construct an NFA N for A^R? Well, we want N to accept a string x if and only if M accepts x^R; that is, we want there to be an accepting path for x in N if and only if M reaches an accept state when run on x^R. To do this, we will let N have the same states as M, but with all the arcs reversed. We will also add a new start state q_0 in N, and we'll add ϵ-transitions from the new start states to all of the accept states of M. Finally, we will set the start state of M to be the only accept state of N.

This way, if there is an accepting path for x in N, it means that there is some path in N leading from an accept state of M to the start state of M; the same path with the arcs reversed would lead from the start state of M to an accept state of M when run on x^R, and hence if N accepts x then

M accepts x^R. Conversely, if M accepts x^R, then there is a path from the start state of M to an accept state of M that reads x^R, and reversing it gives us a path in N from an accept state of M to the start state of M; adding an ϵ-transition at the beginning, we get that N accepts x, as desired.

6.3 Prefix, suffix, and substring

Recall that for a language A, the prefix language, denoted Prefix(A), is the set of all prefixes of strings in A:

$$\text{Prefix}(A) = \{x \in \Sigma^* : xy \in A \text{ for some } y \in \Sigma^*\}.$$

Similarly, Suffix(A) is the set of all suffixes of strings in A, and Substring(A) is the set of all substrings of strings in A. We will now show that regular languages are closed under all of these operations.

First, consider Suffix(A). Suppose A is regular. How do we show that Suffix(A) is regular?

Often, the best way to tackle such problems is to assume we have a DFA for the given language A, and to construct an NFA for the larget language (in this case Suffix(A)). Let's try this.

Let M be a DFA for A. We want to construct an NFA N that recognizes Suffix(A). To do so, let's first consider the set P of all states that are reachable from the start state of M via some path. Of course, when constructing a DFA, we would never want to have unreachable states, since they never do anything; but nothing in the definition of a DFA prevents unreachable states from existing, so we cannot assume M doesn't have them. For that reason, we'll define P to be the set of reachable states in M.

Now, each state in P is reachable, which means some string $x \in \Sigma^*$ can cause M to go to that state (starting from the initial state). To recognize the suffix language Suffix(A), we want to allow the NFA N to skip all possible initial strings. What we will do is as follows: we will include all states and transitions of M in N, and additionally, we'll have a new start state q_0. From q_0, we'll add ϵ-transitions to all states in P. The accept states of N will be the same as the accept states of M.

To see why this works, consider a string $x \in$ Suffix(A). Then $yx \in A$ for some string y. Let p be the state reached by M after reading y. Then $p \in P$, since it is reachable. Also, we know that starting from the initial state, M will reach an accept state when reading yx, and hence starting from p, it will reach an accept state when reading x. Now, in N, we have an accepting path when reading x: we can follow the ϵ-transition from the new initial state q_0 to p, and then we can follow the path M takes from p to an accept state while reading x. This means that N accepts x.

Conversely, if N accepts a string x, consider the accepting path in N that starts from q_0 and reads x. Such a path must use some ϵ-transition to leave q_0, so let p be the state of M reached by taking that ϵ-transition from q_0. Then $p \in P$, so p is reachable from the start state of M; let $y \in \Sigma^*$ be a string that causes M to reach p when starting from its initial state. Since the path in N that reads x was an accepting path, we know that there is a path in M that starts from p, reads x, and gets to an accept state. This means that when M reads yx, then it reaches p after reading y and then reaches an accept state after reading x, so M accepts yx. Thus $yx \in A$, so $x \in$ Suffix(A). This completes the proof that $L(N) =$ Suffix(A).

This shows that regular languages are closed under suffixes. What about prefixes? Here we can observe that Prefix(A) = Suffix(A^R)^R. To see this, suppose that $x \in$ Prefix(A). Then $xy \in A$ for some $y \in \Sigma^*$. This means that $(xy)^R \in A^R$, or $y^R x^R \in A^R$. This implies that x^R is a suffix of a string in A^R, and hence $x^R \in$ Suffix(A^R), and so $x \in$ Suffix(A^R)^R. Conversely, suppose that $x \in$ Suffix(A^R)^R. Then $x^R \in$ Suffix(A^R), and hence $yx^R \in A^R$ for some $y \in \Sigma^*$. This means that $(yx^R)^R \in A$, or $xy^R \in A$, which implies that $x \in$ Prefix(A). Since each string in

Prefix(A) is in Suffix(A^R)^R and each string in Suffix(A^R)^R is in Prefix(A), we conclude Prefix(A) = Suffix(A^R)^R. Finally, since we know that regular languages are closed under reverse and under suffixes, we conclude they are closed under prefixes as well.

Finally, let's handle substring. It's not hard to show that Substring(A) = Prefix(Suffix(A)), so the closure under substrings follows from the closure under prefixes and suffixes.

6.4 The "language per pair of states" trick

As mentioned, one useful trick for showing that some modification A' of a regular language A is regular is to start with a DFA for A and to construct an NFA for A'. One trick we've seen that helps with constructing NFAs is adding an additional start state, and then adding ϵ-transitions from that start state to other desired states; this effectively makes the NFA have multiple different start states.

Another useful trick is one we'll call the "language per pair of states" trick. Suppose we start with a DFA

$$M = (Q, \Sigma, \delta, q_0, F).$$

For any pair of states $p_1, p_2 \in Q$, consider the language of all strings $x \in \Sigma^*$ such that $\delta^*(p_1, x) = p_2$. This is the set of all strings x such that if M starts at p_1 and reads x, then M ends up at p_2. Let's denote this language by A_{p_1,p_2} (remember that this language depends on the DFA M that we started with).

we can construct a DFA M_{p_1,p_2} that accepts exactly the strings in A_{p_1,p_2}. This DFA will have set of states Q, alphabet Σ, and transition function δ, but its initial state will be p_1 and its set of accept states will be $\{p_2\}$. The language $L(M_{p_1,p_2}) = A_{p_1,p_2}$, which consists of all strings that cause M to go to p_2 if it was previously at p_1, is therefore a regular language.

These languages A_{p_1,p_2} (and their corresponding DFAs M_{p_1,p_2}) are often useful tools in showing that languages are regular. Let's do an example.

Problem 6.1. *Let A be a regular language over the alphabet Σ, and consider the following language:*

$$B = \{xy : xay \in A \text{ for some } a \in \Sigma\}.$$

The strings in B are those that can be formed by taking a string in A and deleting some symbol from it. Can you show that B is regular?

To start, we can assume that A has a DFA M. We want to show that B is regular. Well, to accept the strings in B, we want to read some prefix using M, then somehow skip a character in M, and then read the rest of the string using M. More concretely, suppose that M accepts $xcy \in A$. Then M reaches state p_1 when reading x, then transitions to state p_2 when reading c, and then moves from p_2 to an accept state p_3 when reading y. If we want to skip reading the character c, we can take language $A_{q_0,p_1}A_{p_2,p_3}$, which is the concatenation of A_{q_0,p_1} (the language of all strings that cause M to go from its initial state q_0 to p_1) with A_{p_2,p_3} (the language of all strings that cause M to go from its p_2 to p_3). The resulting concatenated language is regular, and consists of all strings that cause M to go from q_0 to p_1, then from p_2 to p_3, while teleporting from p_1 to p_2 somewhere in the middle.

Next, we will take the union of $A_{q_0,p_1}A_{p_2,p_3}$ over all triples (p_1, p_2, p_3) of states of M such that there's a transition that goes from p_1 to p_2 (reading a single character), and such that p_3 is an accept state. There are finitely many such triples, so we've taken the union of finitely many regular languages, and therefore the resulting language is regular.

We claim that that the resulting language is precisely B. To see this, we will show both set inclusions, as usual.

Suppose that $w \in B$; then $w = xy$ and there is a symbol $c \in \Sigma$ such that $xcy \in A$. Let p_1 be the state reached by M when reading x (starting at the initial state), that is, $p_1 = \delta^*(q_0, x)$. Let p_2 be the state M reaches from p_1 after reading c, that is, $p_2 = \delta(p_1, c)$. Let p_3 be the state M reaches from p_2 after reading y, that is, $p_3 = \delta^*(p_2, y)$. Then p_3 must be an accept state (since M recognizes A and $xcy \in A$, and since M reaches p_3 when run on xcy). Therefore, the triple (p_1, p_2, p_3) satisfies the conditions that p_2 is reachable from p_1 by reading a single character and p_3 is an accept state, so $A_{q_0,p_1}A_{p_2,p_3}$ is included in the union. We claim that $w \in A_{q_0,p_1}A_{p_2,p_3}$. Since $w = xy$, we only need to show that $x \in A_{q_0,p_1}$ and $y \in A_{p_2,p_3}$; this is true by definition, as we have $\delta^*(q_0, x) = p_1$ and $\delta^*(p_2, y) = p_3$.

Conversely, suppose that w is in the finite union. We wish to show w is in B. Since w is in the union, it is in $A_{q_0,p_1}A_{p_2,p_3}$ for some triple (p_1, p_2, p_3) such that p_3 is an accept state and p_2 is reachable from p_1 after reading a single symbol. In particular, we can write $w = xy$ with $x \in A_{q_0,p_1}$ and $y \in A_{p_2,p_3}$. This means that $\delta^*(q_0, x) = p_1$, $\delta^*(p_2, y) = p_3$, p_3 is an accept state, and there is some symbol $c \in \Sigma$ such that $\delta(p_1, c) = p_2$. From this, it follows that $\delta(q_0, xcy) = p_3$, so M accepts xcy. Thus $xcy \in A$, so $w = xy \in B$, as desired.

Problem 6.2. *Let A be a regular language over the alphabet Σ, and consider the following language:*

$$C = \{xy : yx \in A \text{ for some } x, y \in \Sigma^*\}.$$

Show that C is regular.

We'll use the same "language per pair of strings" trick. Since A is regular, let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing A. We can form the regular languages A_{p_1,p_2} for any pair of states $p_1, p_2 \in Q$. Note that A can be written as the union of $A_{q_0,p_1}A_{p_1,p_2}$ for all (p_1, p_2) such that $p_2 \in F$ (that is, we can split any string w in A into two parts, $w = xy$, such that M goes to p_1 when reading x and then goes from p_1 to p_2 when reading y, where p_2 is an accept state). Motivated by this, we will take the union over all $(p_1, p_2) \in Q \times F$ of the languages $A_{p_1,p_2}A_{q_0,p_1}$. (Here the notation $Q \times F$ denotes all pairs of elements such that the first comes from Q and the second comes from F.)

Note that this is a finite union of regular languages, so it is regular. We now claim that this union is the same language as C. To show this, we will show both inclusions, as usual (that is, we show that each string in C is in the union, and separately show that each string in the union is in C).

First, let $w \in C$. Then by the definition of C, we can write $w = xy$ with $yx \in A$. Let $p_1 = \delta^*(q_0, y)$ be the state M reaches when reading y, and let $p_2 = \delta^*(p_1, x) = \delta^*(q_0, yx)$ be the state M reaches when reading yx from the beginning, which is also the state M reaches when reading x starting from p_1. Since $yx \in A$, we know that M accepts yx, so p_2 is an accept state; hence $(p_1, p_2) \in Q \times F$. Also, by the definition of the languages A_{q_0,p_1} and A_{p_1,p_2}, we have $y \in A_{q_0,p_1}$ and $x \in A_{p_1,p_2}$, so $xy \in A_{p_1,p_2}A_{q_0,p_1}$, and hence $w = xy$ is in the union we defined.

In the other direction, suppose that w is in the union. Then w must be in some specific set $A_{p_1,p_2}A_{q_0,p_1}$ such that $(p_1, p_2) \in Q \times F$. This means we can write $w = xy$ with $x \in A_{p_1,p_2}$ and $y \in A_{q_0,p_1}$, so $\delta^*(p_1, x) = p_2$ and $\delta^*(q_0, y) = p_1$. Also, $p_2 \in F$. From this we conclude that $\delta^*(q_0, yx) = p_2 \in F$, so M accepts yx. Thus $yx \in A$, so $w = xy \in C$. This shows that C is equal to the union we described, and hence C is regular.