Reminder

- In class we introduced non-deterministic finite automata (NFA). NFAs are allowed to have multiple (or none) transitions for the same symbol from the same state, and are also allowed to have "free" $\epsilon$ transitions. We showed that NFAs are equivalent to DFAs in terms of the languages they can recognize - the regular languages.

- Closure properties of regular languages. If $A, B$ are regular, then so are their union ($A \cup B$), concatenation ($A \circ B = AB$), star ($A^*$), intersection ($A \cap B$), complement ($\overline{A}$), reversal ($A^R$), and difference ($A \setminus B$). We will show the last three in these notes.

- The pumping lemma may be used to show non-regularity of languages. But it cannot be used to show regularity!

**Lemma 1** (The Pumping Lemma). If $A$ is a regular language, then there is a number $p$ (pumping length) such that if $s \in A$ and $|s| \geq p$, then $s$ can be written as $s = xyz$ with the following three properties:

1. $xy^iz \in A$, for $i \geq 0$
2. $|y| > 0$
3. $|xy| \leq p$.

*Note:* the $i$ in $xy^iz$ can be zero and generate $xz$. When $i$ is zero we call the operation "pumping down". When $i$ is greater than zero we call the operation "pumping up".

- Examples of some non-regular languages:
  - $\{a^n b^n \mid n \geq 0\}$
  - $\{w \mid w$ has equal numbers of 0s and 1s$\}$
  - $\{a^i b^j \mid i \neq j \text{ with } i, j \geq 0\}$
  - $\{a^i b^j \mid i > j \geq 0\}$

**Proof 1** — The class of regular languages is closed under complement

If $A$ is any language, let $\overline{A}$ be the complement of $A$. Formally, $\overline{A} = \{w \mid w \notin A\}$.

Prove that the regular languages are closed under complement. That is, prove that if $A$ is regular, then $\overline{A}$ is also regular.
Solution

Given a DFA $M$ that recognizes $A$, we construct another DFA $N$ that recognizes $\overline{A}$. $N$ is the same as $M$, except that its accepting and non-accepting states are switched. Formally, if $M = (Q, \Sigma, \delta, q_0, F)$, then we have $N = (Q, \Sigma, \delta, q_0, Q \setminus F)$. Here, $Q \setminus F$ denotes the difference between set $Q$ and set $F$, that is all elements in $Q$ that are not in $F$.

Proof 2 — The class of regular languages is closed under reversal

If $A$ is any language, let $A^R$ be the reversal language of $A$. Formally, $A^R = \{ w \mid w^R \in A \}$, where for $w = w_1w_2 \cdots w_n$, $w^R = w_nw_{n-1} \cdots w_1$.

Prove that the regular languages are closed under reversal. That is, prove that if $A$ is regular, then $A^R$ is also regular.

Solution

Given a DFA $M$ that recognizes $A$, we describe a NFA $N$ that recognizes $A^R$ using the following idea. The transition function will be the “inverse” of the original transition function, meaning we reverse the directions of all the arrows. We turn the start state into the accepting state and the accepting states into the start state. But what if $M$ had more than one accepting state? We add one additional state to be the new single starting state, and we connect it via $\varepsilon$-transitions to the accepting states of $M$. The formal definition follows.

Let $M = (Q, \Sigma, \delta, q_0, F)$. Construct NFA $N = (Q \cup \{ t \}, \Sigma, \delta', t, F')$, for $t \notin Q$, that recognizing the reversal of $A$ as follows:

1. $F' = \{ q_0 \}$.
2. $\delta'(q, a) = \{ r \in Q \mid \delta(r, a) = q \}$ if $q \neq t$, and $\delta'(t, \varepsilon) = F$.

Note

We start with a DFA for $A$, and we build an NFA (and not a DFA) for $A^R$. This is for two reasons:

1. We use $\varepsilon$-transitions if $A$ has more than one accepting state, and a DFA couldn’t have $\varepsilon$-transitions.
2. When we invert the transition functions (and therefore invert the arrows), there could be multiple arrows going out of a state with the same label! That’d make a DFA invalid.

Note that it is possible to build a DFA for $A^R$ (since any NFA can be converted to a DFA), but it would be much more convoluted.

Proof 3 — The class of regular languages is closed under difference

If $A$ and $B$ are languages, let $A \setminus B$ be the difference between $A$ and $B$. Formally, $A \setminus B = \{ w \mid w \in A \text{ and } w \notin B \}$.
Prove that the regular languages are closed under difference. That is, prove that if $A$ and $B$ are regular, then $A \setminus B$ is also regular.

**Solution**

We can easily show this by using other closure properties we already proved to hold. Specifically, the difference of two sets can be rewritten as the union of one set with the complement of the other: $A \setminus B = A \cap \overline{B}$. Since the regular languages are closed under intersection and complement, it follows that they are also closed under difference.

**Exercise 1 — Proving non-regularity**

Prove that the following languages are non-regular.

1. $L_1 = \{0^n10^n \mid n \geq 0\}$.
2. $L_2 = \{0^i1^j \mid i > j \geq 0\}$.
3. $L_3 = \{w \mid \text{the number of 0s is different from the number of 1s}\}$.
4. $L_4 = \{a^m b^n \mid m \neq n\}$

**Solutions**

1. We use the pumping lemma to “pump up”. Assume $L_1$ is regular. Let $p$ be the pumping length. Set $s = 0^p10^p$. Obviously, $s \in L_1$ and $|s| \geq p$. Thus, the pumping lemma implies that the string $s$ can be written as some concatenation of strings $xyz$. From condition 3 of the pumping lemma ($|xy| \leq p$), we have that the substrings $x$ and $y$ can be composed only of zeroes (because the first $p$ characters of $s$ are zeroes). Then, if we pump up $y$ to, say $y^{10}$, we obtain more zeroes. These zeroes end up in the first part of $s$, but recall that the language requires the number of zeroes before the 1 to be equal to the number of zeroes after the 1. This is clearly not the case anymore after we pump up $y$ so that the number of zeroes before the 1 grows to be bigger than $p$: the resulting string is $0^k10^p$ with $k > p$. This contradicts the pumping lemma. Thus $L_1$ is not regular.

2. We use the pumping lemma to “pump down”. Assume $L_2$ is regular. Let $p$ be the pumping length. Set $s = 0^{p+1}1^p$. Obviously, $s \in L_2$ and $|s| \geq p$. Thus, the pumping lemma implies that the string $s$ can be written as some concatenation of strings $xyz$. From condition 3 of the pumping lemma ($|xy| \leq p$), we have that the substrings $x$ and $y$ can be composed only of zeroes (because the first $p$ characters of $s$ are zeroes). However, the string $s' = xy^0z$ is not in $L_2$, since we have removed at least a zero from the first part of $s$. We started with a string with the number of zeroes larger than the number of ones by exactly 1. We removed at least 1 zero, so that makes the resulting string have at least the same number of zeroes and ones (and possibly less zeroes than ones). Therefore, the resulting string is not in the language. This contradicts the pumping lemma. Thus $L_2$ is not regular.
3. We use the closure properties of regular languages. Assume $L_3$ is regular. Since the regular languages are closed under complement, $\overline{L_3} = \{w \mid w \text{ has the same number of } 0\text{s and } 1\text{s}\}$ is regular. But $\overline{L_3}$ is not regular (can you prove it?). This is a contradiction, because if $L_3$ was regular, then its complement would be regular as well. But the complement of $L_3$ is not regular. Thus $L_3$ is not regular.

4. We use the closure properties of regular languages. Assume $L_4$ is regular. Since the regular languages are closed under difference, the language $L_d = (a^*b^*) \setminus L_4 = a^n b^n$ must be regular, since $a^*b^*$ is also regular (it’s described by a regular expression, and therefore is regular). However, we know that $a^n b^n$ is regular, a contradiction. Thus, $L_4$ is be regular.

Exercise 2 — A regular language in disguise

Let $L = \{w \mid w \text{ has the same number of } “ab” \text{ and } “ba” \text{ substrings}\}$. Prove that $L$ is regular.

Solution

At first glance, $L$ looks non-regular. After all, it seems like an automaton has to count the number of “ab” and “ba” substrings and then check that the number is the same for both. But a finite automaton cannot count to an unbounded amount!

Turns out $L$ can be expressed in a simpler manner, one that shows that the language is indeed regular. Let’s take a look at some example of strings that are in $L$: $ababa, aba, abbbbbba, bbbbbabbbbb$. Do you notice anything?

If we start reading the string and we find an $ab$ substring (with potentially multiple $b$s, e.g. $abbbb$), then we must expect an $a$ immediately after the last $b$ in order for the string to be in the language.

If we see another $b$ after that $a$, then we must expect another $a$, and so on.

If we start reading the string and we find an $ba$ substring (with potentially multiple $a$s, e.g. $baaaa$), then we must expect an $b$ immediately after the last $a$ in order for the string to be in the language.

If we see another $a$ after that $b$, then we must expect another $b$, and so on.

We can easily construct a DFA for this language (we must accept the empty string as well):

![DFA Diagram]

Therefore $L$ is regular.
Appendix 1 — Are non-regular language closed under the regular operations?

Are non-regular languages closed under the same operations as the regular languages? The answer is not necessarily. For instance, the non-regular languages are not closed under the regular operations (union, concatenation, and star) or even under some other operations (e.g., intersection), but are closed under others (e.g., complement and reversal). We will explain why for union, complement, and reversal.

To see why non regular languages are not closed under union, we show an example. Consider a non regular language $A$ and its complement $\overline{A}$. These are two non regular languages (if the complement was regular, then $A$ would be regular as well). However, $A \cup \overline{A} = \Sigma^*$, which is regular. Thus, the non regular languages are not closed under the union operation.

To see why non regular languages are closed under complement and reversal, consider what happens if you assume that they aren’t, and then try to apply the operation twice. You should be able to easily see a contradiction.

Appendix 2 — How does the pumping lemma apply to finite languages?

Recall that in the context of the pumping lemma, the operation of ”pumping” a string up generates an infinite amount of strings (since the $i$ in $xy^iz$ is unbounded). If the language is regular, the pumping lemma holds and all those strings generated by pumping are contained in the language.

But what if the starting language is finite? How can we pump any string in the language and generate infinite strings from it? The answer is that we cannot, and we are not required to!

Before we explain why, notice the following facts about finite languages:

1. Every finite language is regular (see last week’s recitation notes for a proof), so the pumping lemma must hold.

2. Every finite language has a corresponding DFA that recognizes it, and such DFA has no loops (or, to be precise, if there is any loop, it cannot lead to any accepting state).

Recall that the pumping length $p$ is usually set to the number of states. The number of states of a DFA without loops essentially imposes a bound on the maximum length of the strings that such DFA accepts. Therefore, for a DFA with no loops and $p$ states, the maximum length of the strings it can accept is $p - 1$ (this is the maximum number of transitions it can take, and therefore the maximum number of input characters it can read).

Now, the pumping lemma must hold for any string $s$ such that $|s| \geq p$. But we just showed that there can’t be any string in the language that has length greater or equal than $p$! Therefore, we say the pumping lemma is vacuously true, that is, it is true because there’s no string to impose its requirements on. The pumping lemma doesn’t say anything about requirements for strings of length less than the pumping length.