Lecture 7: Nonregular Languages

built on 2021/01/26 at 10:00:24

1 Recap: What's Regular?

- You've seen DFAs. You know what they look like. They have finitely many states, work on finitely many symbols, and require "memory" constant in the length of the input string.
- A language is *regular* if there is a DFA that recognizes it. Here, the word *recognize* means accepts only strings in the language and rejects all the nonmembers.
- DFA == NFA because every DFA is already an NFA, and an NFA can be converted to a DFA with the same behavior.
- regular expressions == NFA because we can convert any regex to an NFA and vice versa.

2 Is Every Language Universally Regular?

Consider the following languages. Please draw a DFA/NFA for each of them (skip what you can't do):

1. L_1 is all binary strings that have an equal number of occurences of 01s and 10s as substrings. (A: $0 + 1 + \varepsilon + 0\Sigma^*0 + 1\Sigma^*1$)
2. L_2 is all binary strings that have an equal number of 0s and 1s as substrings. (A: err.. not possible)
3. L_3 is all binary strings of the form 0^i1^i for $i \geqslant 0$. (A: err... impossible)

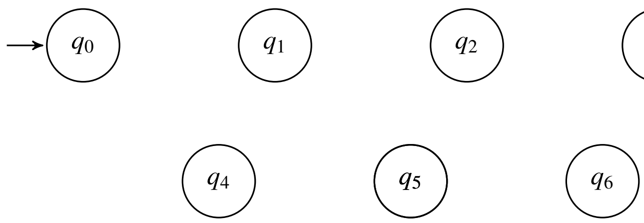
I claim that L_2 and L_3 don't have a corresponding DFA and therefore are not regular. But how can we prove that this is the case? Arguing that we are unable to draw a DFA isn't at all a good argument. Perhaps someone else can.

We need an airtight argument that shows that no DFA whatsoever can correctly recognize L_2 or L_3.

For concreteness, let us focus on the language L_3, which can be written formally as

$$L_3 = \{0^n1^n \mid n \in \mathbb{Z}_{\geqslant 0}\}$$

Why can't this language have a DFA that recognizes it? Say you can somehow draw a DFA for it. I don't know how many states it will have, but let's make up a number. Suppose your DFA has 7 states.



I'll keep the arrows and the accepting states rather hazy for now. It probably doesn't matter all that much. Imagine giving it the input

$$0^7 = 0000000$$

If we keep tabs of where we end up after each symbol, we'll probably have something like this:

After Input	State
–	q_0
0	q_2
00	q_5
000	q_6
0000	q_1
00000	$\boxed{q_3}$
000000	q_4
0000000	$\boxed{q_3}$

The important point here is that there is a state q_i that we end up in multiple times (in this example q_3).

One thing about DFA is that it doesn't remember the past, so *once you are in q_i and given the string y, you'll end up in the same state regardless of which string you used to reach q_i in the first place.*

To be precise, define $\delta^* : Q \times \Sigma^*$, where $\delta^*(q, S)$ is the state we are in starting from q after processing the string S. If you're pedantic, you'll write $S = s_0 s_1 \dots s_{k-1}$. Then $r_0 = q$ and $r_{i+1} = \delta(r_i, s_i)$, and $\delta^*(q, S) = r_k$.

This means: in the above example, this DFA can't tell the difference between 00000 and 0000000. This further means if

$$x = 0000011111 \quad \text{and} \quad y = 000000011111$$

this DFA will either accept both x and y—or reject them both. This can't be good. Specifically, the problem with this DFA is that

- On input x, after consuming five 0s, the machine is in state q_3. Formally, we say $q_3 = \delta^*(q_0, 00000)$. Then, for the following five 1s, it will end up in $q = \delta^*(q_3, 11111)$.
- On input y, after consuming seven 0s, the machine ends up in state q_3—again, $q_3 = \delta^*(q_0, 0000000)$. Then, for the following five 1s, it will end up in $q = \delta^*(q_3, 11111)$.

The crux of the contradiction is that q can't be both accepting and nonaccepting.

2.1 More Formally...

Let's spell this one as a formal proof. Notice that in general:

- we don't know how many states the claimed DFA has.
- we don't know which state is repeated, either.

As you'll see, this can be fixed using the pigeonhole principle.

Proof: Suppose for a contradiction that there exists a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that decides L_3 using $k \geqslant 1$ states. We define r_i to be the state that M is in after consuming the string 0^i stating from the initial state q_0, that is,

$$r_i = \delta^*(q_0, 0^i) \text{ for } i \geqslant 0$$

By pigeonhole, there are repeats among $r_0, r_1, \dots, r_k$. This means $r_s = r_t$ for some $0 \leqslant s \neq t \leqslant k$. Consider the following:

- Because $0^s 1^s \in L_3$, the state $f_s = \delta^*(q_0, 0^s 1^s) = \delta^*(\delta^*(q_0, 0^s), 1^s) = \delta^*(r_s, 1^s)$, which is the final state after reading the input $0^s 1^s$, must be an accepting state, so $f_s \in F$.
- On a different input $0^t 1^s$, its final state $f_t = \delta^*(q_0, 0^t 1^s) = \delta^*(\delta^*(q_0, 0^t), 1^s) = \delta^*(r_s, 1^s) = f_s$.

But this is absurd: $0^t 1^s \notin L_3$, a contradiction. Hence, such a machine M cannot exist. ■

Following this structure, proving that L is not regular typically involves:

1. Assume for contradiction there is a DFA M that recognizes L.
2. Argue (usually by Pigeonhole) there are two strings x and y which reach the same state in M.
3. Show a string z such that $xz \in L$ but $yz \notin L$. This is a contradiction as M acts the same on both.

3 The Pumping Lemma

One common technique that aids in proving a language L is not regular is traditionally called the *pumping lemma*. It states that every regular language has some (defining) special properties. If we can show that L doesn't have some of these special "features," L surely cannot be regular.

Lemma 3.1 (The Pumping Lemma) *Let A be a regular language. There is an integer $p \geqslant 1$ "the pumping length" such that for every string $s \in A$ of length at least p, then s can be written as $s = xyz$ satisfying*

(1) $xy^i z \in A$ *for every* $i \geqslant 0$*;*

(2) $|y| > 0$*; and*

(3) $|xy| \leqslant p$.

Remember that $|x|$ is the length of the string x, and y^k means repeating y a total of exactly k times. At this point, it's worth unpacking the lemma a bit before we see some applications of the lemma.

- Pretty much, the lemma says that if A is regular, there is a pumping length p. This is a property of the language—and this value does *not* depend on the string s that is discussed later in the lemma.
- Now you can pick any string $s \in A$, and this string must be able to be broken into *three* pieces x, y, and z such that ...
- the y part can be pumped an arbitrary number of times ($y^0 = \varepsilon$)
- the y part cannot be empty (it's still true if y could be empty... but not very useful)
- finally xy together cannot be longer than p.

The proof of this lemma is pretty much a more streamlined version of what we did earlier in this lecture. Instead, we'll look at a few applications of the lemma.

Example 3.2 *Consider the language $L = \{0^n1^n \mid n \in \mathbb{Z}_{\geqslant 0}\}$. We'll use the pumping lemma to show that L isn't regular.*

Proof: *The proof is by contradiction. Assume for a contradiction that L is regular, so there is a "pumping length" $p \geqslant 1$ given by the pumping lemma. Consider $s = 0^p1^p$. Clearly $s \in L$—and as $|s| \geqslant p$, s can be split into $s = xyz$ satisfying the conditions of the lemma.*

Because $|xy| \leqslant p$, xy is made up of all 0s, so y itself consists only of 0s. This means, xyyz has more 0s than 1s, and $xyyz \notin L$, a contradiction to the lemma's assertion that xy^2z must be in L! Hence, L is not regular. ∎

Example 3.3 *Let $L_2 = \{w \mid w$ has an equal number of 0s and 1s$\}$. Show that C isn't regular. Try it out for yourself first.*

Pretty much the proof above works.

Example 3.4 *Let $\Sigma = \{0, 1\}$ and $L_3 = \{ww \mid w \in \Sigma^*\}$. Show that L_3 isn't regular.*

Proof: *The proof is by contradiction. Assume for a contradiction that L is regular, so there is a "pumping length" $p \geqslant 1$ given by the pumping lemma. Consider $s = 0^p10^p1$. Clearly $s \in L$—and as $|s| \geqslant p$, s can be split into $s = xyz$ satisfying the conditions of the lemma.*

Because $|xy| \leqslant p$, xy is made up of all 0s, so y itself consists only of 0s. This means, xyyz cannot be in L_3, a contradiction! ∎

The challenge in coming up with such a proof is in finding the right s that exhibits the "essence" of nonregularness.

Example 3.5 *Consider $L_4 = \{0^i1^j \mid i > j\}$. Show that L_4 isn't regular. (Hint: pump down)*

Proof: *The proof is by contradiction. Assume for a contradiction that L is regular, so there is a "pumping length" $p \geqslant 1$ given by the pumping lemma. Consider $s = 0^{p+1}1^p$. Clearly $s \in L$—and as $|s| \geqslant p$, s can be split into $s = xyz$ satisfying the conditions of the lemma.*

Because $|xy| \leqslant p$, xy is made up of all 0s, so y itself consists only of 0s. Pumping up doesn't help in this case: $xyyz \in L_4$, $xyyyz \in L_4$, ...; however, the pumping lemma works even when $i = 0$. So then, consider that $xy^0z = xz \notin L_4$ (because removing y reduces the number of zeros). This gives a contradiction! ∎

4 Practice

1. Show that L_3 above is not regular directly via a contradiction proof (i.e., without using the pumping lemma).
2. Let $A = \{\mathtt{0}^n\mathtt{1}^n\mathtt{2}^n \mid n \geqslant 0\}$. Show that A isn't regular using the pumping lemma.
3. Let $B = \{\mathtt{0}^n\mathtt{1}^m\mathtt{0}^n \mid n \geqslant 0, m \geqslant 0\}$. Show that A isn't regular using the pumping lemma.