CSE547T Class 7

Jeremy Buhler

February 8, 2017

1 The Goal

So far, we know several ways to prove that a language is regular.

- Build it constructively via recursive defn.
- Equivalently, build a regular expression to recognize it.
- Build an FA (DFA, NFA, ε-NFA) to recognize it.

Well and good. But how powerful are regular languages?

- Is *every* language regular? No! (We proved this a while ago)
- OK, but maybe only "weird" languages are non-regular?
- To investigate, we need a way to tell with certainty that a language is *not* regular.
- (It is not enough to say that we tried but couldn't find a machine/RE for it.)

2 The Idea

How can we check for non-regularity?

- **Key idea**: prove that some property holds for *every* regular language.
- That is, "If L is regular, L has property P."
- Turning this around, we can prove...
- "If L lacks property P, L is not regular."
- So far, the only nontrivial property P we know for regular languages is that they can be recognized by an FA. But we know of no way (yet) to show that an FA cannot recognize a language!
- We're going to show another important property (by appeal to DFAs).

Start from: if L is regular, L is recognized by a DFA M.

- Let's suppose that L is infinite (finite languages are trivially regular).

- M has only a finite number of states, say n.
- Yet there are arbitrarily long strings $x \in L$ that M must accept.
- What can we say about the path taken in M on such a long string x?
- I claim that this path must contain a *cycle*!
- Why? Pigeonhole: path has more than n steps but can go through at most n distinct states, so some state repeats.

- So what? Given $x \in L$ that takes a path through M that includes a cycle, can we find a longer string $y \in L$? A shorter string $z \in L$?
- (repeat or drop the cycle, respectively)
- turning x into y or z is called "pumping" the cycle.

Here's what we just argued, restated as a lemma.

3 The Lemma

Pumping Lemma: Let L be a regular language. Then

- there exists an integer n, such that
- for every string $x \in L$ with $|x| \geq n$,
- x can be divided into parts uvw, $|uv| \leq n$, $|v| > 0$, so that
- for every $m \geq 0$, $uv^m w \in L$.
- **Pf**: Let M be a DFA accepting L, and set n to the number of states in M.
- Let $x \in L$ be a string of length $\geq n$.
- x traces out some accepting path p in M.
- Since p contains at least n edges, hence $n+1$ vertices, it contains a cycle. Consider the first such cycle in p.
- Divide x into uvw, such that u labels the (acyclic) part of the path prior to the cycle, v labels the cycle itself, and w labels the part of the path after the cycle.

- The cycle contains at least one edge, so $|v| > 0$.
- Note that the first repeated vertex must occur after traversing at most n edges (= $n + 1$ vertices) of p, so $|uv| \leq n$.
- Finally, we can modify the path p by repeating the cycle any number of times $m \geq 0$.
- Conclude that string $uv^m w$ is also accepted by M, and so $uv^m w \in L$. QED

4 The Contrapositive

How can we use the Pumping Lemma to prove a language non-regular? Turn it around (via the contrapositive). Nasty quantifier hacking ahoy!
Pumping Lemma (CP): Let L be any language. Suppose that

- for every n,
- there exists a string $x \in L$ with $|x| \geq n$, such that
- for every division of x into parts uvw, $|uv| \leq n$, $|v| > 0$,
- there exists an $m \geq 0$ such that
- $uv^m w \notin L$.

Then L is not regular.

5 The Examples

OK, time for examples. First, though, let's describe the general principle.

- As usual with alternating quantifiers, think of this as a game.
- Your victory condition: prove L non-regular.
- You are playing against the "adversary."
- Adversary moves first – picks n.
- You move second – pick $x \in L$ with $|x| \geq n$.
- Adversary moves third – picks a valid division into uvw.
- You move fourth – pick m.
- If you can *always* move so as to force $uv^m w \notin L$, you win!

Here's the canonical example.

- Let $L = \{x \in \{0,1\}^* \mid x \text{ has form } 0^n 1^n\}$.
- **Claim**: L is not regular.
- Let n be chosen. We set $x = 0^n 1^n$, which surely has length $\geq n$.

- Consider any valid division of x into uvw.
- Since $|uv| \leq n$, v consists entirely of 0's!
- We choose $m = 2$ (though any $m \neq 1$ will do).
- Note that uv^2w has more 0's than 1's and so is not in L.
- We win! L is not regular. QED

Hey, what does this mean anyway?

- A good example of a fundamental observation: "FAs cannot count!"
- Informally: if you need to check equality of two numbers to verify $x \in L$, you cannot use an FA to recognize L.
- *Standard example*: parenthesis matching.
- In parsing (say) an arithmetic expression, an FA cannot verify that the expression has balanced parentheses! (Equal # of left and right parens.)
- *Moral*: our machines are not powerful enough to recognize some interesting and useful languages!

Another example:

- Define y^R to be the reverse of a string y.
- Let $L = \{x \in \{0,1\}^* \mid x = yy^R \text{ for some } y \in \{0,1\}^*\}$.
- **Claim**: L is not regular.
- Let n be chosen. We set $x = 0^n110^n$.
- Again, consider any valid division of x into uvw.
- Since $|uv| \leq n$, v again consists entirely of 0's!
- Just for fun, we set $m = 0$ (pumping "down" instead of "up")
- Resulting string uw has unbalanced 0's, hence is of wrong form.
- Conclude that $uw \notin L$, and so L is nonregular. QED

One more example:

- Let $L = \{0^k \mid k \text{ is a prime number}\}$.
- **Claim**: L is not regular.
- Let n be chosen. We set $x = 0^p$, where p is a prime $\geq n$.
- Consider any valid division of x into uvw, and suppose $|v| = j$ $(1 \leq j \leq n)$.
- Let $z = uv^{p+1}w$. String z has length $p - j + (p+1)j = p(j+1)$.
- Both p and $j+1$ are > 1, so $|z|$ is not prime.
- Conclude that $z \notin L$, and so L is nonregular. QED
- (Hence, finite automata cannot perform primality testing on unary numbers!)

6 Caveat

- The Pumping Lemma shows that languages with a certain (complex) property are not regular.
- Is the converse true? That is,can *every* non-regular language be proven non-regular using the technique in the Pumping Lemma?
- *No!* There exist non-regular languages for which the test will not fail, so we can't prove them non-regular using the lemma.
- Can we find a precise characterization of the non-regular languages? Stay tuned...