

Module 4

Properties of regular languages

Not everything is regular.

CS 360: Introduction to the Theory of Computing
Winter 2022

Collin Roberts
University of Waterloo

Topics for Module 4

- ▶ Proving languages non-regular: the Pumping Lemma
- ▶ Closure rules for regular languages
- ▶ Algorithms for decision problems about finite automata and regular languages.

Where are we?

- ▶ We have given definitions for regular languages, and shown their strong connection to FAs.
- ▶ If we apply certain operations to regular languages, we get back a regular language.
- ▶ Are all languages regular?
 - ▶ Obviously no: we are going to have 8 more weeks in the term, and we are getting to the end of regular languages.
 - ▶ In this section, we will think more about what makes a language regular.

Non-regular languages

- ▶ By Kleene's Theorem, a language L is **not** regular if for every DFA M , $L \neq L(M)$.
- ▶ So if we characterize languages of DFAs (that is, regular languages) very carefully, maybe we can also characterize those languages that are **not** regular.

How does a DFA M work?

- ▶ Suppose it has n states.
- ▶ Consider a word x in $L(M)$ with $|x| \geq n$.
- ▶ On its path from q_0 to an accept state, it must repeat a state somewhere along the path.
 - ▶ (Why? There are only n states in total, and the machine starts out in one of them, then reads $\geq n$ input characters.)
 - ▶ Arguments of this type use the **pigeonhole principle**.

Decompose the word into parts

Let's say that we repeat state r .

- ▶ Then the word x can be decomposed: $x = uvw$, where:
 - ▶ u = the part from q_0 to the first time we reach r (i.e. after processing u , we are in state r).
 - ▶ v = the loop from r to itself (i.e. after processing v , we are again in state r).
 - ▶ w = The part from the second time we reach r that leads us to an accept state
 - ▶ **Note:** it is possible that either u or w is ε , but v cannot be ε .
- ▶ This decomposition is possible for any word x in $L(M)$ with $|x| \geq n$.
- ▶ **Fact:** $uvvw$ is also in $L(M)$. Why?
 - ▶ vv also takes M from r back to itself: $\hat{\delta}(r, vv) = r$.
- ▶ Another word in $L(M)$ is $uw = uv^0w$.
- ▶ We can show (by induction) that $uv^*w \subseteq L(M)$.

More about regular languages

We can decompose any word x in $L(M)$ of length at least n this way.

- ▶ If we choose the first time a state is repeated, then $|uv| \leq n$.
 - ▶ Why? (The machine has n states, so we must have the first repeated state by the n th step.)
- ▶ And $|v| \geq 1$, since it is a DFA, and therefore has no ϵ -transitions.

Let's formalize this:

- ▶ Given a DFA M with n states, and a word x in $L(M)$, with $|x| \geq n$, x can be decomposed as $x = uvw$, where
 - ▶ $|uv| \leq n$,
 - ▶ $|v| \geq 1$ and
 - ▶ $uv^*w \subseteq L(M)$.

Pumping lemma

This fact is sometimes called the “Pumping Lemma”:

- ▶ We can pump out many copies of v , and $uvvvvvvvvvvw$ is still part of $L(M)$.

It can be seen as a statement about regular languages.

- ▶ Every regular language L is accepted by a DFA.
- ▶ For a given regular language L , there exists some **smallest** DFA (i.e. with the fewest states), M , that accepts L . Let's say M has n states.
- ▶ Therefore there is some n such that we can make the above statement about M .

Formal Pumping Lemma

For every regular language L , there **exists** some positive integer n such that all words $x \in L$ with $|x| \geq n$ can be decomposed into $x = uvw$, where:

- ▶ $|uv| \leq n$,
- ▶ $v \neq \varepsilon$, and
- ▶ $uv^i w \in L$ for all non-negative integers i .

You can think of n as being the number of states in a machine accepting L .

Again, this describes all long words in a regular language:

- ▶ For some definition of “long”, **all long words can be pumped**.
- ▶ Note that, if L is finite (and therefore regular), then taking any $n > \max_{x \in L} \{|x|\}$ works (because with such an n , L contains **no** long words).

Non-regular languages

We know something about regular languages: long words can be pumped.
Now let's describe some non-regular languages:

- ▶ Suppose that we have a language L .
- ▶ Suppose that **no matter how** we define “long”, there are **still** long words in L that cannot be pumped.
- ▶ Then L is not regular, because **all** regular languages have a definition of “long” for which **all** long words can be pumped.

Formally

- ▶ Let L be a language.
- ▶ Suppose that for **any** positive integer n :
 - ▶ There exists a word $x \in L$ with $|x| \geq n$ such that
 - ▶ for **any** decomposition of x into $x = uvw$, with $|uv| \leq n$ and $v \neq \varepsilon$,
 - ▶ uv^*w is **not** a subset of L .
- ▶ Then L is not a regular language.

That is a pile of negations and existences.

Again, the basis of the Pumping Lemma

- ▶ Language L is regular if it is accepted by **some** DFA.
- ▶ Suppose L is accepted by a DFA, M , with n states.
- ▶ Any word $x \in L$ with at least n letters includes a state cycle: some state r appears two times.
- ▶ This reuse of r corresponds to a substring v of x , so $x = uvw$.
When we start v in state r , we also end in state r : $\hat{\delta}(r, v) = r$.
- ▶ If we got to the start of v (by reading in u), went through the cycle twice, and then finished with w we would wind up at the same accept state in M . So $uvvw$ and $uvvww$, and all of uv^*w is in L .

Explaining Pumping Lemma proofs of non-regularity

Now, what about using the Pumping Lemma to prove a language L is not regular?

- ▶ “Suppose that for any value of $n > 0$, there exists a word $x \in L$ with $|x| \geq n$ ” ... *(If there is always a long word in L)*
- ▶ “such that for any decomposition of x into $x = uvw$, with $|uv| \leq n$ and $v \neq \varepsilon$ ” ... *(that cannot be decomposed into three parts where the first 2 parts are not long and the middle part is non-trivial)*
- ▶ “ $uv^*w \notin L$.” ... *(and the second part cannot be pumped,)*
- ▶ Then L is not a regular language.

An example

Let's show an example:

- ▶ Theorem: $L = \{0^i 1^i \mid i \geq 0\} = \{\epsilon, 01, 0011, 000111, 00001111, \dots\}$ is not a regular language.

Proof:

- ▶ For any $n > 0$, choose a word $x \in L$ whose length is at least n .
- ▶ We will choose $x = 0^n 1^n$. This is our **long word**.
- ▶ Now, consider all decompositions $x = uvw$, where $|uv| \leq n$, and $v \neq \epsilon$.
- ▶ **Fact:** for any such decomposition, $uv = 0^k$ for some $0 < k \leq n$, because the first n characters of $x = uvw$ are all 0 (by the definition of x).
- ▶ Now, we must show that because of what we found, uv^*w is not a subset of L . In particular, we must find an $i \geq 0$ such that $uv^i w \notin L$. (Typically, $i = 0$ or $i = 2$.)
- ▶ Let $i = 0$. Recall that v is all 0's. Then $uv^0 w$ will have fewer 0's than 1's. So $uv^0 w \notin L$.
- ▶ And hence the language L is not regular.

Again, how did that work?

Pumping lemma: to prove languages are not regular.

- ▶ For any definition of **long**, find a long word:
Long: length $\geq n$. Our long word was $x = 0^n 1^n$.
- ▶ Consider all breakdowns of x into $x = uvw$, where uv is short and $v \neq \epsilon$.
For the long word x , if $x = uvw$, and uv is short, then uv is all 0's.
- ▶ If for all of these breakdowns $x = uvw$, we cannot pump v , then L is not regular.
No matter what v is, it must be all 0's. So if we pump v , then $uvvw$ or uw both have the wrong number of 0's. So L is not regular.
- ▶ We can also prove L is not regular by thinking of possible DFAs for L and showing that they cannot exist.
- ▶ This is hard in general. The Pumping Lemma is better.

Another example

We saw that $\{0^i 1^i \mid i \geq 0\}$ is not regular.

Another case:

Theorem: The language $L = \{0^p \mid p \text{ is a prime}\}$ is not regular.

- ▶ (This language includes 00, 000, 00000, 0000000, 00000000000, ...)
- ▶ Proof by Pumping Lemma. (Assume that there are infinitely many primes. There are many nice proofs of this fact.)
 - ▶ Choose a value of $n > 0$.
 - ▶ Choose $x = 0^p$, for a prime $p \geq n$.
 - ▶ Then x is a long word in L .
 - ▶ Now we must argue that no decomposition of x can be pumped.

Why can we not pump the primes?

So $x = 0^p$, for $p \geq n$, p a prime.

Consider all decompositions $x = uvw$, where $|uv| \leq n$ and $v \neq \varepsilon$.

- ▶ Then $v = 0^k$ for some $1 \leq k \leq n$.
- ▶ And $uv^*w = \{0^{p-k}, 0^p, 0^{p+k}, 0^{p+2k}, \dots\}$.
- ▶ Is it possible that all of these are in L ?
- ▶ No. One member of uv^*w is $0^{p+(pk)}$; it is the $(p+2)^{th}$ member in the above list.
- ▶ This word is not a member of L , since $p + pk = (1+k)p$ is composite (both factors are non-trivial, as $k \geq 1$).

For any n , we can find a long word, such that all decompositions of it cannot be pumped.

Therefore L is not regular.

Another example: palindromes

$L = \{s \mid s = s^R\}$ (This is the language of palindromes.)

- ▶ Examples: 0110, 01110, ε , 1111, etc.

L is not regular.

Proof by Pumping Lemma.

- ▶ Given a value of $n > 0$, find a word in L of length at least n .
- ▶ How about $x = 0^n 1 0^n$?
- ▶ Now, consider all decompositions of this into $x = uvw$, where uv is short and v is not ε .
- ▶ Again, v must be 0^i for some $1 \leq i \leq n$.
- ▶ And the number of 0's before the only 1 in uv^2w is more than the number after it, so it cannot be a palindrome.
- ▶ So we cannot pump x , regardless of our choice of decomposition.
- ▶ So L is not regular.

One more example

Let $L = \{y!z \mid |y| > |z|, y, z \in \{0, 1\}^*\}$.

- ▶ $\Sigma = \{0, 1, !\}$

This language includes words like 111!00, 1!, 10001!111. **Fact:** L is not regular.

Proof by Pumping Lemma.

- ▶ Consider a value $n > 0$.
- ▶ The string $x = 0^n!0^{n-1}$ is long, and in L .
- ▶ We will show that uv^0w is not in the language.
 - ▶ Decompose $x = uvw$ with uv of length at most n and nonempty v .
 - ▶ For all such decompositions, $v = 0^k$ for some $k \geq 1$.
 - ▶ And $uv^0w = 0^{n-k}!0^{n-1}$.
 - ▶ This is not a word in L : the part before the ! character is too short.
 - ▶ So v is not pumpable, no matter how we do it.
- ▶ L is not regular.

What can go wrong?

It is easy to misuse the Pumping Lemma.

- ▶ The existence of one bad decomposition of x does not matter.
- ▶ We must show that **all** decompositions of $x = uvw$ with $|uv| \leq n$ and $v \neq \varepsilon$ cannot be pumped.

Example:

- ▶ Obviously, $L = (01)^*$ is regular.
- ▶ For any value of $n > 0$, $(01)^n$ is a long word in L .
- ▶ Decompose into $u = 0, v = 1, w = (01)^{n-1}$.
- ▶ Then $uv^2w = 011(01)^{n-1}$ is not in L .
- ▶ So we conclude that L is not regular (?!?!?)

Clearly we have done something wrong!

- ▶ Problem: We must show that **no** decomposition can be pumped.
- ▶ The decomposition $u = \varepsilon, v = 01, w = (01)^{n-1}$ is pumpable.

More Pumping Lemma: pitfalls

- ▶ The Pumping Lemma:
 - ▶ Long words in regular languages can be pumped.
- ▶ Its contrapositive:
 - ▶ If a language has long words that cannot be pumped, it is not regular.
- ▶ **Note:** the theorem does not give a definition of regular languages. The following is **not** true:
 - ▶ If all long words in a language can be pumped, it is regular.
- ▶ In fact, some non-regular languages can be pumped.

Closure rules

Regular languages are closed under $*$, union and concatenation. This is by definition:

- ▶ A class of languages is closed under a binary operation if applying that operation to 2 languages in the class always yields a language in the class
- ▶ A class of languages is closed under a unary operation if applying that operation to one language in the class always yields a language in the class.

Subsets of regular languages are not necessarily regular: $(0 + 1)^* = \Sigma^*$ is regular, so any language over $\Sigma = \{0, 1\}$ is the subset of a regular language! We just saw examples of languages over Σ which are not regular.

More closure rules

Regular languages are also closed under complement and intersection.

Theorem: If language L is regular, then so is its complement, L' .

Proof:

- ▶ Proof by Kleene's theorem.
- ▶ Since L is regular, it is the language of a DFA, M , with state set Q and accept states $F \subseteq Q$.
- ▶ Construct a new DFA, M' from M , as follows.
- ▶ Swap the accept and reject states in M .
- ▶ Then M' , with accept set $Q \setminus F$ accepts all words which the old DFA, M , rejected and rejects all words which M accepted.
- ▶ M is a DFA, so there is only one path for a particular word.
- ▶ The same is true for M' , so M' is a new DFA.
- ▶ M' accepts L' .
- ▶ By Kleene's theorem, L' is regular.

(This is much easier than exhibiting a regular expression for L' .)

Regular languages are closed under intersection

Theorem: If L_1 and L_2 are regular languages, then so is $L_1 \cap L_2$.

Proof:

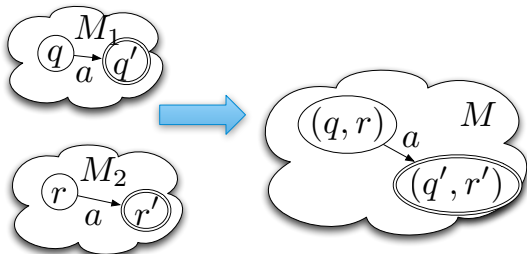
- ▶ This can be proved in lots of ways.
- ▶ One easy one: $L_1 \cap L_2 = (L'_1 \cup L'_2)'$.
(That can take a couple seconds to understand! Draw a suitable Venn diagram if it helps.)
- ▶ And L'_1 is regular (by our last theorem); so is L'_2 .
- ▶ By the definition of regular languages, so is $L'_1 \cup L'_2$.
- ▶ And again, by our closure theorem, so is $(L'_1 \cup L'_2)' = L_1 \cap L_2$.

Or, by actually building a DFA

Theorem: Given regular languages L_1 and L_2 , $L = L_1 \cap L_2$ is also regular.

Proof sketch:

- ▶ Let M_1 and M_2 be machines accepting L_1 and L_2 respectively.
- ▶ New machine M : one state for each pair of states in M_1 and M_2 .
 - ▶ If M_1 is in state q , and M_2 is in state r , then M will be in (q, r) .
 - ▶ If M_1 transitions from q to q' and M_2 from r to r' on letter a , then in M , $\delta_M((q, r), a) = (q', r')$.
 - ▶ Accept states are those that come from accept states in both machines.



Closure under reversal

Recall: w^R is the reversal of the word w .

Given a language L , let L^R be the language that consists of all of the words of L , reversed.

Theorem: If L is regular, then so is L^R .

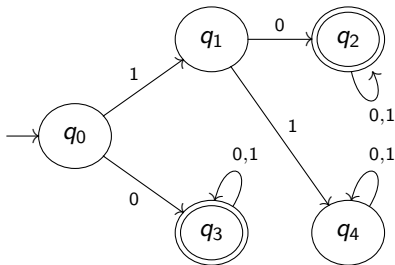
Proof sketch: Let M be a finite automaton whose language is L . Make a new finite automaton R with the same states as M , plus a new start state:

- ▶ All of the edges of R are the reversals of the edges of M .
- ▶ The sole accept state of R is the start state of M .
- ▶ The start state of R has an ε -transition to each accept state of M .

Then R reverses the automaton M : if we start at an accept state of M and work our way back to the start state of M (i.e. if M accepted x), then the new machine R accepts the word x^R , and vice versa.

Closure under reversal

Suppose we have the following DFA, M

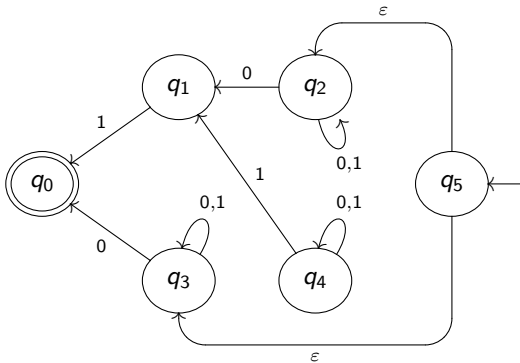


to accept the language

$$L = \{w \mid w \text{ begins with } 0 \text{ or with } 10\}.$$

Closure under reversal

The construction yields this ε -NFA, R



to accept the language

$$L = \{w \mid w \text{ ends with } 0 \text{ or with } 01\}.$$

Note that, although this ε -NFA R can be simplified, the construction is still correct.

Properly proving the reversal of a regular language is regular

We can prove this theorem by structural induction on the construction of the regular language L .

- ▶ Base cases: If $L = \emptyset, \{\varepsilon\}$ or $\{a\}$, then $L^R = L$ is regular.
- ▶ If $L = L_1 \cup L_2$ for regular languages L_1 and L_2 , then $L^R = L_1^R \cup L_2^R$, and both of these languages are regular by induction.
- ▶ If $L = L_1 L_2$ for regular languages L_1 and L_2 , then $L^R = L_2^R L_1^R$, and this is the concatenation of two regular languages and hence regular.
- ▶ If $L = L_1^*$, then $L^R = (L_1^R)^*$, since any word in L is the concatenation of a finite sequence of words of L_1 : if $w \in L$, $w = w_1 \dots w_n$, and $w^R = w_n^R \dots w_1^R$. This is a sequence of words in L_1^R , and so L^R is the Kleene closure of the (by the induction hypothesis) regular language L_1^R .

Algorithmic questions about finite automata

We do not normally create algorithms in this class. Here is an exception:
Is it possible to find algorithms for the following:

- ▶ Given a DFA M and a word x , does M accept x ?
- ▶ Given a DFA M , is $L(M)$ empty?
- ▶ Given a DFA M , is $L(M)$ infinite?
- ▶ Given two DFAs M_1 and M_2 , is $L(M_1) \cap L(M_2)$ empty?
- ▶ Given two DFAs M_1 and M_2 , is $L(M_1) \subseteq L(M_2)$?
- ▶ Given two DFAs M_1 and M_2 , is $L(M_1) = L(M_2)$?
- ▶ Given two regular expressions e_1 and e_2 , do they generate the same language?

Acceptance, empty language

- ▶ Given a DFA M and a word x , does M accept x ?
 - ▶ Just simulate the DFA.
 - ▶ (This may seem obvious. But we will not be able to do this for Turing machines.)
- ▶ Given a DFA M , is $L(M)$ empty?
More fun.
 - ▶ Suppose M has n states.
 - ▶ If M accepts any words, it must accept a word with fewer than n letters.
 - ▶ This is a consequence of the proof of the Pumping Lemma.
 - ▶ We prove it carefully on the next slide.

Acceptance, empty language

Lemma: If a DFA, M , having n states accepts any words, then it must accept a word with fewer than n letters.

Proof:

- ▶ Assume $L(M) \neq \emptyset$.
- ▶ By Kleene's Theorem, $L(M)$ is regular.
- ▶ Let $x_0 \in L(M)$ be arbitrary.
- ▶ If $|x_0| < n$, then we are finished.
- ▶ Otherwise, $|x_0| \geq n$ and by the proof of the Pumping Lemma, we can decompose $x_0 = u_0 v_0 w_0$, with $u_0 w_0 \in L(M)$ and $|v_0| \geq 1$.
- ▶ If $|u_0 w_0| < n$, then we are finished.
- ▶ Otherwise, $|u_0 w_0| \geq n$ and $u_0 w_0 \in L(M)$ and so by the proof of the Pumping Lemma, we can decompose $u_0 w_0 = u_1 v_1 w_1$, with $u_1 w_1 \in L(M)$ and $|v_1| \geq 1$.
- ▶ Continuing in this way we obtain a sequence of words in $L(M)$ having strictly decreasing lengths: $x_0, u_0 w_0, u_1 w_1, \dots, u_j w_j, \dots$
- ▶ As x_0 has finite length, after at most $|x_0| - n + 1$ steps, we will obtain a word in $L(M)$ with length $< n$. \square

Acceptance, empty language

Now, back to the question

- ▶ Given a DFA M , is $L(M)$ empty?
 - ▶ Try **every** word of length less than n (finitely many since our alphabet is finite).
 - ▶ If no short word is accepted, then by the previous Lemma, $L(M) = \emptyset$.

Is the language of an FA finite?

- ▶ Given a DFA M , is $L(M)$ infinite?

Theorem: If M is a DFA with n states, then $L(M)$ is infinite if and only if $L(M)$ includes a word x satisfying $n \leq |x| < 2n$.

Proof:

- ▶ Suppose $x \in L(M)$ and $n \leq |x| < 2n$.
- ▶ From the Pumping Lemma, x must be pumpable.
- ▶ The word $x = uvw$ can be used to generate the infinite language uv^*w , which is a subset of $L(M)$.
- ▶ So $L(M)$ is infinite.

Other half of the proof

- ▶ Other direction: Assume that $L(M)$ is infinite.
 - ▶ For a contradiction, suppose that there does not exist any $x \in L(M)$ satisfying $n \leq |x| < 2n$.
 - ▶ In other words, every word $x \in L(M)$ with length at least n must have length at least $2n$.
 - ▶ Let $x \in L(M)$ be a shortest word with length at least n (so that $|x| \geq 2n$ by the above point).
 - ▶ (If there is no $x \in L(M)$ with length at least n , then $L(M)$ is finite, which cannot happen.)
 - ▶ Decompose $x = uvw$, where $v \neq \varepsilon$ and $|uv| \leq n$, so that $1 \leq |v| \leq n$.
 - ▶ By the Pumping Lemma, uw is also in $L(M)$.
 - ▶ We have only removed at most n letters by removing v , so $|uw| \geq n$, and by construction, $|uw| < |x|$.
 - ▶ Thus uw violates the choice of x as a shortest word in $L(M)$ with length at least n .
 - ▶ This contradiction completes the proof.
- ▶ If $L(M)$ is infinite, we must have a word in $L(M)$ with length between n and $2n$.
- ▶ To see if $L(M)$ is infinite, check all words between n and $2n$ in length.
- ▶ This runs in a finite amount of time.

Disjoint languages, subset

- ▶ Given two DFAs M_1 and M_2 , is $L(M_1) \cap L(M_2)$ empty?
 - ▶ First, construct a DFA for $L(M_1) \cap L(M_2)$.
 - ▶ Then use the algorithm for testing for an empty language of an FA to see if it accepts the empty language.
- ▶ Given two DFAs M_1 and M_2 , is $L(M_1) \subseteq L(M_2)$?
 - ▶ If so, then $L(M_2)' \cap L(M_1)$ is empty.
(There is nothing in $L(M_1)$ that is not in $L(M_2)$.)
 - ▶ Build the DFA for $L(M_2)'$.
 - ▶ Use it to build the DFA for $L(M_2)' \cap L(M_1)$.
 - ▶ Use the algorithm from before to test if its language is empty!

Two FAs with the same language

- ▶ Given two DFAs M_1 and M_2 , is $L(M_1) = L(M_2)$?
 - ▶ Yes, if $L(M_1) \subseteq L(M_2)$ and $L(M_2) \subseteq L(M_1)$.
 - ▶ Use the algorithm for testing for subset twice.
- ▶ Given two regular expressions e_1 and e_2 , do they represent the same language?
 - ▶ Construct the DFAs for each regular expression, using Kleene's Theorem.
 - ▶ Then use the algorithm for testing if two FAs have the same language.

(If you like this kind of stuff, take CS 462.)

End of regular language unit

Is a DFA a decent model of a computer?

- ▶ Yes, if resources are bounded.
- ▶ Regular languages: accepted by computers with very simple access to input and very little memory
- ▶ But $\{0^i 1^i \mid i \geq 0\}$ is a simple language. And yet no DFA can recognize it.

Except:

- ▶ Real computers have finite memory.
- ▶ Suppose a computer has q bits of memory.
- ▶ The computer can only be in 2^q possible states.

Often, we do not care, because that number is so huge.

Typically, we treat computers as having infinite memory, except when it really matters.

Next: Context-free languages (Module 5)

After that: Computers with infinite memory (Module 6)