MA/CSSE 474

Theory of Computation

Regular and Non-regular Languages
Closure Properties
Pumping Theorem Intro

Languages: Regular or Not?

Recall: a*b* is regular. $\{a^n b^n : n \geq 0\}$ is not.

$\{w \in \{a, b\}^*$: every a is immediately followed by b$\}$ is regular.

$\{w \in \{a, b\}^*$: every a has a matching b somewhere$\}$ is not.

How do we

- show that a language is regular?
- show that a language is not regular?

How many languages are there?

- Consider $\Sigma = \{a\}$.
- Clearly the set of languages over {a} is infinite.
- Suppose the set of languages over {a} was countable.
- Then we can enumerate all of the languages as $L_0, L_1, \ldots$, and every language appears in the list.
- Consider $L_d = \{a^i : i \geq 0 \text{ and } a^i \notin L_i\}$.
- Does $L_d = L_i$ for any $i \geq 0$?

How Many Regular Languages?

Theorem: There is a countably infinite number of regular languages over any nonempty alphabet Σ.

Proof:

- Upper bound on number of regular languages: number of DFSMs (or regular expressions).
- Lower bound on number of regular languages:

  {a},{aa},{aaa},{aaaa},{aaaaa},{aaaaaa},…

  are all regular. That set is countably infinite.

Q1

Are Regular or Nonregular Languages More Common?

There is a countably infinite number of regular languages.

There is an uncountably infinite number of languages over any nonempty alphabet Σ.

So there are *many* more nonregular languages than there are regular ones.

Showing that a Language is Regular

Theorem: Every finite language L is regular.

Proof: If *L* is the empty set, then it is defined by the regular expression $\varnothing$ and so is regular.

If L is a nonempty finite language, composed of the strings $s_1, s_2, \ldots s_n$ for some positive integer *n*, then it is defined by the regular expression:

$s_1 \cup s_2 \cup \ldots \cup s_n$

So L is regular.

Q2

Finiteness - Theoretical vs. Practical

Any finite language is regular. The size of the language doesn't matter.

Parity Checking ⟷ Soc. Sec. # Checking

But, from an implementation point of view, it very well may.

When is an FSM a good way to encode the facts about a language?

FSM's are good at looking for repeating patterns. They don't bring much to the table when the language is just a set of unrelated strings.

Regular Does Not Always Mean Tractable

Let $\Sigma = \{12, 13, 21, 23, 31, 32\}$.

Let L be the language of strings that correspond to successful move sequences. The shortest string in L has length $2^{64} - 1$.

There is an FSM that accepts L:

Q3

To Show that a Language L is Regular

We can do any of the following:

Construct a DFSM that accepts L.

Construct a NDFSM that accepts L.

Construct a regular expression that defines L.

Construct a regular grammar that generates L.

Show that there are finitely many equivalence classes under $\approx_L$.

Show that L is finite.

Use one of the closure properties.

Closure Properties of Regular Languages

- Union
- Concatenation
- Kleene star
- Complement
- Intersection
- Difference
- Reverse
- Letter substitution

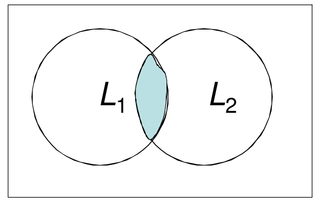
The first three are easy: definition of regular expressions.

We already did Complement and Reverse.

We'll do details of some of the others.

Closure of Regular Languages Under Intersection

$L_1 \cap L_2 =$



Write this in terms of operations we have already proved closure for:

- Union
- Concatenation
- Kleene star
- Complementation

Closure of Regular Languages Under Difference

$L_1 - L_2 = L_1 \cap \neg L_2$

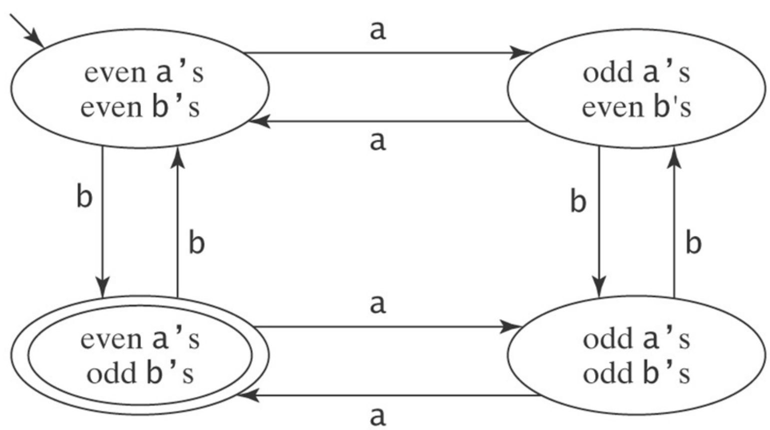
Divide-and-Conquer

Let $L = \{w \in \{a, b\}^* :$ *w* contains an even number of a's and an odd number of b's and all a's come in runs of three}.

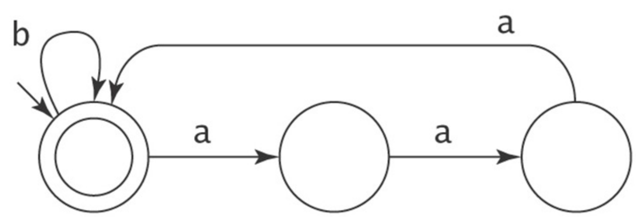
$L = L_1 \cap L_2$, where:

- $L_1 = \{w \in \{a, b\}^* :$ *w* contains an even number of a's and an odd number of b's}, and
- $L_2 = \{w \in \{a, b\}^* :$ all a's come in runs of three}

L_1 is Regular



L_2 is Regular



Don't Try to Use Closure Backwards

One Closure Theorem:

If L_1 and L_2 are regular, then so is

$\underline{L} = \underline{L_1} \cap \underline{L_2}$

↑

But if L is regular, what can we say about L_1 and L_2?

$\underline{L} = L_1 \cap L_2$

ab = ab ∩ (a ∪ b)* (they are regular)

ab = ab ∩ {$a^n b^n$, $n \geq 0$} (they may not be regular)

Q4

Don't Try to Use Closure Backwards

Another Closure Theorem:

If L_1 and L_2 are regular, then so is

$$L = L_1 \; L_2$$

But if L_2 is not regular, what can we say about L?

$$L = \quad L_1 \qquad L_2$$

$\{\mathtt{aba}^n\mathtt{b}^n : n \geq 0\} = \{\mathtt{ab}\}\ \{\mathtt{a}^n\mathtt{b}^n : n \geq 0\}$

$\text{L}(\mathtt{aaa}^*) = \{\mathtt{a}\}^* \{\mathtt{a}^p : p \text{ is prime}\}$

Showing that a Language is Not Regular

Every regular language can be accepted by some FSM.

It can only use a finite amount of memory to record essential properties.

Example:

$\{\mathtt{a}^n\mathtt{b}^n, n \geq 0\}$ is not regular

Showing that a Language is Not Regular

The only way to generate/accept an infinite language with a finite description is to use:

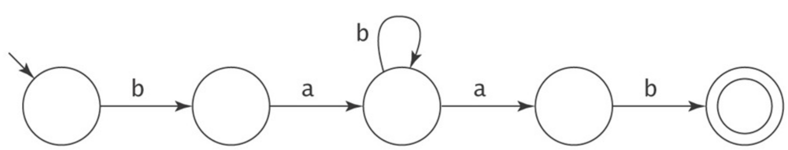
- Kleene star (in regular expressions), or
- cycles (in automata).

This forces some kind of simple repetitive cycle within the strings.

Example:
`ab*a` generates `aba`, `abba`, `abbba`, `abbbba`, etc.

Example:
{a^n : $n \geq 1$ is a prime number} is not regular.

Exploiting the Repetitive Property



If an FSM with n states accepts at least one string of length $\geq n$, how many strings does it accept?

L = `bab*ab`

`b a` `b` `b b b a b`
x y z

xy^*z must be in L.

So L includes: `baab`, `babab`, `babbab`, `babbbbbbbbbbab`

Q5

Theorem – Long Strings

Theorem: Let $M = (K, \Sigma, \delta, s, A)$ be any DFSM. If M accepts any string of length $|K|$ or greater, then that string will force M to visit some state more than once (thus traversing at least one loop).

Proof: M must start in one of its states.
Each time it reads an input character, it visits some state. So, in processing a string of length n, M does a total of $n + 1$ state visits.

If $n+1 > |K|$, then, by the pigeonhole principle, some state must get more than one visit.

So, if $n \geq |K|$, then M must visit at least one state more than once.

The Pumping Theorem* for Regular Languages

If L is regular, then every long string in L is "pumpable".
Formally, if L is regular, then

So, $\exists k \geq 1$ such that
($\forall$ strings $w \in L$, where $|w| \geq k$
($\exists$ x, y, z ($w = xyz$,
$|xy| \leq k$,
$y \neq \varepsilon$, and
$\forall q \geq 0$ (xy^qz is in L))))

* a.k.a. "the pumping lemma".
We will use the terms interchangeably.

Q6-7

Using The Pumping Theorem to show that L is not Regular:

We use the contrapositive of the theorem:
If some long enough string in L is not "pumpable", then L is not regular.

What we need to show in order to show L non-regular:

$\forall k \geq 1$
($\exists$ string $w \in L$, where $|w| \geq k$
($\forall$ x, y, z ($w = xyz$,
$|xy| \leq k$,
$y \neq \varepsilon$, and
$\exists$ $q \geq 0$ (xy^qz is not in L)))).

Before next class: Be sure that you are convinced that this really is the negation of the conclusion of the pumping theorem.

A way to think of it: adversary argument (following J.E. Hopcroft and J.D.Ullman)

1. Choose the language L you want to prove non-regular.
2. The "adversary" picks k, the constant mentioned in the theorem. We must be prepared for any positive integer to be picked, but once it is chosen, the adversary cannot change it.
3. We select a string $w \in L$ that cannot be "pumped".
4. The adversary breaks w into w=xyz, subject to the constraints $|xy| \leq k$ *and* $y \neq \varepsilon$. Our choice of w must take into account that any such x and y can be chosen.
5. All we must do is produce a single number $q \geq 0$ such that $xy^qz \notin L$.

Note carefully what we get to choose and what we do not get to choose.

Example: {$a^n b^n$: $n \geq 0$} is not Regular

k is the number from the Pumping Theorem.
We don't get to choose it.

Choose *w* to be $a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$ ("long enough").

```
             1                      2
a a a a a … a a a a a | b b  b b  … b b b b b b
             x        |  y          z
```

Adversary chooses *x*, *y*, *z* with the required properties:

$|xy| \leq k$,

$y \neq \varepsilon$,

$\forall q \geq 0$ (xy^qz is in L).

Three cases to consider:

- *y* entirely in region 1:
- *y* partly in region 1 and partly in 2:
- *y* entirely in region 2:

A Complete Proof

We prove that $L = \{a^n b^n: n \geq 0\}$ is not regular

If *L* were regular, then there would exist some *k* such that any string *w* where $|w| \geq k$ must satisfy the conditions of the theorem. Let $w = a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$. Since $|w| \geq k$, *w* must satisfy the conditions of the pumping theorem. So, for some *x*, *y*, and *z*, $w = xyz$, $|xy| \leq k$, $y \neq \varepsilon$, and $\forall q \geq 0$, xy^qz is in *L*. We show that no such *x*, *y*, and *z* exist. There are 3 cases for where *y* could occur: We divide *w* into two regions:

```
aaaaa…..aaaaaa   | bbbbb…..bbbbbb
        1        |      2
```

So *y* can fall in:

- (1): $y = a^p$ for some *p*. Since $y \neq \varepsilon$, *p* must be greater than 0. Let $q = 2$. The resulting string is $a^{k+p}b^k$. But this string is not in *L*, since it has more a's than b's.
- (2): $y = b^p$ for some *p*. Since $y \neq \varepsilon$, *p* must be greater than 0. Let $q = 2$. The resulting string is $a^k b^{k+p}$. But this string is not in *L*, since it has more b's than a's.
- (1, 2): $y = a^p b^r$ for some non-zero *p* and *r*. Let $q = 2$. The resulting string will have interleaved a's and b's, and so is not in *L*.

There exists one long string in *L* for which no pumpable *x*, *y*, *z* exist. So *L* is not regular.

What You Need to Write

We prove that $L = \{a^n b^n : n \geq 0\}$ is not regular

Let $w = a^{\lceil k/2 \rceil} b^{\lceil k/2 \rceil}$. (If not completely obvious, as in this case, show that w is in fact in L.)

There are 3 cases for where y could occur:

```
aaaaa.....aaaaaa   | bbbbb.....bbbbbb
       1           |         2
```

So y can fall in:

- (1): $y = a^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^{k+p}b^k$. But this string is not in L, since it has more a's than b's. .
- (2): $y = b^p$ for some p. Since $y \neq \varepsilon$, p must be greater than 0. Let $q = 2$. The resulting string is $a^k b^{k+p}$. But this string is not in L, since it has more b's than a's.
- (1, 2): $y = a^p b^r$ for some non-zero p and r. Let $q = 2$. The resulting string will have interleaved a's and b's, and so is not in L.

Thus L is not regular.

A better choice for w

Second try. A choice of w that makes it easier:

Choose w to be $a^k b^k$

(We get to choose any w whose length is *at least* k).

```
          1                  |        2
a a a a a ... a a  a a  a | b b b b  ... b b b b b b
        x           y      |           z
```

We show that there is no x, y, z with the required properties:

$|xy| \leq k$,

$y \neq \varepsilon$,

$\forall\ q \geq 0$ ($xy^q z$ is in L).

Since $|xy| \leq k$, y must be in region 1. So $y = a^p$ for some $p \geq 1$. Let $q = 2$, producing:

$$a^{k+p}b^k$$

which $\notin L$, since it has more a's than b's.

We only have to find **one** q that takes us outside of L.

Recap: Using the Pumping Theorem

If L is regular, then every "long" string in L is pumpable.

To show that L is not regular, we find one that isn't.

To use the Pumping Theorem to show that a language L is not regular, we must:

1. Choose a string w where $|w| \geq k$. Since we do not know what k is, we must state w in terms of k.
2. Divide the possibilities for y into a set of equivalence classes that can be considered together.
3. For each such class of possible y values where $|xy| \leq k$ and $y \neq \varepsilon$:

   Choose a value for q such that xy^qz is not in L.