Properties of Regular Languages

Lecture 5
Andrew Tolmach
(with material from Andrew P Black and Tim Sheard)
Closure properties

1. The union of two regular languages is regular
2. The concatenation of two regular languages is regular
3. The Kleene-closure (*) of a regular language is regular
4. The complement of a regular language is regular
5. The intersection of two regular languages is regular

Which of these properties do we have to prove?
Proof of Closure under Complement
Proof of Closure under Complement

1. Start with an DFA $M$ that accepts $L$. 
Proof of Closure under Complement

1. Start with an DFA $M$ that accepts $L$.

2. Modify $M$ to accept when it used to reject, and to reject when it use to accept.
   
   • That is, change all the final states into non-final states, and all the non-final states into final states.
Proof of Closure under Complement

1. Start with an DFA M that accepts L.

2. Modify M to accept when it used to reject, and to reject when it use to accept.
   • That is, change all the final states into non-final states, and all the non-final states into final states.

3. The new machine accepts the complement of L.
Proof of Closure under Complement

1. Start with an DFA $M$ that accepts $L$.

2. Modify $M$ to accept when it used to reject, and to reject when it used to accept.
   - That is, change all the final states into non-final states, and all the non-final states into final states.

3. The new machine accepts the complement of $L$.
   - Query: Would this proof work if we started with an NFA instead?
Proof of Closure under Intersection

• Recall the product construction by which we proved that given DFA’s $M_1$ and $M_2$, we can always construct a machine $M$ that recognizes $L(M_1) \cup L(M_2)$.

• In that construction, each state of $M$ corresponds to a pair of states $(q_1,q_2)$, with $q_1 \in Q_1$ and $q_2 \in Q_2$.

• The final states of $M$ are those for which either $q_1 \in F_1$ or $q_2 \in F_2$ (or both)

• To build a machine that recognizes $L(M_1) \cap L(M_2)$, we just make the final states be those for which both $q_1 \in F_1$ and $q_2 \in F_2$
Another, quicker proof
Another, quicker proof

\[ L \cap M = \overline{L} \cup \overline{M} \]
Other Closure Properties

• The regular languages stay closed under a remarkable variety of operations
  • Difference
  • Reversal (see IALC)
  • Shuffle
  • DROP-OUT(L) =
    \{xz \mid xyz \in L, \text{ where } x,z \in \Sigma^*, y \in \Sigma\}
  • A/B = \{w \mid wx \in A \text{ for some } x \in B\} \text{ when A is a regular language and B is any language}
Limits of finite state machines

• Consider the language
  \[ L = \{0^k1^k \mid k=0,1,2, \ldots\} \]

• Is this language regular?

• If so, there is some DFA that recognizes it

• Intuitively, this should not be possible

  ▶ such a machine would have to keep track of an **arbitrarily large** number \( k \)

  ▶ but DFAs only have a finite number of states!
Long strings need loops

- But some DFA’s certainly can recognize arbitrarily long strings
- How? By entering some state(s) more than once.
  - i.e. by going into a loop
Consequences of loops

Consider this DFA. The input string $01011$ is accepted after an execution that goes through the state sequence $s \rightarrow p \rightarrow q \rightarrow p \rightarrow q \rightarrow r$. This path contains a loop (corresponding to the substring $01$) that starts and ends at $p$.

There are two simple ways of modifying this path without changing its beginning and ending states:
(1) delete the loop from the path, showing that 011 is accepted

(2) instead of going around the loop once, do it several times, showing that 0101011, 010101011, ... are accepted

In general, we see that all strings of the form $0(10)^i11$ (where $i \geq 0$) are accepted.
Long paths *must* contain a loop

- Suppose a DFA has \( n \) states but it accepts a string of length \( \geq n \)
  - In so doing, it visits at least \( n+1 \) states
  - Therefore it must visit some state twice
    - This is a consequence of the *pigeonhole principle*

- Thus, every path of length \( n \) or longer must contain a loop
Loops make “pumps”

• Suppose $L$ is a regular language, and $w=xuy$ is a string in $L$, where $u$ is non-empty.

• We say that $u$ is a **pump** in $w$ if all strings $xu^iy$ ($i \geq 0$) belong to $L$.

  • So, $xy$, $xuy$, $xuuy$, $xuuuy$, … are all in $L$
    • When we increase $i$ we’re “pumping up”
    • When we decrease $i$ to 0 we’re “pumping down”
The Pumping Lemma

• For every regular language $L$, there is a number $n$ (called the **pumping length** of $L$) such that every string in $L$ of length at least $n$ contains a “pump.”
  
  ▶ In fact, it contains a pump in the first $n$ symbols.
  ▶ In practice it doesn’t matter exactly what $n$ is — just that it exists.

• **Formally:** If $L$ is regular, then $\exists n$ such that if $w \in L$ and $|w| \geq n$ then we can write $w$ as $xyz$ where:
  
  1. $xy^iz \in L$ for every $i \geq 0$ (y is a “pump”)
  2. $y \neq \varepsilon$ (the “pump” is non-empty)
  3. $|xy| \leq n$ (it appears in the first $n$ symbols)
Proof Idea

• Let M be a DFA that recognizes L
• Choose $n$ to be the number of states in M
• Choose any $w \in L$ such that $|w| \geq n$
  ▶ What if there is none?
• What sequence of transitions does M make to accept w?
What happens when $M$ accepts $w$?

- it starts in the start state $q_0$
- it moves through a series of $|w|$ other states,
- ending in a final state, say $q_f$

Since $|w| \geq n$, this path must have at least one repeated state, and hence a loop

- say the first state to be entered twice is $q_j$
• Label the pieces of w as in the diagram

• Then xz, xyz, xyyz, xyyyz, etc. are all in L
  ▶ |y| > 0 or else there would be no loop
  ▶ |xy| ≤ n, or else we could have found a repeated state sooner
Using the Pumping Lemma

• The Pumping Lemma says that regular languages follow a very restrictive pattern
  ▶ If L is regular, any sufficiently long string in L can be “pumped” to produce many other strings in L

• We can use it to show a language is not regular by showing that it doesn’t follow the pattern
  ▶ We exhibit an arbitrarily long string in L which, when “pumped,” produces some string not in L
Informal proof example 1

• Let’s argue that $L = \{0^k1^k \mid k=0,1,2, \ldots\}$ is not regular.

• Why? Because there is an arbitrarily long string in $L$ that, when pumped, produces a string not in $L$.

• In fact, that’s true of every string in $L$:
  ▶ Consider $0^n1^n$ for any $n$ and suppose it has a “pump”
  ▶ If the pump is all 0’s, pumping will change the number of 0’s but not the number of 1’s, so result is not in $L$.
  ▶ If the pump is all 1’s...(similarly)...result is not in $L$.
  ▶ If the pump is of the form $0^+1^+$, pumping up produces a string not of the form $0^*1^*$, so result is not in $L$. 
We prove that \( L = \{0^k1^k \mid k=0,1,2, \ldots \} \) is not regular.

- **Assume** \( L \) is regular. We’ll show that this leads to a contradiction.
- Let the pumping length of \( L \) be \( n \).
- Take \( w=0^n1^n \). Then certainly \(|w| \geq n\).
- So, by the pumping lemma, we can write \( w \) as \( xyz \) with
  - \( xy^iz \in L \) for \( i \geq 0 \), \( |y| > 0 \), and \(|xy| \leq n\).
- There are three possibilities for \( y \):
  1. \( y = 0^m \) for some \( m > 0 \). But then, taking \( i = 2 \), \( xyyz = 0^{n+m}1^n \in L \).
  2. \( y = 1^m \) for some \( m > 0 \) ...(by similar argument)... \( 0^n1^{m+n} \in L \).
  3. \( y = 0^q1^r \) for some \( q,r > 0 \). But then, taking \( i = 2 \), \( xyyz = 0^n1^r0^q1^n \in L \).
- But each of these cases leads to a contradiction with the definition of \( L \). Hence our **assumption** that \( L \) is regular must have been wrong. So \( L \) is **not** regular. QED.
We prove that \( L = \{0^k1^k \mid k=0,1,2, \ldots\} \) is not regular.

- **Assume** \( L \) is regular. We’ll show that this leads to a contradiction.
- Let the pumping length of \( L \) be \( n \).
- Take \( w=0^n1^n \). Then certainly \( |w| \geq n \).
- So, by the pumping lemma, we can write \( w \) as \( xyz \) with

  - \( xy^iz \in L \) for \( i \geq 0 \), \( |y| > 0 \), and \( |xy| \leq n \).
  
- Since \( |xy|: \)

  - \( y = 0^m \) for some \( m > 0 \). But then, taking \( i = 2 \), \( xyyz = 0^{n+m}1^n \in L \).

- But this leads to a contradiction with the definition of \( L \). Hence, our **assumption** that \( L \) is regular must have been wrong. So \( L \) is not regular. 

QED.
Quick Logic Review

• Suppose we know that $A \Rightarrow B$.

• The **contrapositive** statement is $\neg B \Rightarrow \neg A$.
  
  ‣ The contrapositive of a true fact is always automatically true too: $A \Rightarrow B \equiv \neg B \Rightarrow \neg A$

  ‣ In proof by contradiction, we show that $A \land \neg B \Rightarrow$ falsehood, and conclude $A \Rightarrow B$.

• Also, recall how negation interacts with quantification:
  
  ‣ $\neg(\forall x. P(x)) \iff \exists x. \neg P(x)$
  
  ‣ $\neg(\exists x. P(x)) \iff \forall x. \neg P(x)$
Pumping lemma contrapositive

**Pumping Lemma** says:

\((L \text{ is regular})\)

\(\Rightarrow\)

\((\exists n, \forall w \in L \text{ where } |w| \geq n, \exists x, y, z \text{ where } w = xyz \text{ and } y \neq \epsilon \text{ and } |xy| \leq n, \forall i \geq 0, xy^iz \in L).\)

**Contrapositive** says:

\(\neg (\exists n, \forall w \in L \text{ where } |w| \geq n, \exists x, y, z \text{ where } w = xyz \text{ and } y \neq \epsilon \text{ and } |xy| \leq n, \forall i \geq 0, xy^iz \in L)\)

\(\Rightarrow\)

\(\neg (L \text{ is regular}).\)
Contrapositive, rewritten

**Contrapositive** says:
\[ \neg( \exists n, \forall w \in L \text{ where } |w| \geq n, \exists x, y, z \text{ where } w = xyz \text{ and } y \neq \epsilon \text{ and } |xy| \leq n, \forall i \geq 0, xy^i z \in L) \] \[ \Rightarrow \neg(L \text{ is regular}). \]

Equivalently, **Contrapositive** says:
\[ (\forall n, \exists w \in L \text{ where } |w| \geq n, \forall x, y, z \text{ where } w = xyz \text{ and } y \neq \epsilon \text{ and } |xy| \leq n, \exists i \geq 0, xy^i z \not\in L) \] \[ \Rightarrow (L \text{ is not regular}). \]
Example 2

Show that
\[ L = \{ w \in \{0,1\}^* \mid w \text{ contains an equal number of 0s and 1s} \} \]
is not regular.

We apply the contrapositive of the Pumping Lemma:

• For any \( n \), choose \( w = 0^n1^n \). Then \( |w| \geq n \).

• For any \( x, y, z \) where \( w = xyz \), \( |y| > 0 \) and \( |xy| \leq n \), it must be the case that \( y = 0^m \) for some \( m > 0 \). (Why?)

• Now choose \( i = 2 \). Then \( xy^iz = xyyz = 0^{n+m}1^n \) which is not in \( L \)

Hence \( L \) is not regular.

• Note that in this proof, choice of \( w \) matters!
Example 2a
Example 2a

Here’s another way to show that
\[ L = \{ w \in \{0,1\}^* \mid w \text{ contains an equal number of } 0\text{s and } 1\text{s}\} \]
is not regular.
Example 2a

Here’s another way to show that

\[ L = \{ w \in \{0,1\}^* \mid w \text{ contains an equal number of 0s and 1s} \} \]

is not regular.

Let \( M = 0^*1^* \), a regular language
Example 2a

Here’s another way to show that

\[ L = \{w \in \{0,1\}^* \mid w \text{ contains an equal number of 0s and 1s}\} \]

is not regular.

Let \( M = 0^*1^* \), a regular language

- Then \( L \cap M = \{0^n1^n \mid n \geq 0\} \)
Example 2a

Here’s another way to show that

\[ L = \{ w \in \{0,1\}^* \mid w \text{ contains an equal number of } 0\text{s and } 1\text{s} \} \]

is not regular.

Let \( M = 0^*1^* \), a regular language

- Then \( L \cap M = \{ 0^n1^n \mid n \geq 0 \} \)
- If \( L \) were regular, then \( L \cap M \) would be regular. But it’s not, so neither is \( L \).
It’s a game!

• We pick a language \( L \) to prove non-regular

• Our opponent picks \( n \), but doesn’t tell us what it is

• We give \( w \) of length \( \geq n \) (\( w \) can depend on \( n \))

  ▶ This is a key move! It requires *skill and ingenuity*: we must find \( w \) that will work for us in the last move no matter how our opponent plays

• Our opponent factors \( w \) into \( xyz \), obeying only the constraints \(|y| > 0\) and \(|xy| \leq n\).

• We show that for some \( i \), \( xy^i z \) is not in \( L \).

  ▶ Sometimes picking \( i \) also requires cleverness.
Example 3

Show that $L = \{ uu | u \in \{a,b\}^* \}$ is not regular.

• Suppose it were and let $n$ be its pumping length.

• Then we choose $w = a^n b a^n b$, which clearly has length greater than $n$.

• The opponent has to divide $w$ into $xyz$, where $|xy| \leq n$ and $|y| > 0$. But then $y$ must have the form $a^m$ for some $m > 0$.

• We choose $i = 0$. That has the effect of dropping $m$ as. So $a^{n-m} b a^n b$ must be in $L$. But it isn’t, so we “win”: $L$ is not regular.

• Question: if we choose $w = a^n a^n$, the opponent has a chance to win. How?
Example 4

- We claim that \( L = \{1^p \mid p \text{ is prime}\} \) isn’t regular
- Suppose it were, with pumping length \( n \).
- We choose \( w = 1^p \) for any \( p \geq n+2 \).
  - Such a \( p \) must exist, because there are an infinity of primes
- The opponent picks \( x=1^q, y=1^r, z = 1^s \) where \( r > 0, q+r \leq n \), and \( q+r+s = p \). (Note that the opponent has no choice here.)
- We cleverly pick \( i = p-r \). Then \( xy^{p-r}z = 1^m \in L \), where \( m = q+(p-r)r + s \). So \( m \) must be prime.
- But \( m = (q+s) + (p-r)r = (p-r) + (p-r)r = (r+1)(p-r) \)
  - Moreover \( r+1 > 1 \) (why?) and \( p-r > 1 \) (why?)
  - So \( m \) is the product of two integers each > 1, and therefore not prime!
Some Important DFA Facts
— that we won’t study

• There is an algorithm to convert any DFA $M$ to a **minimum-state** DFA recognizing $L(M)$.

• The minimum-state DFA is **unique** up to renaming of states.

• There is thus an algorithm to determine whether two DFA’s recognize the same language.