CSE-217: Theory of Computation

Non Regular Language

Md Jakaria

Lecturer
Department of Computer Science and Engineering
Military Institute of Science and Technology

August 27, 2019

NONREGULAR LANGUAGE

NONREGULAR LANGUAGE

1. To understand the power of finite automata, one must also understand their limitations.

NONREGULAR LANGUAGE

1. To understand the power of finite automata, one must also understand their limitations.
2. Let's take the language $B = \{0^n 1^n | n \geq 0\}$.

NONREGULAR LANGUAGE

Example

Consider two languages over alphabet $\Sigma = \{0, 1\}$:

C = {w| w has an equal number of 0's and 1's}

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0 belong
- 1

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0 belong
- 1 belong
- 01

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0 belong
- 1 belong
- 01 not belong
- 10

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0 belong
- 1 belong
- 01 not belong
- 10 not belong
- 010

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0 belong
- 1 belong
- 01 not belong
- 10 not belong
- 010 belong
- 101

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0 belong
- 1 belong
- 01 not belong
- 10 not belong
- 010 belong
- 101 belong
- 0110

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0 belong
- 1 belong
- 01 not belong
- 10 not belong
- 010 belong
- 101 belong
- 0110 belong
- 01100

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- ϵ belong
- 0 belong
- 1 belong
- 01 not belong
- 10 not belong
- 010 belong
- 101 belong
- 0110 belong
- 01100 belong
- 1101110011

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

ϵ	belong
0	belong
1	belong
01	not belong
10	not belong
010	belong
101	belong
0110	belong
01100	belong
1101110011	belong

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- 110

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- 110 not belong
- 11010

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- 110 not belong
- 11010 not belong
- w should toggle between 0 and 1 an equal number of times

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- 110 not belong
- 11010 not belong
- w should toggle between 0 and 1 an equal number of times

D = {w | w starts and ends with same symbol }.

NONREGULAR LANGUAGE

D = {w| w has an equal number of occurrences of 01 and 10 as substrings}.

- 110 not belong
- 11010 not belong
- w should toggle between 0 and 1 an equal number of times

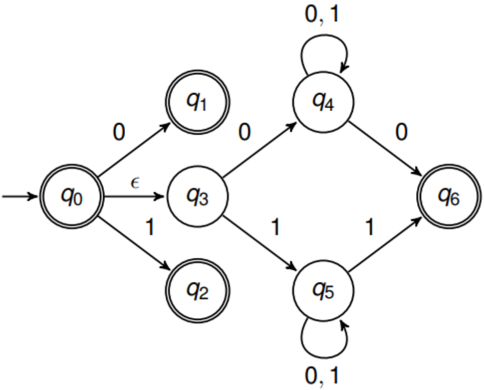
D = {w | w starts and ends with same symbol }.

$$\epsilon \cup 0 \cup 1 \cup 0\Sigma^*1 \cup 1\Sigma^*1$$

NONREGULAR LANGUAGE

D = {w | w starts and ends with same symbol }.

$$\epsilon \cup 0 \cup 1 \cup 0\Sigma^*1 \cup 1\Sigma^*1$$



THE PUMPING LEMMA FOR REGULAR LANGUAGES

Pumping Lemma

1. The technique for proving nonregularity stems from a theorem called **the pumping lemma.**
2. This theorem states that all regular languages have a special property.
3. If a language does not have this property, we are guaranteed that it is not regular.

The Pumping Lemma for Regular Languages

Sipser, 1.4, p-77

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

- $|s|$ represents the length of string s.
- y^i means that i copies of y are concatenated together.
- y^0 equals ϵ.

The Pumping Lemma for Regular Languages

Sipser, 1.4, p-77

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

- When *s* is divided into *xyz*, either *x* or *z* may be ϵ.
- But condition 2 says that $y \neq \epsilon$.
- Without condition 2 the theorem would be trivially true.

The Pumping Lemma for Regular Languages

Sipser, 1.4, p-77

THEOREM 1.70

Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

- Condition 3 states that the pieces x and y together have length at most p.
- It is an extra technical condition that we occasionally find useful when proving certain languages to be nonregular.

The Pumping Lemma for Regular Languages — *continued*

Sipser, 1.4, p-77

PROOF IDEA

- Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA that recognizes A.
- We assign the pumping length p to be the number of states of M.
- We show that any string s in A of length at least p may be broken into the three pieces xyz, satisfying our three conditions.
- What if no strings in A are of length at least p?
- Then our task is even easier because the theorem becomes vacuously true.
- Obviously the three conditions hold for all strings of length at least p if there aren't any such strings.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

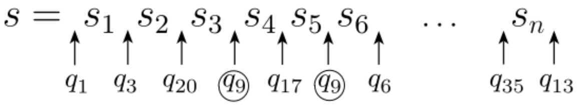


FIGURE 1.71
Example showing state q_9 repeating when M reads s

- If s in A has length at least p, consider the sequence of states that M goes through when computing with input s.
- It starts with q_1 the start state, then goes to, say, q_3, then, say, q_{20}, then q_9, and so on, until it reaches the end of s in state q_{13}.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

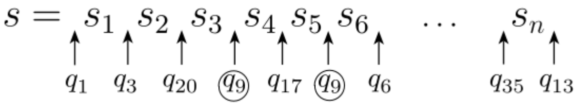


FIGURE 1.71
Example showing state q_9 repeating when M reads s

- With s in A, we know that M accepts s, so q_{13} is an accept state.
- If we let n be the length of s, the sequence of states $q_1, q_3, q_{20}, q_9, \ldots, q_{13}$ has length $n + 1$.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

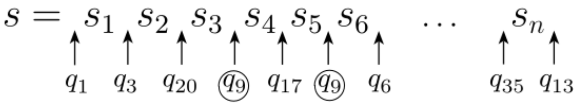


FIGURE 1.71
Example showing state q_9 repeating when M reads s

- Because *n* is at least *p*, we know that $n+1$ is greater than *p*, the number of states of *M*.
- Therefore, the sequence must contain a repeated state.
- This result is an example of the pigeonhole principle.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

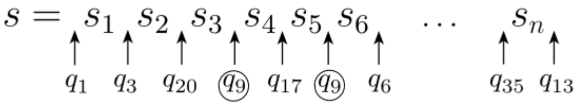


FIGURE 1.71

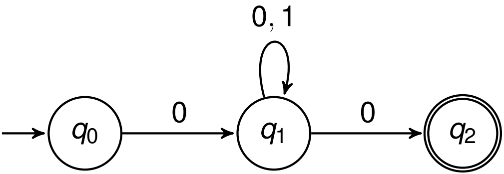
Example showing state q_9 repeating when M reads s

- State q_9 is the one that repeats.

The Pumping Lemma for Regular Languages

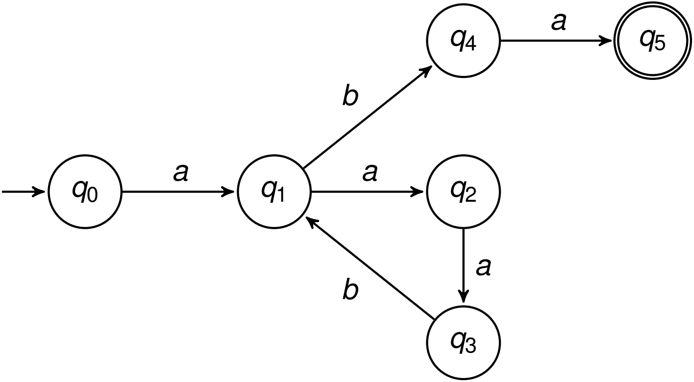
$L = \{w \mid w \text{ starts and ends with } 0, |w| \geq 2\}$

$L = 0\Sigma^*0$



The Pumping Lemma for Regular Languages

$L = a(aab)^*ba$



The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

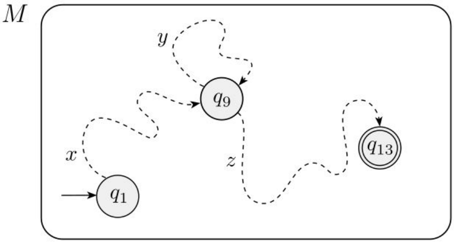


FIGURE 1.72
Example showing how the strings x, y, and z affect M

- Piece x is the part of s appearing before q_9.
- Piece y is the part between the two appearances of q_9.
- Piece z is the remaining part of s, coming after the second occurrence of q_9.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

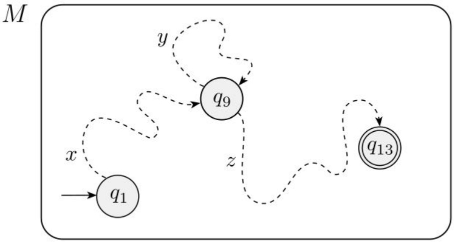


FIGURE 1.72
Example showing how the strings x, y, and z affect M

- x takes M from the state q_1 to q_9.
- y takes M from q_9 back to q_9.
- z takes M from q_9 to the accept state q_{13}.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

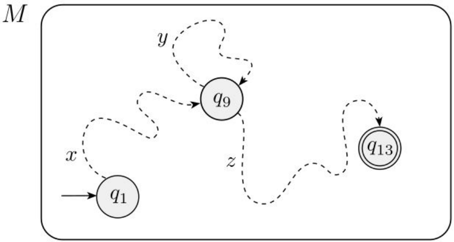


FIGURE 1.72
Example showing how the strings x, y, and z affect M

- Suppose that we run M on input $xyyz$.
- We know that x takes M from q_1 to q_9.

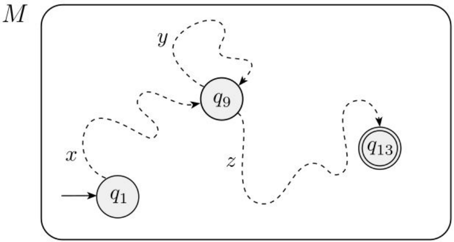


FIGURE 1.72
Example showing how the strings x, y, and z affect M

- Then the first y takes it from q_9 back to q_9 , as does the second y.
- Then z takes it to q_{13}.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

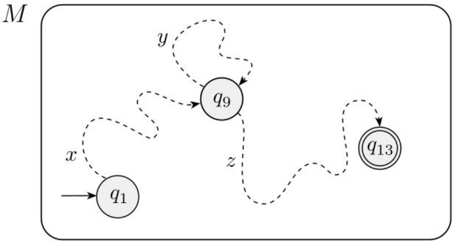


FIGURE 1.72
Example showing how the strings x, y, and z affect M

- With q_{13} being an accept state, M accepts input $xyyz$.
- Similarly, it will accept xy^iz for any $i > 0$.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

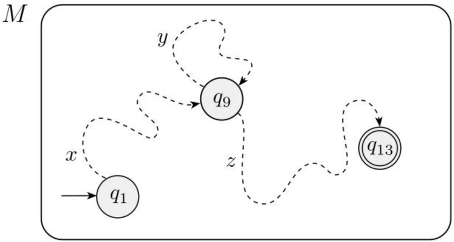


FIGURE 1.72
Example showing how the strings x, y, and z affect M

- For the case $i = 0$, $xy^i z = xz$, which is accepted for similar reasons.
- That establishes condition 1.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

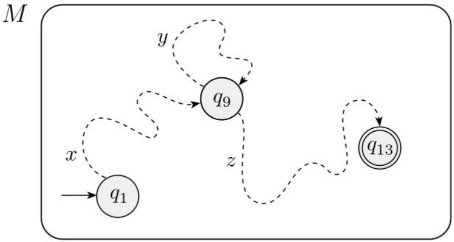


FIGURE 1.72
Example showing how the strings x, y, and z affect M

- Checking condition 2, we see that $|y| > 0$, as it was the part of s that occurred between two different occurrences of state q_9.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

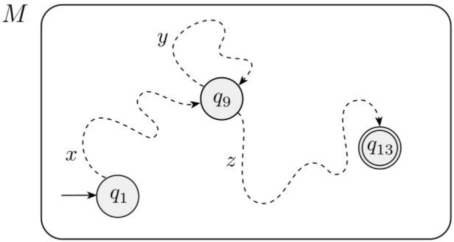


FIGURE 1.72
Example showing how the strings x, y, and z affect M

- In order to get condition 3, we make sure that q_9 is the first repetition in the sequence.
- By the pigeonhole principle, the first $p + 1$ states in the sequence must contain a repetition.
- Therefore, $|xy| \leq p$.

PROOF

- Let $M = (Q, \Sigma, \delta, q_1, F)$ be a DFA recognizing A and p be the number of states of M.
- Let $s = s_1 s_2 \dots s_n$ be a string in A of length n, where $n \geq p$.
- Let $r_1, r_2, \dots, r_{n+1}$ be the sequence of states that M enters while processing s.
- So $r_{i+1} = \delta(r_i, s_i)$ for $1 \geq i \geq n$.
- This sequence has length $n + 1$, which is at least $p + 1$.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-77

- Among the first $p+1$ elements in the sequence, two must be the same state.
- By the pigeonhole principle. We call the first of these r_j and the second r_ℓ.
- Because r_ℓ occurs among the first $p+1$ places in a sequence starting at r_1, we have $\ell \geq p+1$.

The Pumping Lemma... — *continued*

Sipser, 1.4, p-77

- Let $x = s_1 \ldots s_{j-1}$.
- $y = s_j \ldots s_{\ell-1}$.
- $z = s_\ell \ldots s_n$.

The Pumping Lemma... — *continued*

Sipser, 1.4, p-77

- Let $x = s_1 \dots s_{j-1}$.
- $y = s_j \dots s_{\ell-1}$.
- $z = s_\ell \dots s_n$.
- x takes M from r_1 to r_j.
- y takes M from r_j to r_j.
- z takes M from r_j to r_{n+1}, which is an accept state, M must accept xy^iz for $i \geq 0$.

The Pumping Lemma… — *continued*

Sipser, 1.4, p-77

- We know that $j \neq \ell$, so $|y| > 0$.
- $\ell \geq p + 1$, so $|y| > 0$.
- $\ell \geq p + 1$, so $|xy| \geq p$.
- Thus we have satisfied all conditions of the pumping lemma.

The Pumping Lemma… — *continued*

Sipser, 1.4, p-80

- To use the pumping lemma to prove that a language B is not regular, first assume that B is regular in order to obtain a contradiction.
- Then use the pumping lemma to guarantee the existence of a pumping length p such that all strings of length p or greater in B can be pumped.

- Next, find a string s in B that has length p or greater but that cannot be pumped.

- Finally, demonstrate that s cannot be pumped by considering all ways of dividing s into x, y, and z (taking condition 3 of the pumping lemma into account if convenient).
- For each such division, find a value i where $xy^i z \notin B$.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-80

- This final step often involves grouping the various ways of dividing s into several cases and analyzing them individually.
- The existence of s contradicts the pumping lemma if B were regular.
- Hence B cannot be regular.

The Pumping Lemma. . . — *continued*

Sipser, 1.4, p-80

- Finding s sometimes takes a bit of creative thinking.
- You may need to hunt through several candidates for s before you discover one that works.
- Try members of B that seem to exhibit the “essence” of B’s nonregularity.

Example

Sipser, Example 1.73, p-80

- Let B be the language $\{0^n 1^n \mid n \geq 0\}$.
- We use the pumping lemma to prove that B is not regular.
- The proof is by contradiction.

Example — *continued*

Sipser, Example 1.73, p-80

- Assume to the contrary that B is regular.
- Let p be the pumping length given by the pumping lemma.

Example — *continued*

Sipser, Example 1.73, p-80

- Choose s to be the string 0^p1^p.
- Because s is a member of B and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$.
- Where for any $i \geq 0$ the string xy^iz is in B.

Example — *continued*

Sipser, Example 1.73, p-80

- We consider three cases to show that this result is impossible.

---

1. The string y consists only of 0s.

- In this case, the string $xyyz$ has more 0s than 1s and so is not a member of B, violating condition 1 of the pumping lemma.
- This case is a contradiction.

Example — *continued*

Sipser, Example 1.73, p-80

- We consider three cases to show that this result is impossible.

---

2. The string y consists only of 1s.

- This case also gives a contradiction.

Example — *continued*

Sipser, Example 1.73, p-80

- We consider three cases to show that this result is impossible.

---

3. The string y consists of both 0s and 1s.

- In this case, the string $xyyz$ may have the same number of 0s and 1s, but they will be out of order with some 1s before 0s.
- Hence it is not a member of B, which is a contradiction.

Example — *continued*

Sipser, Example 1.73, p-80

- Thus a contradiction is unavoidable if we make the assumption that B is regular.
- So B is not regular.

Example

Sipser, Example 1.74, p-80

- $C = \{w \mid w \text{ has an equal number of 0s and 1s}\}$.
- We use the pumping lemma to prove that C is not regular.
- The proof is by contradiction.

Example — *continued*

Sipser, Example 1.74, p-80

- Assume to the contrary that C is regular.
- Let p be the pumping length given by the pumping lemma.
- Let s be the string 0^p1^p.
- With s being a member of C and having length more than p, the pumping lemma guarantees that s can be split into three pieces.
- $s = xyz$, where for any $i \geq 0$ the string xy^iz is in C.

Example — *continued*

Sipser, Example 1.74, p-80

- We would like to show that this outcome is impossible.

Example — *continued*

Sipser, Example 1.74, p-80

- But wait, it is possible!
- If we let x and z be the empty string and y be the string 0^p1^p, then xy^iz always has an equal number of 0s and 1s and hence is in C.
- So it seems that s can be pumped.

Example — *continued*

Sipser, Example 1.74, p-80

- Here condition 3 in the pumping lemma is useful.
- It stipulates that when pumping s, it must be divided so that $|xy| \leq p$.
- That restriction on the way that s may be divided makes it easier to show that the string $s = 0^p 1^p$ we selected cannot be pumped.
- If $|xy| \leq p$, then y must consist only of 0s, so $xyyz \notin C$.
- Therefore, s cannot be pumped.
- That gives us the desired contradiction.

Example

Sipser, Example 1.75, p-81

- $F = \{ww \mid w \in \{0,1\}^*\}$.
- We use the pumping lemma to prove that F is not regular.

Example — *continued*

Sipser, Example 1.75, p-81

- Assume to the contrary that F is regular.
- Let p be the pumping length given by the pumping lemma.
- Let s be the string $0^p 10p^1$.
- Because s is a member of F and s has length more than p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$, satisfying the three conditions of the lemma.

Example — *continued*

Sipser, Example 1.75, p-81

- We show that this outcome is impossible.
- Condition 3 is once again crucial because without it we could pump s if we let x and z be the empty string.
- With condition 3 the proof follows because y must consist only of 0s, so $xyyz \notin F$.

Example — *continued*

Sipser, Example 1.75, p-81

- Observe that we chose $s = 0^p10^p1$ to be a string that exhibits the “essence” of the nonregularity of F, as opposed to, say, the string 0^p0^p.
- Even though 0^p0^p is a member of F, it fails to demonstrate a contradiction because it can be pumped.

Example

Sipser, Example 1.76, p-82

- We demonstrate a nonregular unary language.
- $D = \left\{ 1^{n^2} \mid n \geq 0 \right\}$.
- We use the pumping lemma to prove that D is not regular.
- The proof is by contradiction.

Example — *continued*

Sipser, Example 1.76, p-82

- Assume to the contrary that D is regular.
- Let p be the pumping length given by the pumping lemma.

Example — *continued*

Sipser, Example 1.76, p-82

- Let s be the string 1^{p^2}.
- Because s is a member of D and s has length at least p, the pumping lemma guarantees that s can be split into three pieces, $s = xyz$.
- where for any $i \geq 0$ the string xy^iz is in D.

Example — *continued*

Sipser, Example 1.76, p-82

- We show that this outcome is impossible.
- The sequence of perfect squares:

$$0, 1, 4, 9, 16, 25, 36, 49, \ldots$$

- Note the growing gap between successive members of this sequence.
- Large members of this sequence cannot be near each other.

Example — *continued*

Sipser, Example 1.76, p-82

- Now consider the two strings xyz and xy^2z.
- These strings differ from each other by a single repetition of y.
- Consequently their lengths differ by the length of y.
- By condition 3 of the pumping lemma, $|xy| \leq p$ and thus $|y| \leq p$.

Example — *continued*

Sipser, Example 1.76, p-82

- We have $|xyz| = p^2$ and so $|xy^2z| \leq p^2 + p$.
- But $p^2 + p < p^2 + 2p + 1 = (p + 1)2$.
- Moreover, condition 2 implies that y is not the empty string and so $|xy^2z| > p^2$.
- Therefore, the length of xy^2z lies strictly between the consecutive perfect squares p^2 and $(p + 1)2$.
- Hence this length cannot be a perfect square itself.
- So we arrive at the contradiction $xy^2z \notin D$ and conclude that D is not regular.

Example

Sipser, Example 1.77, p-82

- Let E be the language $\{0^i1^j \mid i > j\}$.
- We use the pumping lemma to prove that E is not regular.
- The proof is by contradiction.

Example — *continued*

Sipser, Example 1.77, p-82

- Assume that E is regular.
- Let p be the pumping length for E given by the pumping lemma.

Example — *continued*

Sipser, Example 1.77, p-82

- Let $s = 0^{p+1}1^p$.
- Then s can be split into xyz, satisfying the conditions of the pumping lemma.
- By condition 3, y consists only of 0s.

Example — *continued*

Sipser, Example 1.77, p-82

- Let's examine the string $xyyz$ to see whether it can be in E.
- Adding an extra copy of y increases the number of 0s.
- But, E contains all strings in 01 that have more 0s than 1s.
- So increasing the number of 0s will still give a string in E.
- No contradiction occurs.

Example — *continued*

Sipser, Example 1.77, p-82

- We need to try something else.
- The pumping lemma states that $xy^iz \in E$ even when $i = 0$.

Example — *continued*

Sipser, Example 1.77, p-82

- So let's consider the string $xy^0z = xz$.
- Removing string y decreases the number of 0s in s.
- Recall that s has just one more 0 than 1.
- Therefore, xz cannot have more 0s than 1s, so it cannot be a member of E.
- Thus we obtain a contradiction.

Example

Hopcroft, Motwani, and Ullman, Example 4.3, p-129

- Let us show that the language L_{pr} consisting of all strings of 1's whose length is a prime is not a regular language.
- Suppose it were.
- Then there would be a constant n satisfying the conditions of the pumping Lemma.

Example — *continued*

Hopcroft, Motwani, and Ullman, Example 4.3, p-129

- Consider some prime $p \geq n + 2$.
- There must be such a p, since there are an infinity of primes.

Example — *continued*

Hopcroft, Motwani, and Ullman, Example 4.3, p-129

- Let $w = 1^r$.
- By the pumping lemma, we can break $w = xyz$ such that $y \neq \epsilon$ and $|xy| \leq n$.
- Let $|y| = m$.
- Then $|xz| = p - m$.

Example — *continued*

Hopcroft, Motwani, and Ullman, Example 4.3, p-129

- Now consider the string $xy^{p-m}z$.
- This must be in L_{pr} by the pumping lemma, if L_{pr} really is regular.
- However,

$$\begin{aligned} |xy^{p-m}z| &= |xz| + (p-m)|y| \\ &= p - m + (p-m)m \\ &= (m+1)(p-m) \end{aligned}$$

- It looks like $|xy^{p-m}z|$ is not a prime, since it has two factors $(m+1)$ and $(p-m)$.

Example — *continued*

Hopcroft, Motwani, and Ullman, Example 4.3, p-129

- However, we must check that neither of these factors are 1.
- Since then $(m+1)(p-m)$ might be a prime after all.
- But $m+1 > 1$, since $y \neq \epsilon$ tells us $m \geq 1$.
- Also, $p-m \geq 1$, since $p \geq n+2$ was chosen, and $m \leq n$ since

$$m = |y| \leq |xy| \leq n$$

- Thus, $p-m \geq 2$.

Example — *continued*

Hopcroft, Motwani, and Ullman, Example 4.3, p-129

- Again we have started by assuming the language in question was regular.
- We derived a contradiction by showing that some string not in the language was required by the pumping lemma to be in the language.
- Thus, we conclude that L_{pr} is not a regular language.


It's over now!