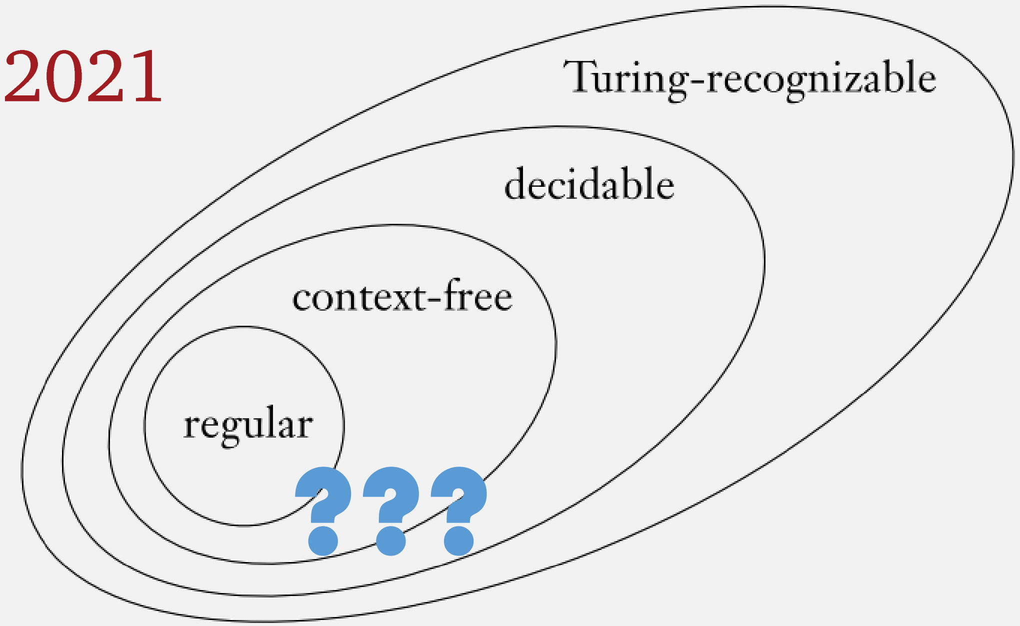
More Induction & Non-Regular Languages

Monday, February 22, 2021



Logistics

- New TA: Welcome Nick!
  - See course site for additional office hours
- HW3 in
- HW 4 out
  - Due Sunday 2/28 11:59pm
  - Create a regexp matcher! Practically interesting!
- HW4 is the last one with coding (based on your feedback)
  - And HW4 coding part is only a fraction of the points
  - Early assignments weighted less

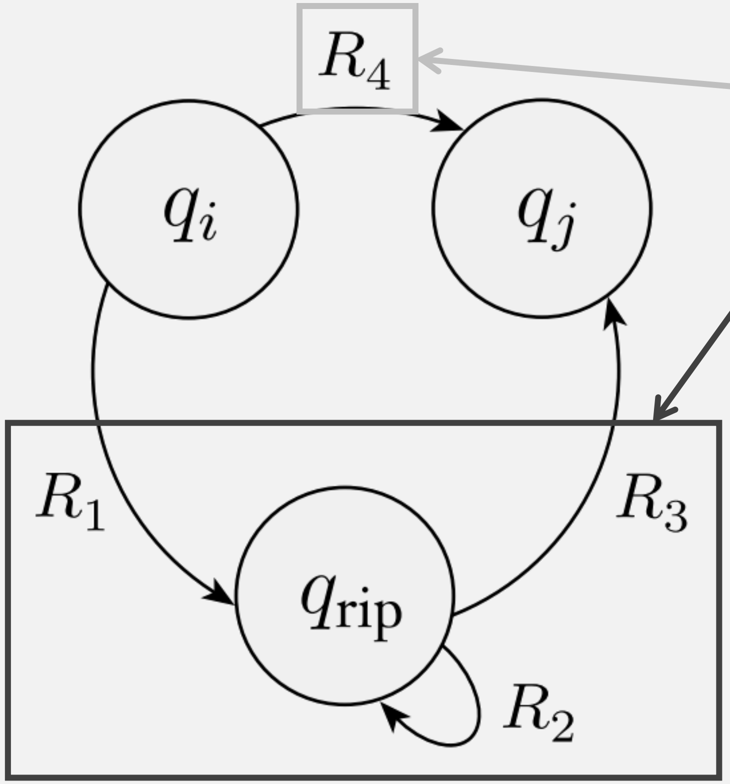
Last Time: Regular Language ⇔ Regular Expression

- => If a language is regular, it is described by a regular expression
  - We know a regular lang has an NFA recognizing it (Thm 1.40)
  - Use GNFA->Regexp function to convert NFA to equiv regular expression
- <= If a language is described by a regular expression, it is regular
  - Convert the regular expression to an NFA (Thm 1.55)

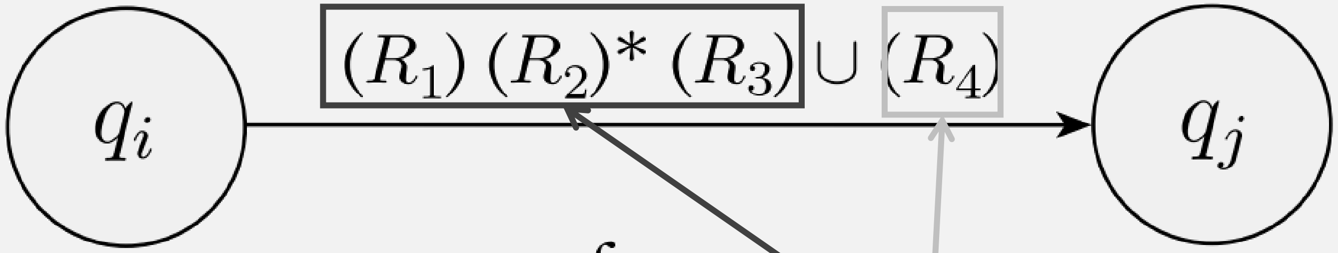
So a regular language has these equivalent representations:
- **DFA**
- **NFA**
- **Regular Expression**

Last time: **GNFA->Regexp** "Rip/Repair" Step

Before: two paths from q_i to q_j:
1. Not through q_{rip}
2. Through q_{rip}



before



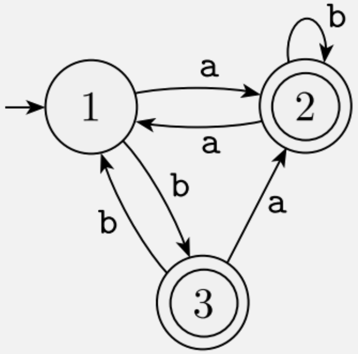
after

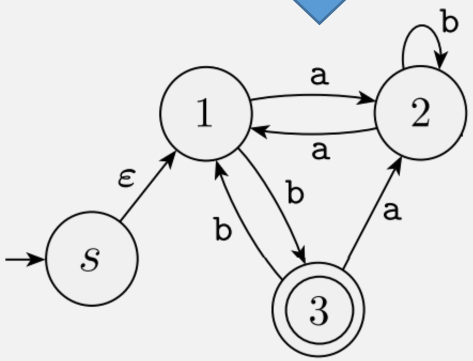
After: two regexp "paths" from q_i to q_j
1. Not through q_{rip}
2. Through q_{rip}

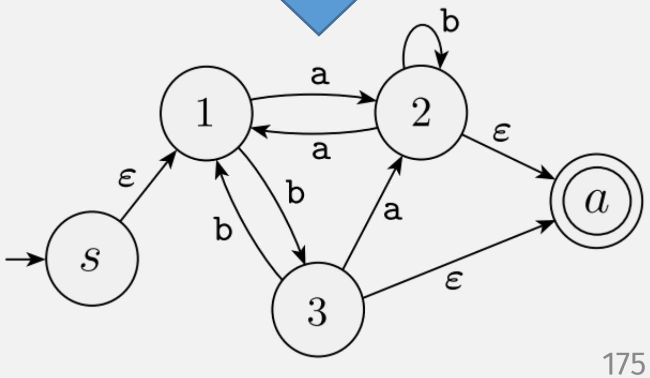
Question: What if q_{rip} is an accept state?
Answer: q_{rip} cannot be a start or accept state

Update: GNFA->Regexpr

- First modifies input machine to have:
  - New start state
    - With no incoming transitions
    - And epsilon transition to old start state
  - New, single accept state
    - With epsilon transitions from old accept states





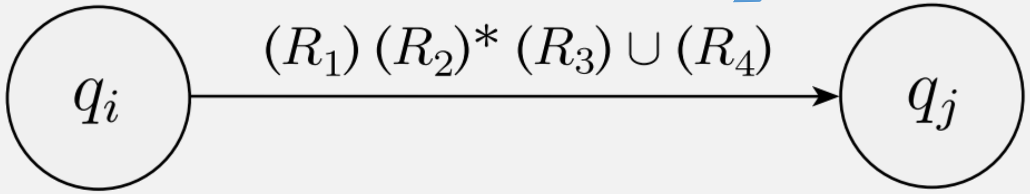


Last time: GNFA->Regexp function

- On GNFA input G:
  - If G has 2 states, return the regular expression transition, e.g.:

Base case

Equivalent Regular expression



GNFA

Inductive case

  - Else:
    - "Rip out" one state and "repair" to get G' (has one less state than G)
    - Recursively call GNFA->Regexp(G')

Recursive call is "smaller"

This is a recursive (inductive) definition!

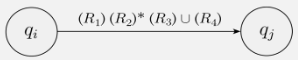
Last time: Kinds of Mathematical Proof

- Proof by construction
- Proof by contradiction
- Proof by induction
  - Use to prove properties of recursive (inductive) defs or functions
  - Proof steps follow the inductive definition

<u>Last time</u>: Proof by Induction

EXAMPLE OF A "P":
LANGOF (G)
=
LANGOF (GNFA->Regexp (G))

To prove that a **property** P is true for a **thing** x:



1. Prove that P is true for the <u>base case</u> of x (usually easy)

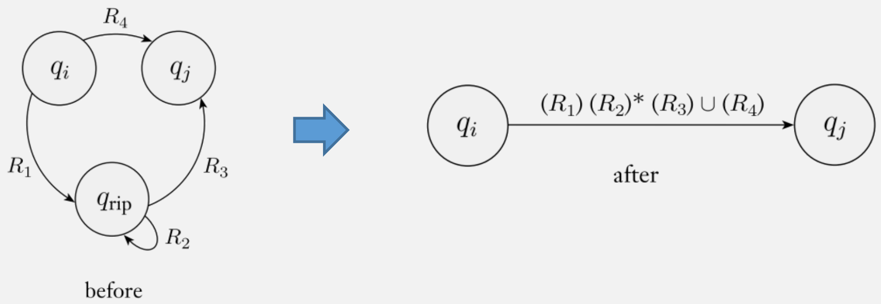
   G has two states

2. Prove the <u>induction step</u>:
   - Assume the <u>induction hypothesis</u> (IH):
     - P(x) is true, for some $x_{smaller}$ that is smaller than x

     LANGOF (G')
     =
     LANGOF (GNFA->Regexp (G'))
     (Where G' smaller than G)

   - and use it to prove P(x)

     Show that "rip/repair" step converts G to smaller, equiv G'



Regular Expressions, Formal Definition

DEFINITION 1.52

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
6. (R_1^*), where R_1 is a regular expression.

This is weird?
Regular expressions defined
using regular expressions?

It's a Recursive Definition!

DEFINITION 1.52

Say that R is a ***regular expression*** if R is

1. a for some a in the alphabet Σ,

2. ε, 3 base cases

3. $\emptyset$,

4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,

5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or

6. (R_1^*), where R_1 is a regular expression.

3 inductive cases

"smaller" self-references

How to prove a theorem about Reg ~~Exprs~~? **Languages!**

- Proof by construction

- Proof by contradiction

- Proof by induction
  - On Regular Expressions!

How to prove a theorem about Reg ~~Exprs~~ **Languages!**?

We now have 2 proof techniques! You choose

- Proof by construction (can still prove things this way)
  - Construct DFA or NFA ←
- Proof by contradiction
- Proof by induction
  - On Regular Expressions!

Homomorphism: Closed under Reg Langs

A ***homomorphism*** is a function $f\colon \Sigma \longrightarrow \Gamma$ from one alphabet to another.

- Assume f can be used on both strings and characters
- E.g., like a secret decoder!
  - f (“x”) -> “c”
  - f (“y”) -> “a”
  - f (“z”) -> “t”
  - f (“xyz”) -> “cat”
- <u>Prove</u>: homomorphisms are <u>closed</u> under regular languages
  - E.g., if lang A is regular, then $f(A)$ is regular

How to prove a theorem about Reg ~~Exprs~~ **Languages!**?

We now have 2 proof techniques! You choose

- Proof by construction ←
  - Construct DFA or NFA
- Proof by contradiction
- Proof by induction ←
  - On Regular Expressions!

Thm: Homomorphism Closed for Reg Langs

- Proof by construction
  - If a lang A is regular, then we know DFA M recognizes it.
  - So modify M such that transitions use the new alphabet
  - (Details left to you to work out)

- Proof by induction:
  - If a lang A is regular, then some reg expression R describes it.

A ***homomorphism*** is a function $f\colon \Sigma \longrightarrow \Gamma$ from one alphabet to another.

Homomorphism Closure: Inductive proof

DEFINITION 1.52

Say that R is a ***regular expression*** if R

Inductive proof must handle all cases, e.g.,
- If: regexpr “a” describes a reg lang,
- then: f(“a”) is describes a reg lang
- because: it’s still a single-char regexpr,
- so: homomorphism closed under reg langs (for this case)

3 base cases

1. a for some a in the alphabet Σ,
2. ε,
3. $\emptyset$,
4. $(R_1 \cup R_2)$
5. $(R_1 \circ R_2)$, wh… , or
6. (R_1^*), where R_1 is a regular expression.

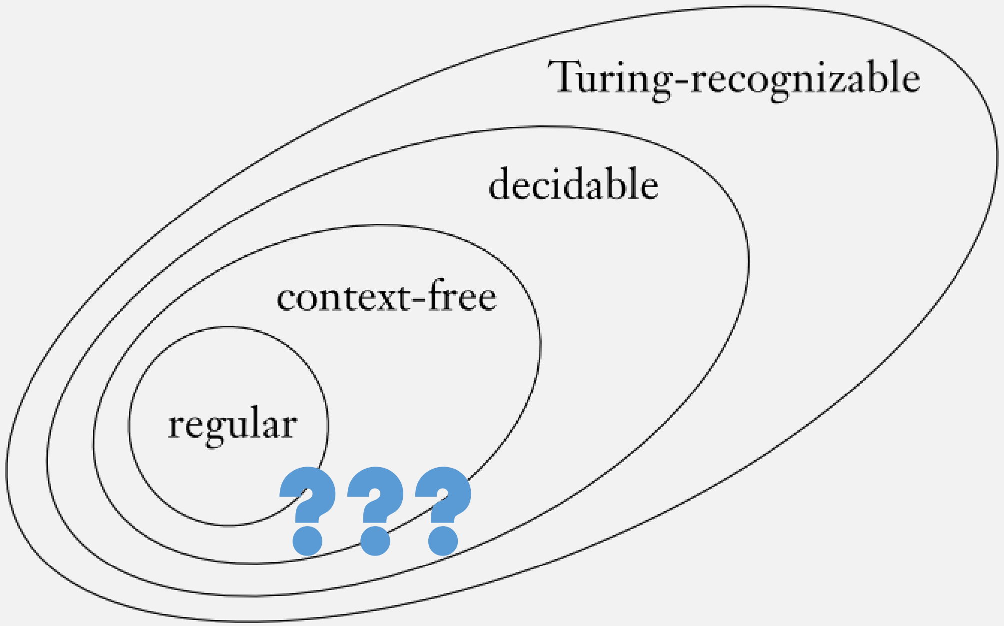
IH: assume applying homomorphism f to **smaller** R_1 (and R_2) produces a regular lang, i.e., $f(R_1)$ and $f(R_2)$ are regular langs

To finish proof: need to show $f(R_1) \cup f(R_2)$ is a reg lang

(If only union operation were closed for reg langs ☺)

A ***homomorphism*** is a function $f\colon \Sigma \longrightarrow \Gamma$ from one alphabet to another.

Non-Regular Languages



Non-Regular Languages

- We now have many ways to prove that a language is regular:
  - Construct a <u>DFA</u> or <u>NFA</u> (or GNFA)
  - Come up with a <u>regular expression</u> describing the language

- But how to show that a language is **not regular**?

- E.g., HTML / XML is not a regular language
  - But how can we prove it

- <u>Preview</u>: The Pumping Lemma!

Flashback: Designing DFAs or NFAs

- States = the machine's **memory**!
  - Each state "stores" some information
  - Finite states = finite amount of memory
  - And must be allocated in advance

- This means DFAs can't keep track of an arbitrary count!
  - would require infinite states

A Non-Regular Language

- $L = \{ 0^n 1^n \mid n >= 0 \}$

- A DFA recognizing L would require infinite states! (impossible)

- This language is the essence of XML!
  - To better see this replace:
    - “0” -> “<tag>“
    - “1” -> “</tag>”

Still, how do we prove non-regularness?

- The problem is tracking the **nestedness**
  - Regular languages cannot count arbitrary nesting depths
  - So most programming languages are also not regular!

The Pumping Lemma

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. $|xy| \leq p$.

Pumping lemma specifies three conditions that a regular language must satisfy

Specifically, strings in the language longer than some length p must satisfy the conditions

But it doesn't tell you an exact p! You have to find it.

The Pumping Lemma

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. $|xy| \leq p$.

Because a finite lang is regular, then these conditions must be true for all strings in the lang "of length at least p"

- Example: a finite-sized language, e.g., {"ab", "cd"}
  - All finite langs are regular bc we can easily construct DFA/NFA recognizing them
  - One possible p = length of longest string in the language, plus 1
  - In a finite lang, # strings "of length at least p" = 0
    - Therefore "all" strings "of length at least p" satisfy the pumping lemma criteria!

The Pumping Lemma

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. $|xy| \leq p$.

In an *infinite* regular lang, these conditions must be true for all strings in the lang "of length at least *p*"

- <u>Example</u>: a *infinite* language, e.g., {"00", "010" , "0110" , "01110", ...}
  - This language is regular bc it's described by regular expression 01*0
  - E.g., "010" is in the lang, and we can split into three parts: x = 0, y = 1, z = 0
    - And any pumping (ie, repeating) of y creates a string that is still in the language
      - E.g., i = 1 -> "010", I = 2 -> "0110", i = 3 -> "01110"
    - This is what the pumping lemma requires

The Pumping Lemma

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:

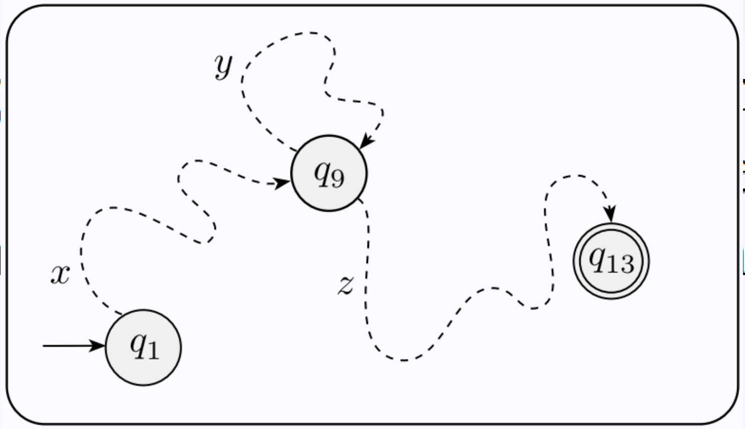
1. for each $i \geq 0$, $xy^iz \in A$,
2. $|y| > 0$, and
3. $|xy| \leq p$.

In an *infinite* regular lang, these conditions must be true for all strings in the lang "of length at least *p*"

- <u>Example</u>: a *infinite* language, e.g., {"00", "010" , "0110" , "01110", ...}
  - This language is regular bc it's described by regular expression 01*0
  - p = ????

The Pumping Lemma, a Closer Look

Pumping lemma If A is a regular lang … nber p (the pumping length) where if s is any string in … en s may be divided into three pieces, $s = xyz$, satisfyin … s:



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

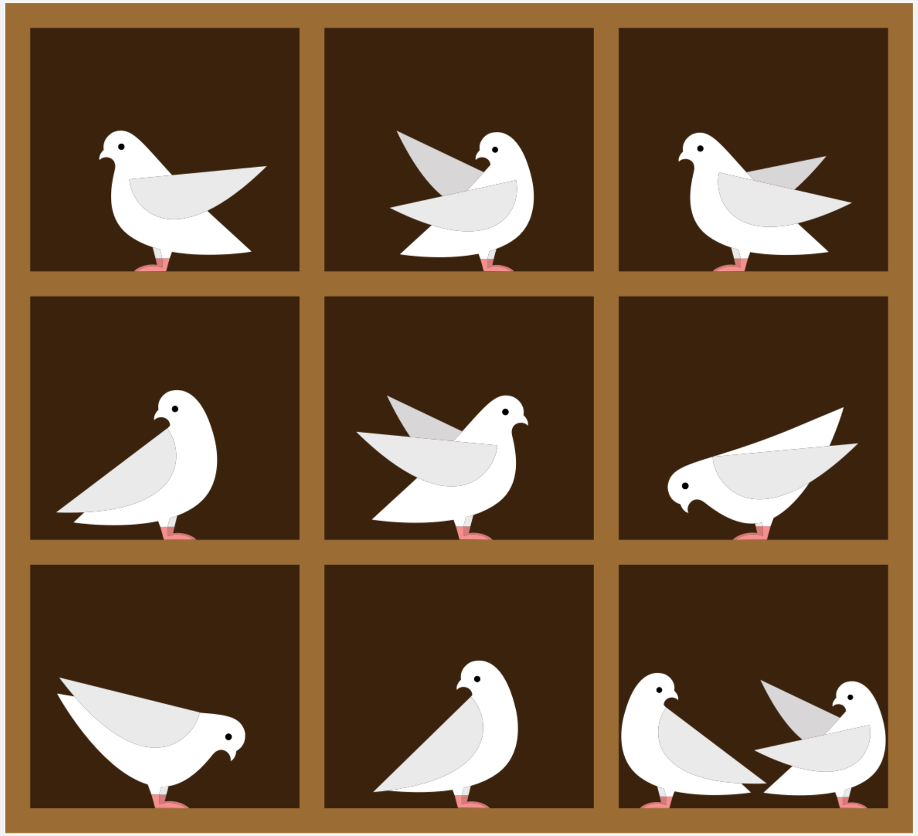
Pumping lemma says that for "long enough" strings, you should be able to <u>repeat</u> a part of it, and that "pumped" string will still be in the language

- Strings that have a <u>repeatable</u> part can be split into:
  - x = the part <u>before</u> any repeating
  - y = the repeated part
  - z = the part <u>after</u> any repeating

This makes sense because DFAs have a finite number of states, so for "long enough" inputs, some state must repeat

The Pigeonhole Principle!

The Pigeonhole Principle



The Pumping Lemma

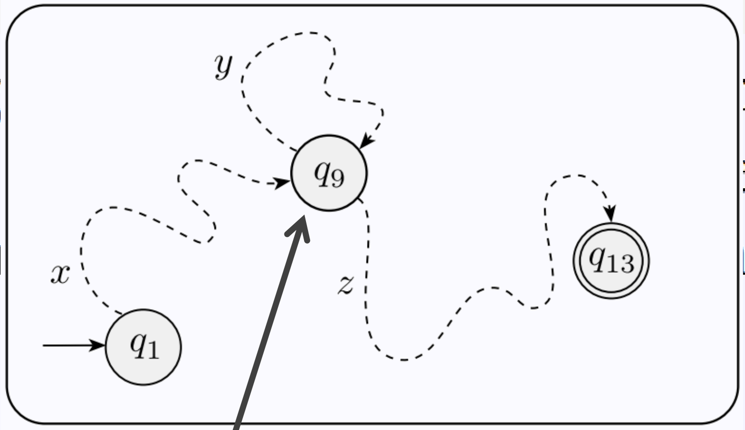
Pumping lemma If A is a regular language, then there is a number p (the pumping length) where if s is any string in A of length at least p, then s may be divided into three pieces, $s = xyz$, satisfying the following conditions:

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

- Example: a *infinite* language, e.g., {"00", "010" , "0110" , "01110", ...}
  - This language is regular bc it's described by regular expression 0*
  - p = ????

The Pumping Lemma

Pumping lemma If A is a regular lang nber p (the pumping length) where if s is any string in en s may be divided into three pieces, $s = xyz$, satisfyin s:



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

"pumpable" part of string

But how does this prove that a language is **NOT** regular??

- <u>Example</u>: a *infinite* language, e.g., {"00", "010" , "01
  - This language is regular bc it's described by regular ex
  - p = number of states, plus 1
    - When running an input longer than p, one state is guaranteed to be visited twice
    - That state represents the "pumpable" part of the string

Poll: Conditional Statements

Equivalence of Conditional Statements

- Yes or No? “If X then Y” is equivalent to:
  - “If Y then X” (converse)
    - No!
  - “If not X then not Y” (inverse)
    - No!
  - “If not Y then not X” (contrapositive)
    - <u>Yes!</u>
    - Proof by contradiction relies on this equivalence

Kinds of Mathematical Proof

- Proof by construction
  - Construct the object in question
- Proof by contradiction
  - Proving the contrapositive
- Proof by induction
  - Use to prove properties of recursive definitions or functions

Pumping Lemma: Proving Non-Regularity

... then the language is **not** regular

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 followin

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

If any of these are **not** true ...

IMPORTANT NOTE: The pumping lemma **cannot** prove that a language is regular

Contrapositive:
“If X then Y” is equivalent to “If **not** Y then **not** X

Pumping Lemma: Non-Regularity Example

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. $|xy| \leq p$.

Let B be the language $\{0^n1^n | n \geq 0\}$. We use the pumping lemma to prove that B is not regular. The proof is by contradiction.

Check-in Quiz 2/22

On gradescope

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language.
   Then it must satisfy the pumping lemma where p is the pumping length.

Theorem

The language $B = \{0^n1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language.
   Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string 0^p1^p.

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language.
   Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^p 1^p$.
4. Show contradiction of assumption: Because $s \in B$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^i z \in B$ for $i \geq 0$. But we show this is impossible:

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^p 1^p$.
4. Show contradiction of assumption: Because $s \in B$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^i z \in B$ for $i \geq 0$. But we show this is impossible:
5. The contradiction step typically requires detailed case analysis of scenarios. There are three possible cases:

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ is not regular.

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language.
   Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^p 1^p$.
4. Show contradiction of assumption: Because $s \in B$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^i z \in B$ for $i \geq 0$. But we show this is impossible:
5. The contradiction step typically requires detailed case analysis of scenarios.
   There are three possible cases:
   5.1 y is all 0s: Pumped strings, e.g., $xyyz$, are not in B because they have more 0s than 1s, breaking condition 1 of the pumping lemma. So we have a contradiction.

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language.
   Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^p 1^p$.
4. Show contradiction of assumption: Because $s \in B$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^i z \in B$ for $i \geq 0$. But we show this is impossible:
5. The contradiction step typically requires detailed case analysis of scenarios.
   There are three possible cases:
   5.1 y is all 0s: Pumped strings, e.g., $xyyz$, are not in B because they have more 0s than 1s, breaking condition 1 of the pumping lemma. So we have a contradiction.
   5.2 y is all 1s: Same as above.

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language.
   Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^p 1^p$.
4. Show contradiction of assumption: Because $s \in B$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^iz \in B$ for $i \geq 0$. But we show this is impossible:
5. The contradiction step typically requires detailed case analysis of scenarios.
   There are three possible cases:
   - 5.1 y is all 0s: Pumped strings, e.g., $xyyz$, are not in B because they have more 0s than 1s, breaking condition 1 of the pumping lemma. So we have a contradiction.
   - 5.2 y is all 1s: Same as above.
   - 5.3 y has both 0s and 1s: Pumped strings preserve equal counts, but is out of order and therefore not in B, breaking condition 1.

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language.
   Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^p 1^p$.
4. Show contradiction of assumption: Because $s \in B$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^i z \in B$ for $i \geq 0$. But we show this is impossible:
5. The contradiction step typically requires detailed case analysis of scenarios.
   There are three possible cases:
   5.1 y is all 0s: Pumped strings, e.g., $xyyz$, are not in B because they have more 0s than 1s, breaking condition 1 of the pumping lemma. So we have a contradiction.
   5.2 y is all 1s: Same as above.
   5.3 y has both 0s and 1s: Pumped strings preserve equal counts, but is out of order and therefore not in B, breaking condition 1.
6. Conclusion: Since all cases result in contradiction, B must not be regular.

Theorem

The language $B = \{0^n 1^n \mid n \geq 0\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that B is a regular language.
   Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^p 1^p$.
4. Show contradiction of assumption: Because $s \in B$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^i z \in B$ for $i \geq 0$. But we show this is impossible:
5. The contradiction step typically requires detailed case analysis of scenarios.
   There are three possible cases:
   5.1 y is all 0s: Pumped strings, e.g., $xyyz$, are not in B because they have more 0s than 1s, breaking condition 1 of the pumping lemma. So we have a contradiction.
   5.2 y is all 1s: Same as above.
   5.3 y has both 0s and 1s: Pumped strings preserve equal counts, but is out of order and therefore not in B, breaking condition 1.
6. **Alternate Proof:** Last 2 cases not needed; see pumping lemma, condition 3.

Using Condition 3 of the Pumping Lemma

Theorem

The language $F = \{ww \mid w \in \{0,1\}^*\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.

Using Condition 3 of the Pumping Lemma

Theorem

The language $F = \{ww \mid w \in \{0,1\}^*\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that F is regular. Then it must satisfy the pumping lemma where p is the pumping length.

Using Condition 3 of the Pumping Lemma

Theorem

The language $F = \{ww \mid w \in \{0,1\}^*\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that F is regular. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string 0^p10^p1.

Using Condition 3 of the Pumping Lemma

Theorem

The language $F = \{ww \mid w \in \{0,1\}^*\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that F is regular. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string 0^p10^p1.
4. Show contradiction of assumption: Because $s \in F$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^iz \in F$ for $i \geq 0$. But this is impossible.

Using Condition 3 of the Pumping Lemma

Theorem

The language $F = \{ww \mid w \in \{0,1\}^*\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that F is regular. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string 0^p10^p1.
4. Show contradiction of assumption: Because $s \in F$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^iz \in F$ for $i \geq 0$. But this is impossible.
5. This time there is only one possible case, but we must explain why. According to condition 3 of the pumping lemma $|xy| \leq p$. So p is all 0s. But then $xyyz \notin F$, breaking condition 1 of the pumping lemma. So we have a contradiction.

Using Condition 3 of the Pumping Lemma

Theorem

The language $F = \{ww \mid w \in \{0,1\}^*\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that F is regular. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string 0^p10^p1.
4. Show contradiction of assumption: Because $s \in F$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^iz \in F$ for $i \geq 0$. But this is impossible.
5. This time there is only one possible case, but we must explain why. According to condition 3 of the pumping lemma $|xy| \leq p$. So p is all 0s. But then $xyyz \notin F$, breaking condition 1 of the pumping lemma. So we have a contradiction.
6. Conclusion: Since all cases result in contradiction, F must not be regular.

□

Pumping Down

Theorem

The language $E = \{0^i 1^j \mid i > j\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.

Pumping Down

Theorem

The language $E = \{0^i 1^j \mid i > j\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that E is regular. Then it must satisfy the pumping lemma where p is the pumping length.

Pumping Down

Theorem

The language $E = \{0^i 1^j \mid i > j\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that E is regular. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^{p+1}1^p$.

Pumping Down

Theorem

The language $E = \{0^i 1^j \mid i > j\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that E is regular. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^{p+1}1^p$.
4. Show contradiction of assumption: Because $s \in E$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^iz \in E$ for $i \geq 0$. But this is impossible.

Pumping Down

Theorem

The language $E = \{0^i 1^j \mid i > j\}$ is not regular.

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that E is regular. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^{p+1}1^p$.
4. Show contradiction of assumption: Because $s \in E$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^iz \in E$ for $i \geq 0$. But this is impossible.
5. Again, one possible case. According to condition 3 of the pumping lemma $|xy| \leq p$. So p is all 0s. But then $xz \notin E$ $(i = 0)$, breaking condition 1 of the pumping lemma. So we have a contradiction.

Pumping Down

Theorem

The language $E = \{0^i 1^j \mid i > j\}$ *is not regular.*

Proof.

This proof is annotated with commentary in blue. (Commentary not needed for hw proofs.)

1. State the kind of proof: The proof is by contradiction.
2. State assumptions: Assume that E is regular. Then it must satisfy the pumping lemma where p is the pumping length.
3. Present counterexample: Choose s to be the string $0^{p+1}1^p$.
4. Show contradiction of assumption: Because $s \in E$ and has length $> p$, the pumping lemma guarantees that s can be split into three pieces $s = xyz$ where $xy^iz \in E$ for $i \geq 0$. But this is impossible.
5. Again, one possible case. According to condition 3 of the pumping lemma $|xy| \leq p$. So p is all 0s. But then $xz \notin E$ $(i = 0)$, breaking condition 1 of the pumping lemma. So we have a contradiction.
6. Conclusion: Since all cases result in contradiction, E must not be regular.