San José State University
Department of Computer Science

Ahmad Yazdankhah

ahmad.yazdankhah@sjsu.edu
www.cs.sjsu.edu/~yazdankhah

Non-Regular Languages

(Part 1)

Lecture 24

Day 28/31

CS 154

Formal Languages and Computability

Spring 2018

Agenda of Day 28

- About Final Exam
- Solution and Feedback of Quiz 9
- Summary of Lecture 23
- Lecture 24: Teaching ...
  - Non-Regular Languages (Part 1)

About Final Exam

Reminder 1

- **Value**: 20%
- **Topics**: Everything covered from the beginning of the semester
- **Type**: Closed all materials

Section	Date	Time	Venue
01 (TR 4:30)	Thursday, May 17	2:45 – 5:00 pm	MH 233
02 (TR 6:00)	Thursday, May 17	5:15 – 7:30 pm	MH 233
03 (TR 3:00)	Tuesday, May 22	2:45 – 5:00 pm	SCI 311

- We won't need whole 2:15 hours.
- As usual, I'll announce officially the type and number of questions via Canvas. (study guide)

Solution and Feedback of Quiz 9 (Out of 30)

Metrics	Section 1	Section 2	Section 3
Average	25	23	24
High Score	29	27	30
Low Score	20	18	11

Summary of Lecture 23: We learned ...

Grammars: Parser Algorithms

- There are two types of algorithms for parsers:
  - Top-down and bottom-up
- Exhaustive parsing algorithm is ...
  - ... a top-down algorithm that check all possible derivations to find a derivation sequence for a given string.
- This algorithm has two serious problems:
  - It is extremely inefficient: $O(|P|^{2|w|+1})$
  - It is possible that it never terminates.

- Two good news:
  1. Theorem: there exists an efficient algorithm for every CFG with complexity $|w|^3$.
  2. If we use s-grammar, the efficiency would be $O(|w|)$.

Ambiguity in Grammars

- Ambiguity of grammars ...

  ... happens when for some strings in the language, we can construct two or more parse-tree.

Any Question

Objective of This Lecture

- We defined "regular languages" as ...
- A language is called regular iff there exists a ...
  - DFA/NFA to accept it.
  - REGEX to generate it.
  - regular grammar to generate it.

- The main question of this lecture is:

  How to PROVE a language is NON-REGULAR?

- Obviously, we cannot say:

  L is non-regular because I cannot construct a DFA/NFA/REGEX/regular grammar for it!

Objective of This Lecture

- Also, we used a heuristic technique to figure out a language was non-regular.
  - We looked at the language's pattern and if it needed some kind of memory, it could not be regular.

- But this is NOT a mathematical proof!
- Also, in some cases, we might make mistakes.
  - e.g.: $L = \{w : w \text{ has an equal number of ab and ba}\}$

- So, in this lecture we are looking for a ...
  ... solid technique to prove a language is non-regular.

Background

Required & Recommended Background

Required

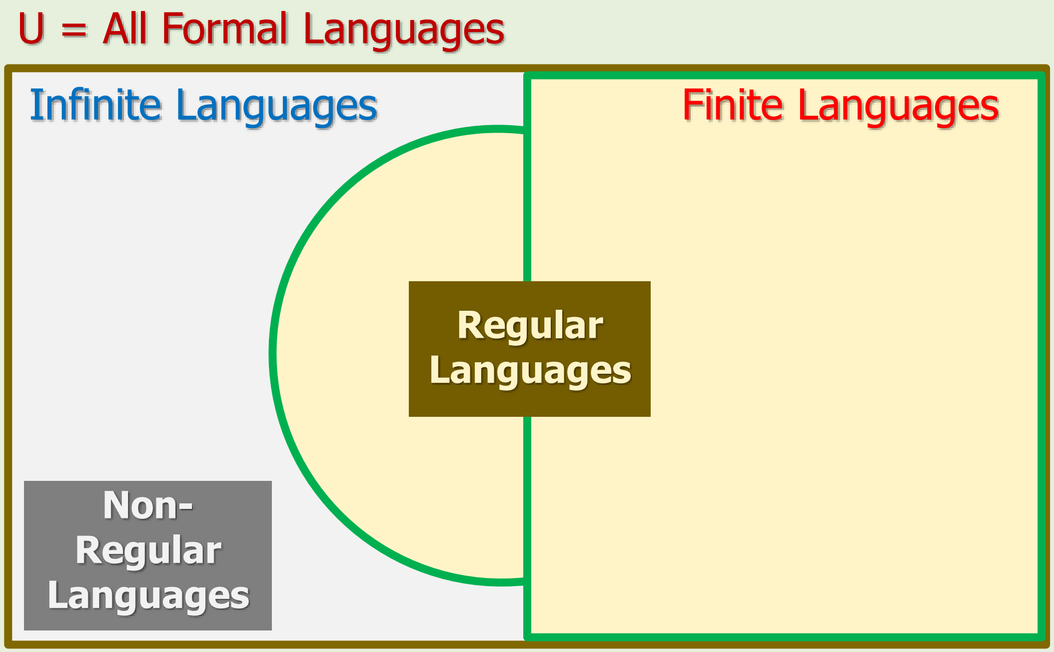
1. The concept of regular and non-regular languages
2. Proof by contradiction
3. Cycle and simple cycle definitions in graphs
4. One-dimensional projection of a walk
5. Pigeonhole principle (will be covered)

Recommended

1. Predicate calculus

Regular and Non-Regular Languages

Recap



Proof by Contradiction

Recap

- Logically, proving a theorem means to assume the truth of some statements (e.g.: p) and entailing the truth of another statement (e.g.: q)
- Sometimes, it is hard to follow this procedure.
- In these cases, we might use the following logical equivalency:

Contradiositive

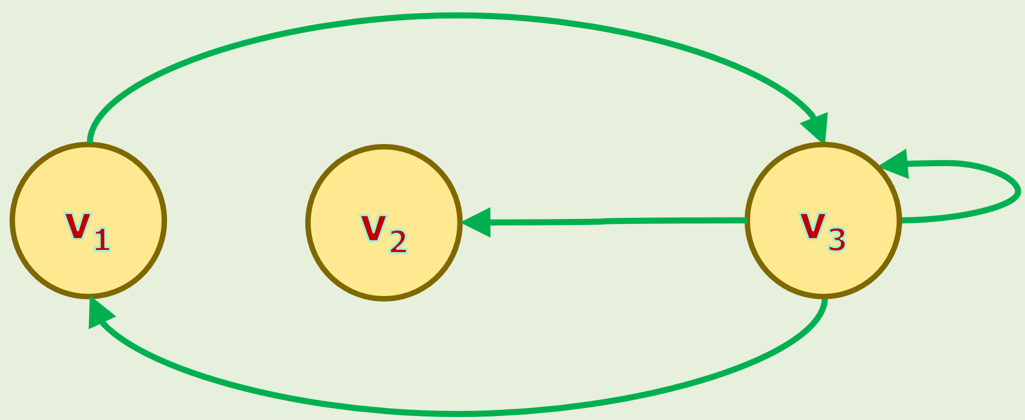
$$p \rightarrow q \equiv \sim q \rightarrow \sim p$$

- In fact, we prove that if the negation of the desired result (e.g. ~q) is true, then it leads to a contradiction.
- And to resolve the contradiction, we have no choice except blaming our assumption (~q is true) and this means $q \equiv T$.
- This technique is called "proof by contradiction".

Cycle

Recap

- A walk from a vertex (called base) to itself with no repeated edges.
- But: Walk + No repeated edges = path
- Rewording: A cycle is a path from a vertex (called base) to itself.



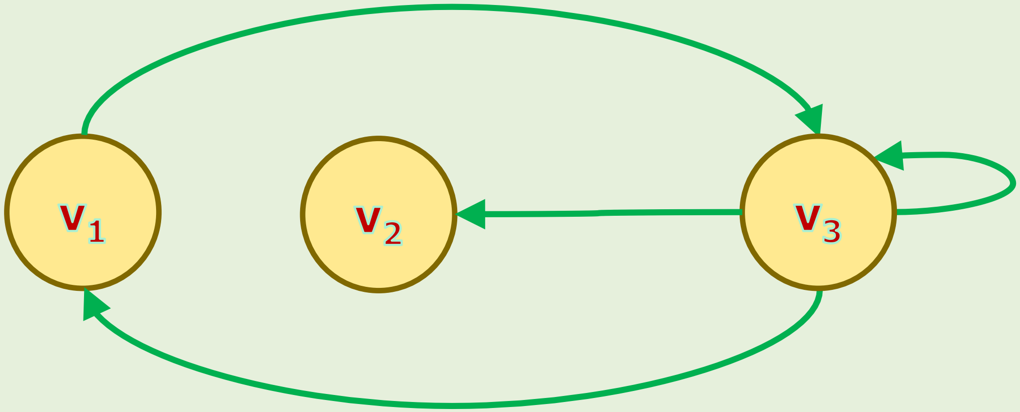
Examples 1

- Walk 1: $(v_1, v_3), (v_3, v_1)$
- Walk 2: $(v_1, v_3), (v_3, v_3), (v_3, v_1)$
- Walk 3: (v_3, v_3)

Simple Cycle

Recap

- A cycle that no vertex other than the base is repeated.



Examples 2

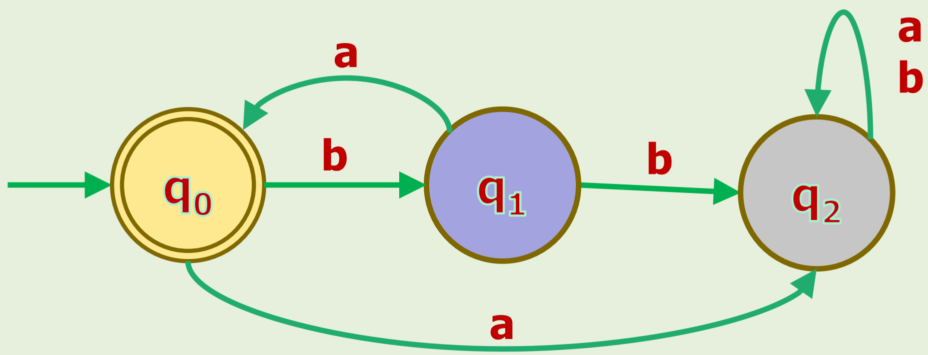
- Walk 1: $(v_1 , v_3) , (v_3 , v_1)$
- Walk 2: $(v_3 , v_1) , (v_1 , v_3)$

One-Dimensional Projection of a String

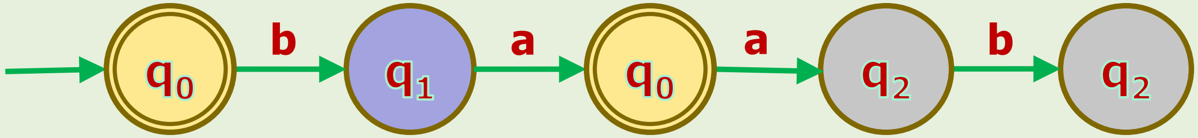
Recap

Example 3

- Given following DFA with 3 states over $\Sigma = \{a, b\}$:



- Show one-dimensional projection of w = baab.

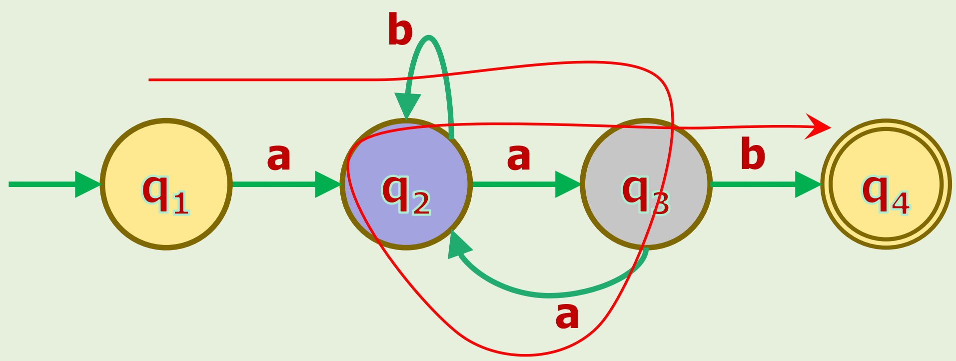


One-Dimensional Projection of a Walk

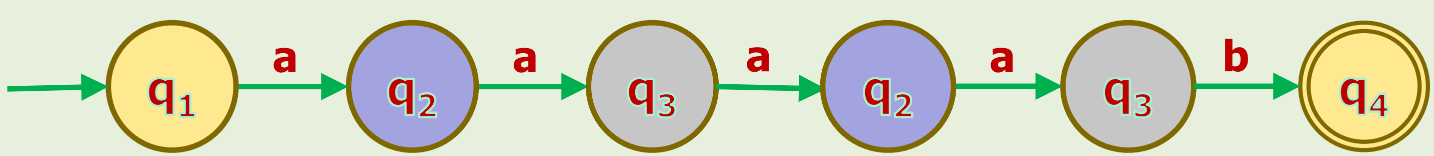
Recap

Example 4

- Given following NFA with 4 states over $\Sigma = \{a, b\}$:



- Show one-dimensional projection of w = aaaab.



Pumping Lemma

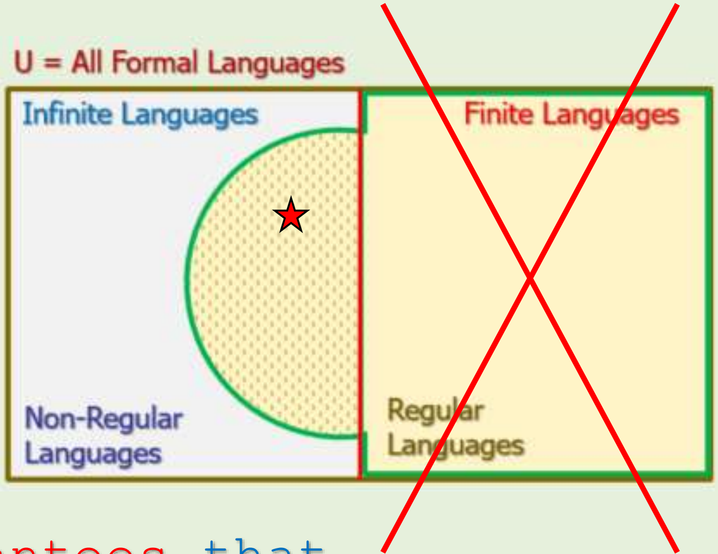
What is a Lemma?

Etymology

- "Lemma" is a smaller theorem to help proving a bigger one.

- Very occasionally lemmas can take on a life of their own.
- In computer science, "pumping lemma" is one of them.

Pumping Lemma



If L is an INFINITE regular language,

Then there exists an $m \geq 1$ such that

If $w \in L$ and $|w| \geq m$

Then `//pumping lemma guarantees that ...`

We must be able to divide w into three parts xyz in such a way that all of the following conditions are satisfied:

$|xy| \leq m$, and

$|y| \geq 1$, and

$w_i = x\,y^i\,z \in L$ for $i = 0, 1, 2, 3, \ldots$.

Formal Statement of Pumping Lemma

If L is an infinite regular language,

Then

there exists an $m \geq 1$ such that

If $w \in L$ and $|w| \geq m$

Then //P. L. guarantees that ...

We must be able to divide w into xyz in such a way that all of the following conditions are satisfied:

$|xy| \leq m$, and

$|y| \geq 1$, and

$w_i = x\, y^i\, z \in L$

for $i = 0, 1, 2, \ldots$.

If L is an infinite regular language,

Then

$(\exists\, m \geq 1)$

$[(w \in L \text{ and } |w| \geq m) \rightarrow$

$(\exists\, x,y,z)\, ($

$w = xyz \land$

$|xy| \leq m \land$

$|y| \geq 1 \land$

$(\forall\, i \in \mathbb{N})\, (w_i = x\, y^i\, z \in L)$

$)]$

Pumping Lemma

Example 5

- Verify the pumping lemma property on the following infinite regular language.

  $L = \{a^n b: n \geq 0\}$

- Let's take the m = 2. Why not 3?
- OK, let's take it as m.
- If we need, we'd make some boundary on it later.
- Let's take $w = a^m b$
  //note that m is constant and finite.
- Check its size:
  $|w| = |a^m b| = m+1 \geq m$ ✓

- Pumping lemma guarantees that:
- There exists x, y, z such that:
- $w = a^m b = xyz = \lambda \quad a \quad a^{m-1}b$
- $|xy| = |a| = 1 \leq m$ ✓
- $|y| = 1 \geq 1$ ✓
- $xz = a^{m-1}b \in L$ $i=0$
- $xy^1z = a^m b \in L$ $i=1$
- $xy^2z = a^{m+1}b \in L$ $i=2$
- $xy^3z = a^{m+2}b \in L$ $i=3$
- ...
- $xy^iz \in L$ ✓

Pumping Lemma: Notes

1. In the previous example, we took
   $w = a^m b = xyz = \lambda \quad a \quad a^{m-1}b$
   - Note that $a^m b$ is a string of the language, and not a pattern because m is a constant.
   - We should make sure that no string gets negative power.
   - For example, if we something like $a^{m-3}b$, then we should mention "we pick $m \geq 3$".
   - So, in the previous example, we should mention $m \geq 1$ somewhere, but in this particular case we don't need because by default $m \geq 1$.
   - Recall that the pumping lemma has the power to make a boundary for 'm'.

Pumping Lemma: Notes

2. One might take w something else such as:
   - $a^{2m}b$ or $a^{m+100}b$
   - But, take it as simple as possible.
3. One might take x, y, and z something else such as:
   - $w = xyz = a^{m-5}\, a^2\, a^3b$ //in this case, you need to mention $m \geq 5$.
   - Again, take it as simple as possible.

Pumping Lemma

Example 6

- Verify the pumping lemma property on the following infinite regular language.

$$L = \{bba^n : n \geq 0\}$$

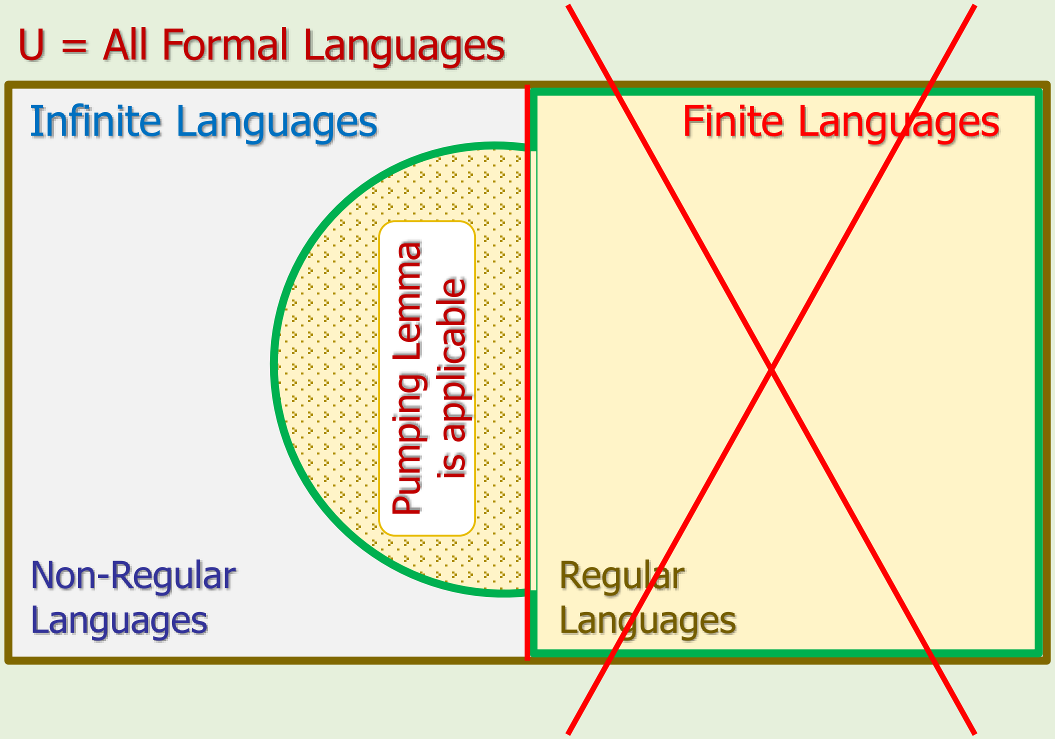
Homework

- Verify the pumping lemma property on the following infinite regular languages.

1. $L = \{a^n b^k : n \geq 0, k \geq 0\}$
2. $L = \{aaab^n (ab)^k : n \geq 0, k \geq 0\}$
3. $L = \{(ab)^n : n \geq 0\}$

Conclusion

- This is an important property of "INFINITE regular languages".



- If an "infinite language" does not have this property, it is "non-regular".

References

1. Linz, Peter, "An Introduction to Formal Languages and Automata, 5th ed.," Jones & Bartlett Learning, LLC, Canada, 2012
2. Kenneth H. Rosen, "Discrete Mathematics and Its Applications, 7th ed.," McGraw Hill, New York, United States, 2012
3. Costas Busch's website: http://csc.lsu.edu/~busch/
4. Michael Sipser, "Introduction to the Theory of Computation, 3rd ed.," CENGAGE Learning, United States, 2013
ISBN-13: 978-1133187790