San José State University
Department of Computer Science

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

Regular Languages

Lecture 12

Day 13/31

CS 154

Formal Languages and Computability

Spring 2018

Agenda of Day 13

- Summary of Lecture 11
- Quiz 4
- Lecture 12: Teaching ...
  - Regular Languages

Summary of Lecture 11: We learned ...

NFAs Formal Definition

- Formally, we define NFAs by a quintuple $M = (Q, \Sigma, \delta, q_0, F)$
- Except δ, the rest items are similar to DFAs'.

$$\delta : Q \times (\Sigma \cup \{\lambda\}) \rightarrow 2^Q$$

δ is total function.

Machines and Languages Association

- Every machine has an associated language.
- BUT we do NOT know yet whether or not for every language, we can construct a machine!

DFAs vs NFAs

- What is power?
- Automata class A is more powerful than class B iff ...
  - ... the set of languages recongnized by class B is a proper subset of the set of the languages recognized by class A.

Theorem

- The set of languages accepted by NFAs are equal to the set of languages accepted by DFAs.
- NFAs and DFAs have the same power.

Any question?

Quiz 4

No Scantron Needed!

Introduction

Example 1

- Design a DFA/NFA to recognize our famous language:
- $L = \{a^n b^n : n \geq 0\}$ over $\Sigma = \{a, b\}$
- You're struggling!
- Let's forget about this, and take a simpler example!

Example 2

- Design a DFA/NFA to recognize the following language:
- $L = \{ww^R : w \in \Sigma^*\}$ over $\Sigma = \{a, b\}$

You!

Me!

Why We Could NOT Construct Machines

- After some struggling, we realized that we could not construct such machines.

What is the reason?

- Because we need to store some info from one part of the construction and use that info to construct the other parts.

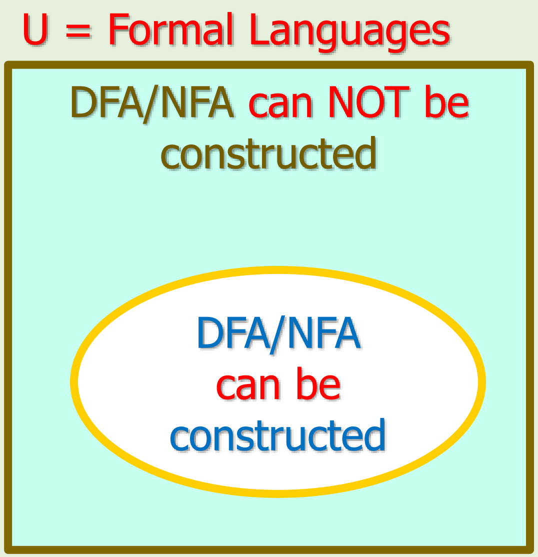
- In example 1, to make sure that the number of a's and b's are equal, we need to count and store the number of a's somehow.
- But we cannot implement counters by DFAs/NFAs!
  - They don't have memory!

- How about example 2?
- Explain why we could not construct the machine?

Categorizing Formal Languages

- We just realized that there are different kinds of formal languages.
  - Some languages are more complex than the others.
- To study formal languages, we need to categorize them.

- Up to this point, we've realized that ...
  - We can construct a DFA/NFA for some languages while we cannot for the others.

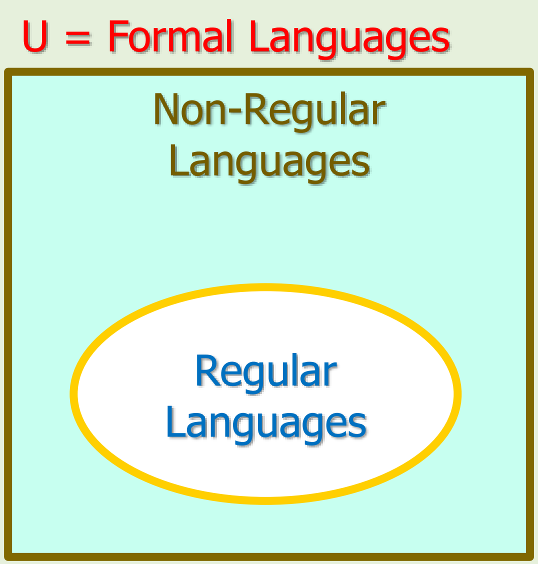


- Let's give a name to these two categories!

Regular Languages

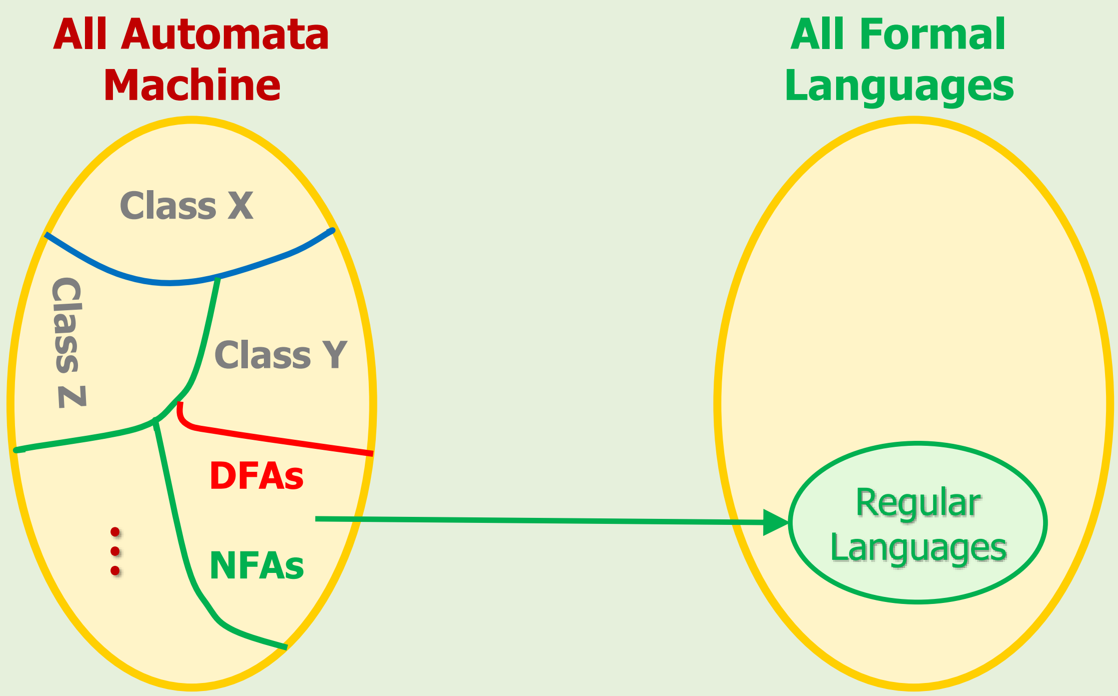
Definition

- A language L is called regular iff there exists a DFA/NFA to recognize it.
- Note that this definition has two sides:
  - If there exists a DFA/NFA, then its associated language is regular.
  - If L is regular, then there exists a DFA/NFA to recognize it.



Machines and Languages Association

- We already saw the association between machines and languages.
- Now we have a name for the languages that DFAs/NFAs recognize.



Categorizing Formal Languages

- Recall that before, we categorized formal languages as "finite" and "infinite".
- And this is our second categorization:
  1. "Regular Languages", and
  2. "Non-Regular Languages"
- Note that the correct English word is "irregular" but in computer science we use "non-regular".
- How can we prove that a language is regular?
- We need to construct a DFA/NFA for it.

Regular Languages

Example 3

- Which of the following languages is regular over $\Sigma = \{a, b\}$?
- $L = \{abbaa\}$
- $L = \{\lambda\}$
- $L = \{ \}$
- $L = \{\lambda, a, abb\}$
- $L = \{a, b\}^*$
- $L = \{a^n b : n \geq 0\}$

- We've already constructed DFAs for almost all of the above languages.
- So, all of them are regular.

Homework

a. Prove that the language $L = \{awa : w \in \{a, b\}^*\}$ is regular.

b. Write a set-builder for L^2.

c. Prove that L^2 is regular.

Recognizing Regular Languages Heuristically

- Sometimes we can heuristically find out whether a language is regular or not. How?
- Let's explain it through some examples.

Example 4

- Which of the following languages are regular?

1. $L = \{ab\ w : w \in \Sigma^*\}$ over $\Sigma = \{a, b\}$
2. $L = \{w\ w : w \in \Sigma^*\}$ over $\Sigma = \{a, b\}$
3. $L = \{w\ abb\ w : w \in \Sigma^*\}$ over $\Sigma = \{a, b\}$
4. $L = \{1^{2k} : k \geq 0\}$ over $\Sigma = \{1\}$
5. $L = \{1^n + 1^m = 1^{n+m} : n, m \geq 1\}$ over $\Sigma = \{1, +, =\}$ (Unary addition)

Finite Languages

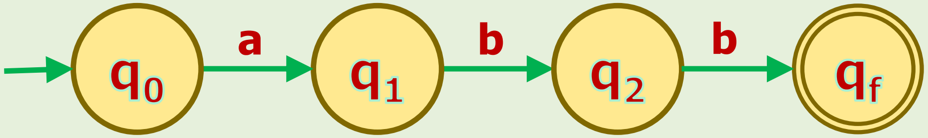
Finite Languages

Theorem

- All "finite" languages are regular.

Proof

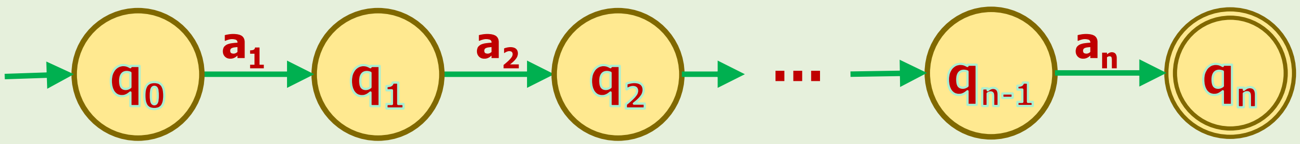
- To prove this theorem, we need to construct an NFA for a general finite language $L = \{w_1, w_2, ..., w_n\}$
  - Where $w_j \in \Sigma^*$ for $j = 1, 2, ..., n$.
- We know that strings are finite sequence of symbols.
- So, we can construct a separate NFA for every string.
- For example if w = abb, then the following DFA can recognize it.



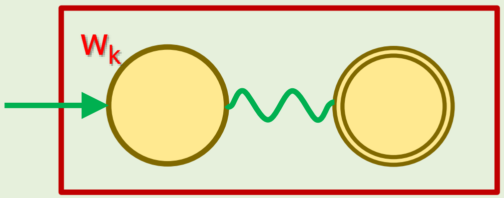
Finite Languages Are Regular

Proof (cont'd)

- Let $w_k = a_1\ a_2 \ldots a_m$ be a general string where $a_i \in \Sigma$ for $i = 1, 2, \ldots, m$.
- We can construct the following NFA to recognize w_k.



- For simplicity, let's put the NFA in a box ...

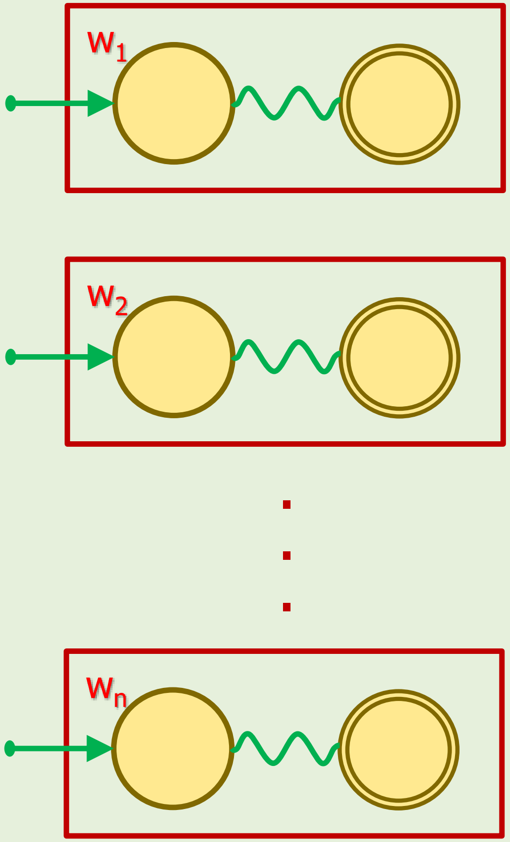


- So, this NFA can accept the string w_k.

Finite Languages Are Regular

Proof (cont'd)

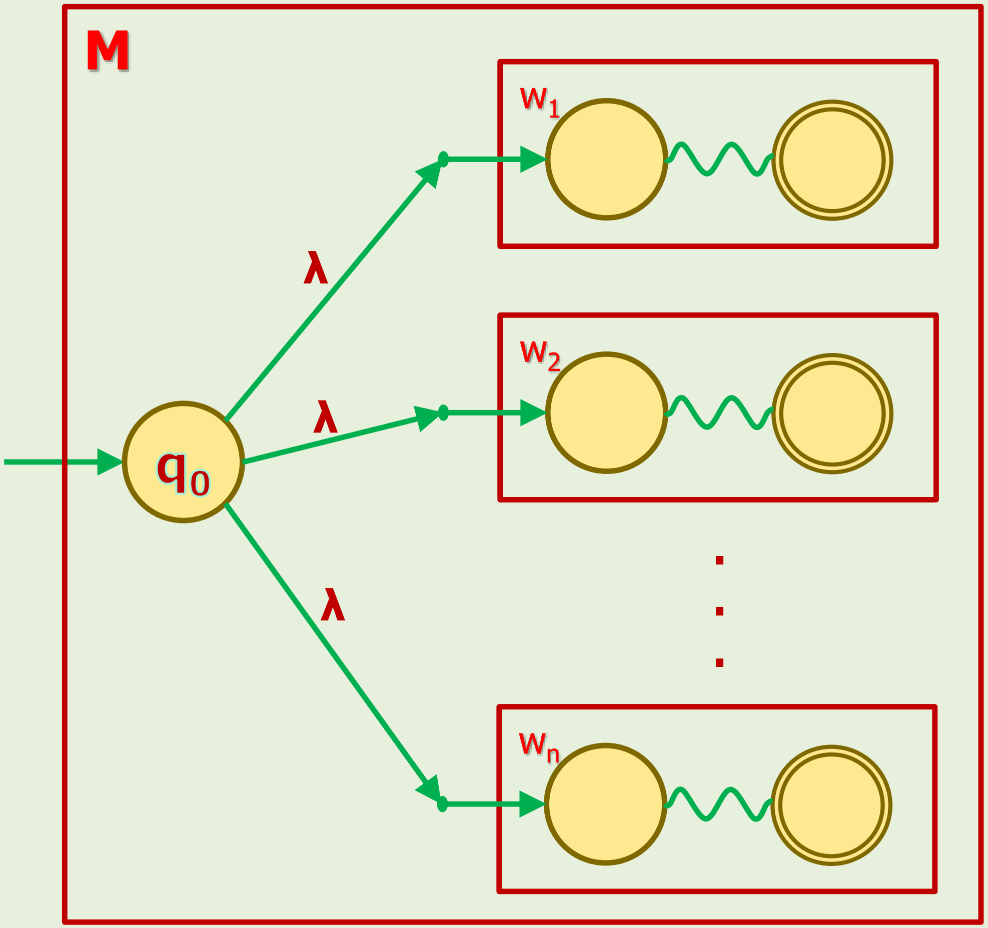
- So, in the similar way, we can construct an NFA for every w_j in the language.
- We should combine these simple NFAs and construct an NFA to recognize L.
- But how?



Finite Languages Are Regular

Proof (cont'd)

- We can combine them by using λ-transitions.
- This new machine recognizes L.
- Explain why?
- Since L is a general finite language, so, we proved all finite languages are regular.



Non-Regular Languages Are Infinite

- The contrapositive of every theorem is also true.

Recap: Contrapositive

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

- The theorem we just proved:

If L is finite, then L is regular.

- L is finite. $\equiv$ f
- L is regular. $\equiv$ r

$$\mathbf{f \rightarrow r \equiv \sim r \rightarrow \sim f}$$

- Translation:

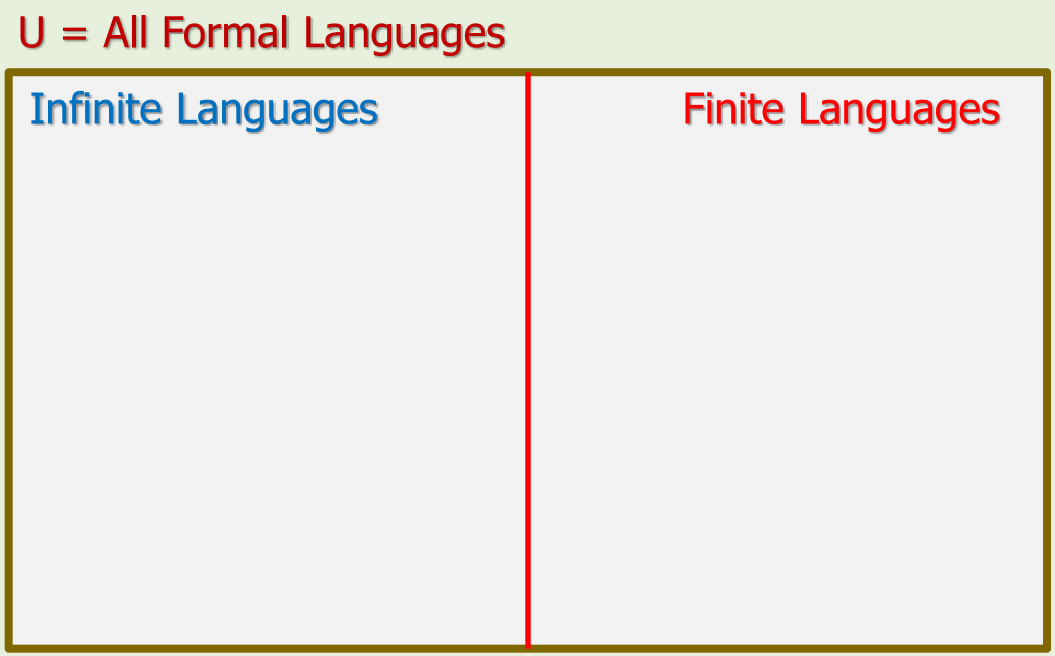
If L is non-regular (= not regular), then L is infinite (= not finite).

- The compact version:

All non-regular languages are infinite.

Categories of Formal Languages

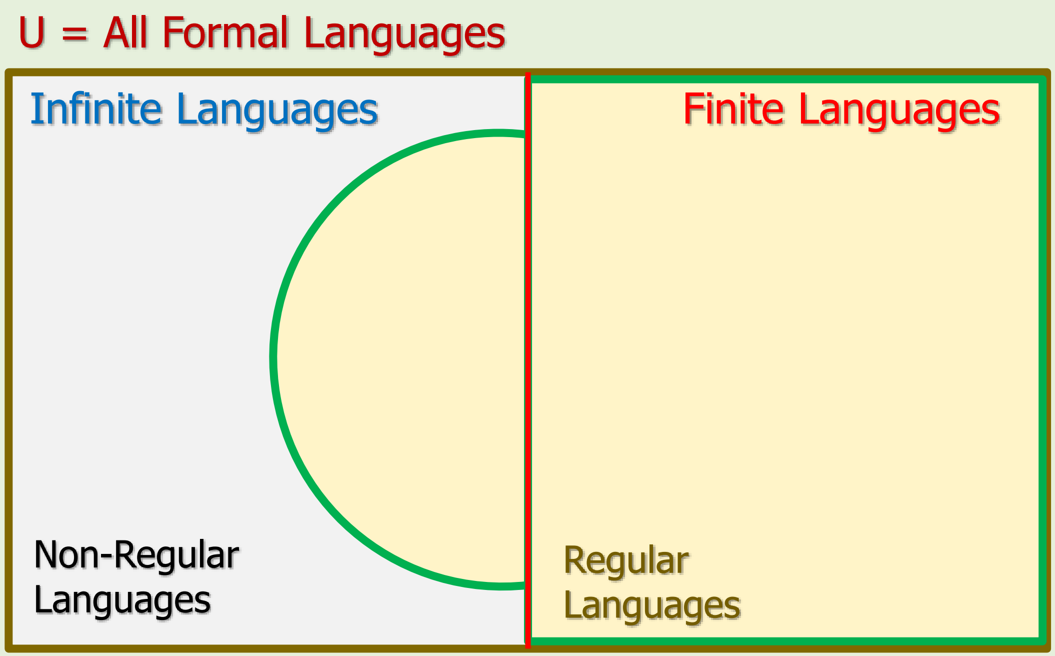
1st Categorization: Finite and Infinite



- Where would you locate "regular" and "non-regular" languages?

Categories of Formal Languages

2nd Categorization: Regular and Non-Regular



Closure Properties of Regular Languages

Theorem

- If L, L_1 and L_2 are all regular languages, then:

Union	$L_1 \cup L_2$
Concatenation	$L_1 L_2$
Star-Closure	L^*
Reversal	L^R
Complement	$\overline{L}$
Intersection	$L_1 \cap L_2$
Minus	$L_1 - L_2$

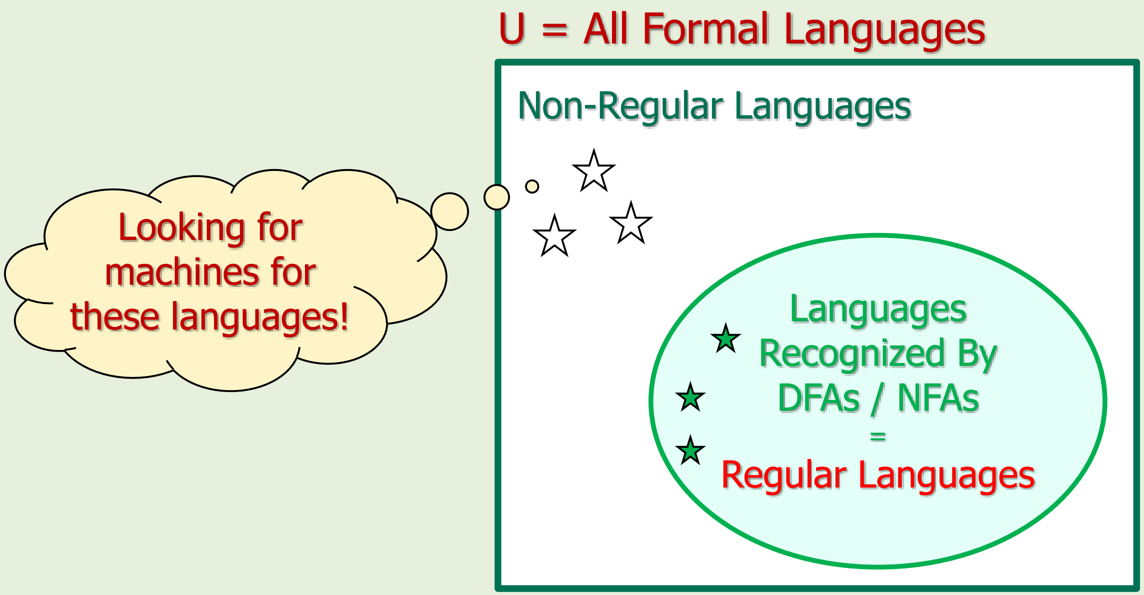
are regular languages too.

- It means: The family of "regular languages" is closed under the above operations.

What is the Next Step?

Conclusion

- NFAs and DFAs recognize "regular languages".
- The next step is to define a new class of machines that recognizes "non-regular languages".



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. Michael Sipser, "Introduction to the Theory of Computation, 3rd ed.," CENGAGE Learning, United States, 2013
   ISBN-13: 978-1133187790