

Welcome to CS622!
Theory of Formal Languages

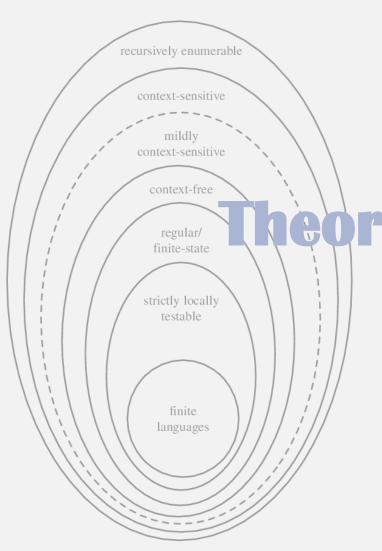
UMass Boston Computer Science

Instructor: Stephen Chang

Fall 2021

Lecture Logistics

- This is a <u>remote</u> class!
 - At least until Sept 30th
- Lectures will be recorded and posted to Blackboard/Echo360
 - Slides will typically be posted to the course web page before class
- Type questions into Zoom's chat
 - Don't use the hand raise feature
 - Please be patient since I may only monitor occasionally
- Keep audio and video off normally
- I may call on students randomly during lecture
 - Turn on audio and video when speaking
 - Please be presentable
- Quiz (5min) at end of every lecture (on gradescope)



Welcome to CS 622!
heory of Formal Languages

UMass Boston Computer Science Instructor: Stephen Chang Fall 2021

What's a "language"???

What's a "language"?



What's a "language"?

```
public class Main {
  public static void main(String[] args) { object HelloWorld extends App {
     System.out.print("hello world");
                                                 println("hello world")
                                 program hello }
    <?xml version="1.0"?>
    <greeting>hello world</greeting> print *, "hello world" .model small
                                 end program hello
                                                           .stack 100h
  main = putStrLn "hello world"
                <?php echo 'hello world' ?>
                                                           msg db 'hello world$'
package main
import "fmt"
                       <?xml version="1.0"?>
func main() {
  </application> 68 65 6c 6c 6f 20 77 6f 72 6c 64 mov ax, 4C00h int 21h
createTextField("hello", 0, 0, 0, 100, 100); @echo off
hello.text = "hello world"
            IDENTIFICATION DIVISION. Imports System
PROGRAM-ID. HELLO-WORLD.
PROCEDURE DIVISION. Sub Main() write('hello world'),nl.
Console.WriteLine ("hello world")
<html><body> PROCEDURE DIVISION.
SELECT 'hello world' AS hello halt;
                                   (null, "hello world"); KTHXBYE
#include <cstdio>(let ((hello-world (lambda() (display "hello world")(newline))))
int main() {
    (hello-world))
   puts("hello world"); program Helloworld; puts 'hello world'
 int main() {
                           writeln('hello world'); Print["hello world"]
                                      <cfoutput>hello world</cfoutput>
  $(function(){$('body').text('hello world');});
Transcript show: 'hello world' public class Hello {
                                        public static void Main() {
   System.Console.WriteLine("hello world");
        .... ... #!/usr/bin/lua
              print("hello world"), document.write('hello world');
                     <% HelloWorldLabel.Text = "hello world"; %>
                    , <html><body>
  <asp:Label runat="server" id="HelloWorldLabel"></asp:Label>
                        int main (int argc, const char * argv[])
                          NSLog (@"hello world"); with Ada.Text_IO; use Ada.Text IO;
                                                     procedure Hello is
                             begin
#include <iostream> Put_Line("hello world");
                                                    end Hello;
                              int main() {
           'hello world'); cout << "hello world" << endl;
```



The Formal Definition of a Language

• A language is a (possibly infinite) set of strings

• A **string**/word is a (finite) sequence of chars from an <u>alphabet</u>

• An **alphabet** is a (finite, non-empty) set of chars/symbols

The Formal Definition of a Language

- A **language** is a (possibly infinite) set of <u>strings</u>
 - <u>E.g.</u>, the set of all binary numbers
 - all Python programs
 - all words in English dictionary
 - Σ^* = language of all possible strings over alphabet Σ
 - For all languages L over alphabet Σ , $L \subseteq \Sigma^*$
- A string/word is a (finite) sequence of chars from an <u>alphabet</u>
 - <u>E.g.</u>, **010101**
 - hello
 - ϵ (sometimes λ) is the empty string (length zero string)
- An **alphabet** is a (finite, non-empty) set of chars/symbols
 - <u>E.g.</u>, **{0, 1}** (binary digits, the alphabet of computers)
 - {a, b, ..., z} (lowercase letters)
 - set of ASCII chars
 - Alphabets are often denoted with the Σ symbol

recursively enumerable

context-sensitive

mildly
context-sensitive

context-free

Theory of Formal Languages

strictly locally

(In mathematical logic)

A theory consists of:

- Axioms
 - Accepted facts and definitions
- Theorems
 - I.e., additional facts derived from axioms and previous theorems
 - Using a deductive system
 - I.e., "if p then q" and "p", then "q" (modus ponens)

Each of these language classes corresponds to a different kind of computer!

Why study languages formally?

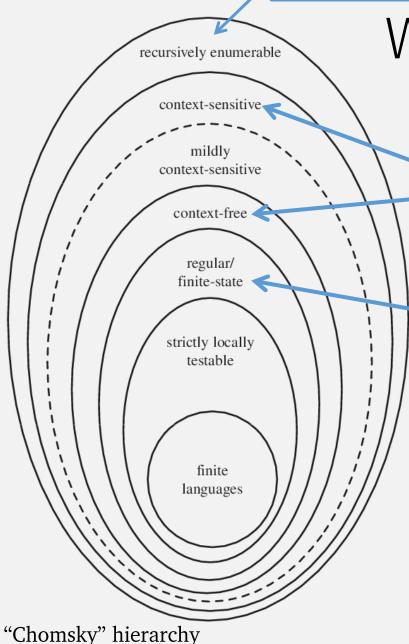
- 1. To communicate with computers!
 - We need to know what "languages" they can understand
 - <u>E.g.</u>, Python, C++, Java programs?
- Simpler languages are often more convenient
 - <u>E.g.</u>, text search, arithmetic



Don't want to write Python here

• Different languages require different computing power to understand/recognize

So the formal study of **languages** is also the formal study of **computation**!



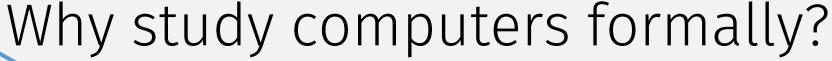
Why study computers formally?

- 2. To predict what programs will do
 - (without running them!)

```
unction check(n)
 // check if the number n is a prime
 var factor; // if the checked number is not a prime, this is its first factor
  // try to divide the checked number by all numbers till its square root
  for (c=2; (c <= Math.sqrt(n)); c++)
     if (n%c == 0) // is n divisible by c?
        { factor = c; break}
  return (factor);
   // end of check function
unction communicate()
             // i is t
                         checked number
  var factor; // if the
                        necked number is not prime, this is its first factor
  i = document.primetes
                        number.value;
                                               t the checked number
  // is it a valid inpu
 if ((isNaN(i)) || (i • 0) || (Math.floor(i = i))
   { alert ("The checked ect should be a de positive number")} ;
  else
    factor = check (i);
    if (factor == 0)
       {alert (i + " is a prime")} ;
                                                          + "X" + i/factor) }
      // end of communicate function
```



???



- 3. To know the <u>limits</u> of computers
 - <u>I.e.</u>, what they can't do



- More practically, resource-limited computers
 - <u>I.e.</u>, what can I compute with ...
 - · ... a certain amount of time?



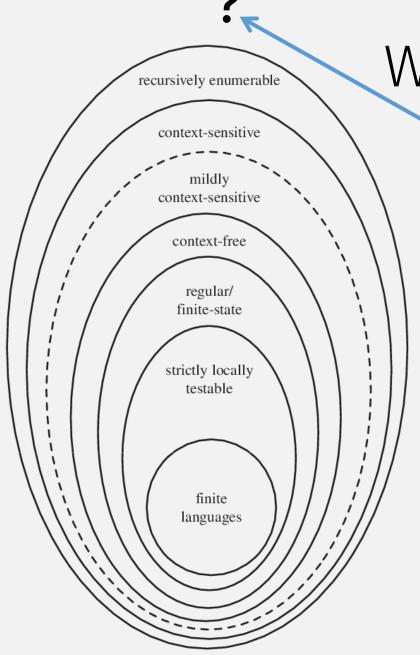
... a certain amount of memory space?



- ... a certain probability?
- ... a limited circuit size?









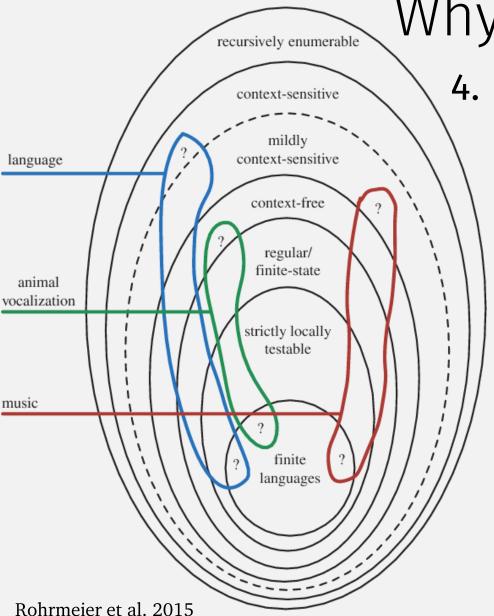
4. Many, many <u>practical</u> applications

• E.g., Can we formally model ...

• ... human spoken language?

• ... animal communication?

• ... music?



More Practical Applications

Writing secure software:



The LANGSEC (Language-Theoretic Security) community posits that the only path to trustworthy software ...

... is treating all valid inputs as a **formal language** ...

... where input-parsing is handled by **automata** with the required computation power.

---- langsec.org

Applications of Formal Langs: Beyond CS

Lindenmeyer grammars model plant growth

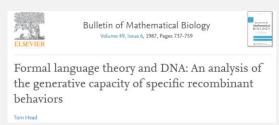
variables : X F constants : + - [] start : X rules : (X \rightarrow F+[[X]-X]-F[-FX]+X), (F \rightarrow FF) angle : 25°

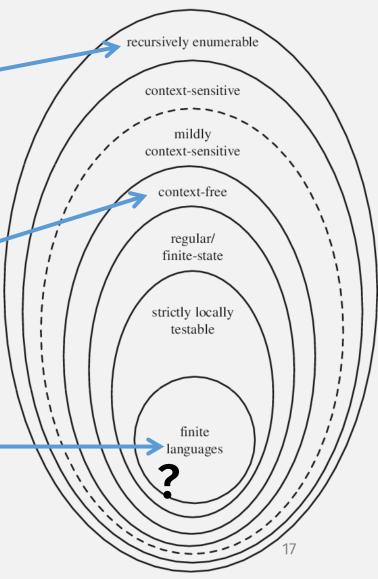
Many fractal patterns in nature are CFGs





DNA has its own formal language





Computation Models Languages In this class Turing machines recursively enumerable Linear bounded automaton context-sensitive mildly Embedded bounded automaton context-sensitive Pushdown automata context-free We'll start here regular/ Finite state automata finite-state strictly locally Local automata testable finite languages

CS420



CS622

- Deeper topics, faster paced
- More proofs (so brush up on CS220/CS320)





Course Logistics

Course website:

https://www.cs.umb.edu/~stchang/cs622/f21/

Quiz 9/8
See gradescope.com