Welcome to CS622!

Theory of Formal Languages

UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2021
Lecture Logistics

• This is a remote 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!

Theory of Formal Languages

UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2021

What’s a “language”???
What’s a “language”?
What’s a “language”?
Welcome to CS 622!

Theory of Formal Languages

UMass Boston Computer Science
Instructor: Stephen Chang
Fall 2021

“Defined mathematically”
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 **alphabet**.

- 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 **strings**
  - E.g., the set of all binary numbers
  - all Python programs
  - all words in English dictionary
  - \( \Sigma^* = \) language of all possible strings over alphabet \( \Sigma \)
    - For all languages \( L \) over alphabet \( \Sigma \), \( L \subseteq \Sigma^* \)
- A **string/word** is a (finite) sequence of chars from an **alphabet**
  - E.g., \( 010101 \)
  - hello
  - \( \varepsilon \) (sometimes \( \lambda \)) is the empty string (length zero string)
- An **alphabet** is a (finite, non-empty) set of chars/symbols
  - E.g., \( \{0, 1\} \) (binary digits, the alphabet of computers)
  - \( \{a, b, \ldots, z\} \) (lowercase letters)
  - set of ASCII chars
  - Alphabets are often denoted with the \( \Sigma \) symbol
Welcome to CS 622!

Theory of Formal Languages

(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)
Why study languages formally?

1. To communicate with computers!
   - We need to know what “languages” they can understand
     - E.g., Python, C++, Java programs?

2. Simpler languages are often more convenient
   - E.g., text search, arithmetic

3. Different languages require different computing power to understand/recognize

"Chomsky" hierarchy

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!)

```javascript
function check(n) {
  var factor; // if the checked number is not a prime, this is its first factor
  var c = 6; // 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.
}

function communicate() {
  // communicate with the user
  var number; // i.e. the checked number
  var factor; // if the checked number is not a prime, this is its first factor
  var i = document.primes.value; // get the checked number
  // is i a valid input?
  if (isNaN(i) || i < 1 || Math.floor(i) != i) {
    alert("The checked input should be a single positive number");
  } else {
    factor = check(i);
    if (factor == 0)
      alert("i + " + i + " is a prime");
    else
      alert("i + " + i + " is not a prime, " + i + " = " + factor + " x " + (i/factor));
  }
} // end of communicate function
```
Why study computers formally?

3. To know the limits of computers
   • I.e., what they can’t do

   • More practically, resource-limited computers
     • I.e., what can I compute with ...
       • ... a certain amount of time?
       • ... a certain amount of memory space?
       • ... a certain probability?
       • ... a limited circuit size?
Why study languages formally?

4. Many, many **practical** 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

\[
\begin{align*}
\text{variables: } & X, F \\
\text{constants: } & +, -, \{, \} \\
\text{start: } & X \\
\text{rules: } & (X \rightarrow F+[X]-X[-F]X), (F \rightarrow FF) \\
\text{angle: } & 25^\circ
\end{align*}
\]

• Many fractal patterns in nature are CFGs

• DNA has its own formal language
In this class

We’ll start here

Languages
- recursively enumerable
- context-sensitive
- mildly context-sensitive
- context-free
- regular/finite-state
- strictly locally testable
- finite languages

Computation Models
- Turing machines
- Linear bounded automaton
- Embedded bounded automaton
- Pushdown automata
- Finite state automata
- Local automata
CS420 vs 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