

CS311 – Computational Structures

**What's it all about
or
Why do I need those
prerequisites, anyway?**

Lecture 1

The Geography Game

- How could you write a computer program to play the geography game?

What's the course about?

- “Formal” Languages, and computers that “recognize” languages.
- What computers can do, and what they *can't* do
- What's easy to compute, and what's hard

What's the course about?

- “Formal” Languages, and computers that “recognize” languages.
- What computers can do, and what they *can't* do
- What's easy to compute, and what's intractable

Major Topics

- Regular languages and regular expressions, deterministic and nondeterministic automata, constructing efficient automata, topics (regular grammars, pumping lemma).
- Context-free languages and pushdown automata, topics (transforming grammars, pumping lemma).
- Turing machines, Church-Turing thesis, equivalent models of computation.
- Computability: undecidable problems, hierarchy of languages, complexity (P, NP, PSPACE, completeness).

Course Objectives

Upon the successful completion of this course students will be able to:

1. Find regular grammars and context-free grammars for simple languages whose strings are described by given properties.
2. Apply algorithms to: transform regular expressions to NFAs, NFAs to DFAs, and DFAs to minimum-state DFAs; construct regular expressions from NFAs or DFAs; and transform between regular grammars and NFAs.
3. Apply algorithms to transform: between PDAs that accept by final state and those that accept by empty stack; and between context-free grammars and PDAs that accept by empty stack.
4. Describe LL(k) grammars; perform factorization if possible to reduce the size of k; and write recursive descent procedures and parse tables for simple LL(1) grammars.
5. Transform grammars by removing all left recursion and by removing all possible productions that have the empty string on the right side.

6. Apply pumping lemmas to prove that some simple languages are not regular or not context-free.
7. State the Church-Turing Thesis and solve simple problems with some of the following models of computation: Turing machines (single-tape and multi-tape); while-loop programs; partial recursive functions; Markov algorithms; Post algorithms; the lambda calculus; and Post systems.
8. Describe the concepts of unsolvable and partially solvable; state the halting problem and prove that it is unsolvable and partially solvable; and use diagonalization to prove that the set of total computable functions cannot be enumerated.
9. Describe the hierarchy of languages and give examples of languages at each level that do not belong in a lower level.
10. Describe the complexity classes P, NP, and PSPACE.
11. Use an appropriate programming language as an experimental tool for testing properties of computational structures.

Things to Do

- Register for the course!
- Read the **class web page**
 - <http://www.cs.pdx.edu/~black/CS311>
- Sign up for the email list
- Make sure that you have **Blackboard access**
 - <http://bb.pdx.edu/>
- Do the first homework assignment.

Warning!

- I expect you to read the class web page
- I expect you to read the class email list
- I expect you to read the textbook

If you can't or won't or don't have time to read, you will have trouble with this course.

Onward!

Regular Languages

- *A language* is a set of strings over an *alphabet*
- Suppose that the alphabet $A = \{x, y, z\}$

The Regular Languages

1. \emptyset , $\{\Lambda\}$, and $\{a\}$ are regular languages for all $a \in A$.
2. If L and M are regular languages, then the following languages are also regular: $L \cup M$, ML , and L^* .

Hein, §11.1 (p 634)

Regular Expressions

The Regular Expressions

1. Λ , \emptyset , and a are regular expressions for all $a \in A$.
2. If R and S are regular expressions, then the following expressions are also regular: (R) , $R + S$, $R \cdot S$, and R^* .

- Regular expressions (so far) are just formulae involving the operations $+$, \cdot and * .
- Next, let's give them *meaning*