Computer Science Theory
COMS W3261
Lecture 24

Alexander Roth

2014 – 12 – 03

What will be on the test?

1. 2 Regular Languages
2. 2 Context-Free Languages
3. 2 Computability Problems
4. 2 Complexity Problems
5. 1 Lambda Calculus
6. 1 Surprise

- Know the basic facts. Nothing will be esoteric

What have we learned in this course?

- Computational Thinking: How we go about solving problems in the age of the Internet.
- We should find an appropriate model of computation, so that problems in that area can be solved using algorithms and computational steps.
- Look at the Internet.
- Model of Computation: A mathematical abstraction, in which problems can be posed and solved using sequences of algorithmic steps. Some of the enduring models of life.

Problem Areas

1. Finding patterns in strings
    - Applications
    - Textual analysis
    - NLP
    - Compliling
    - Genomic Analysis

Problem

Given a description of a set of strings or a regular expression R and an input string w, is w in $L(R)$?

R to the MYT-Algorithm to the ϵ-NFA takes time complexity $O(|R| \times |w|)$.

R to the MYT-Algorithm to the ϵ-NFA to the subset construction to a DFA, takes time complexity $O(2^|R| + |w|)$

What to know

- Pumping lemma for regular languages
- Closure properties
- Decidability results.

The Universe of Languages

1. Regular languages
2. Context Free Languages
3. Recursive Langauges (Algorithms)
4. Recursively Enumerable Languages (Turing Machine Recognizable, either halts or goes into an infinite loop)
5. All Languages on $\{0, 1\}$ (Contains uncountably infinite languages)

Algorithm A Turing machine that halts on all inputs.

All languages are countably infinite.

We don't have a pumping lemma for Recursively Enumerable languages. Instead, we use the diagonalization language L_d to show that a language cannot be recursively enumerable. We have the universal language for Recursively Enumerable L_u. Recursive languages can be proving with the Turing machine.

We say a problem is undecidable if it's either recursively enumerable or unrecursively enumerable.

Complexity Theory

To show that SAT is NP-Complete, we need to show that SAT is NP, and that every language in NP is reducible to SAT in polynomial time. That is, there is an algorithm that must take polynomial time that will map every language in NP to SAT. We are restricting the reductions to those that can be done with a deterministic Turing machine in polynomial time.

If L_1 is NP-Complete, and L_2 is in NP and there is a polynomial time reduction of L_1 to L_2, then L_2 is NP-Complete.

Is P = NP?