CS 3313
Foundations of Computing:

Course Summary

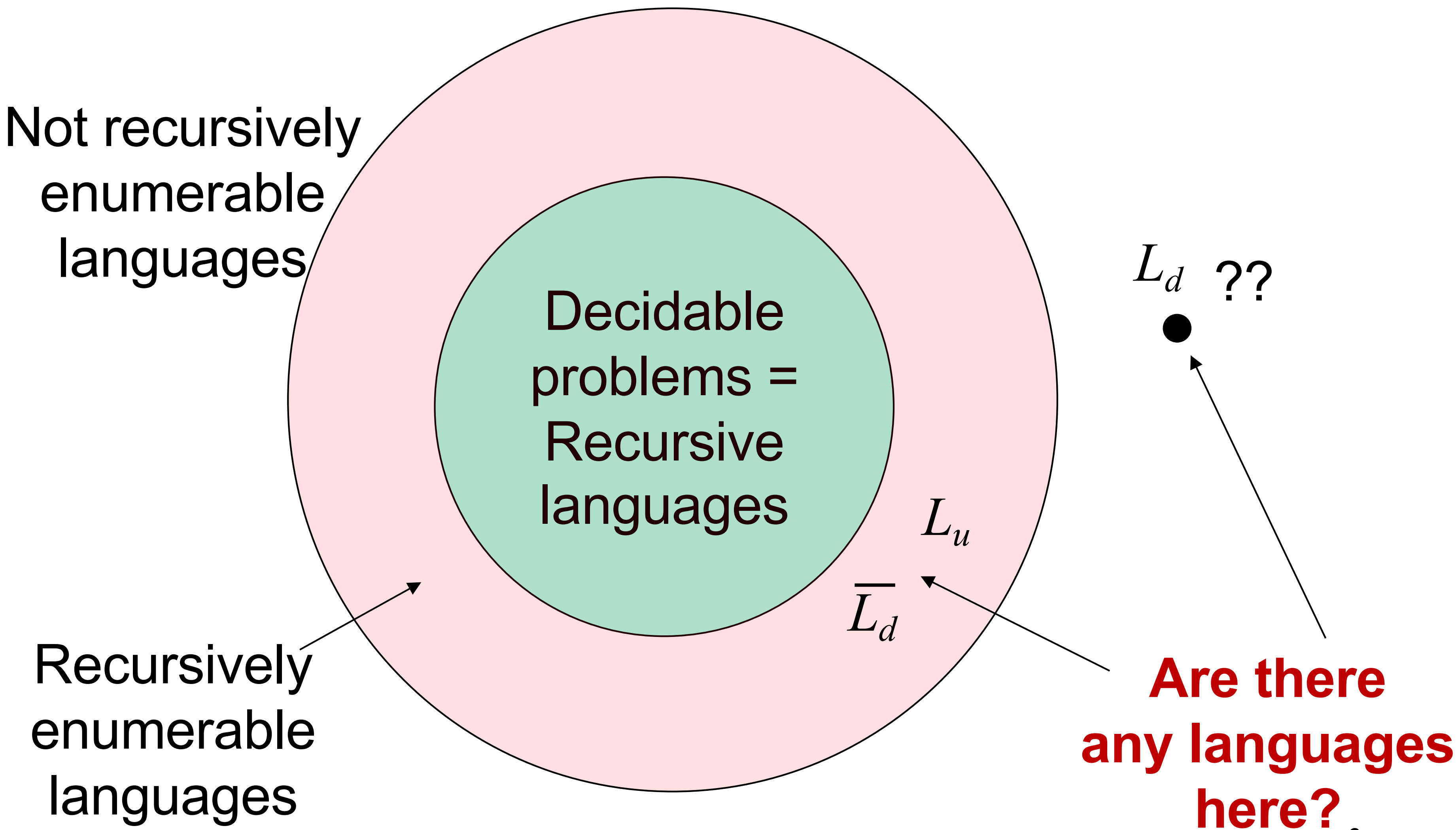
http://gw-cs3313-2021.github.io

Decidable Problems

- A problem is *decidable* if there is an algorithm to answer it.
  - Recall: An "algorithm," formally, is a TM that halts on all inputs, accepted or not.
  - Put another way, "decidable problem" = "recursive language."
- Otherwise, the problem is *undecidable*.

- Language is recursive if it is accepted by a TM that halts on all inputs.
- Language is recursively enumerable (r.e.) if it is accepted by a TM
  - TM halts and accepts if the string is in the language
  - However, TM may not halt if the string is not in the language

Undecidable Problems



A key proof technique: Reducability

- Reducibility of a problem A to problem B
- Given two problems A and B,

  problem A is <u>reducible</u> to problem B if an algorithm for solving B can be used to solve problem A

  – Therefore, solving A cannot be harder than solving B

  • *If A is undecidable and A is reducible to B, then B is undecidable*
- Idea: If you had a black box that can solve instances of B, can you solve instances of A using calls to this Black box.
  - The black box is the assumed Algorithm for B.

Today….

- Review our undecidability results/proofs
  - One more example
- The Post correspondence problem
- Rice’s Theorem: Undecidability properties of r.e. languages

Our current "collection" of undecidable languages

1. We proved that L_d *and* $\overline{L_d}$ *are not decidable*
2. $L_u = \{ <M,w> \mid M$ accepts $w \}$ ....*Halting Problem*
3. Does M halt on all inputs is undecidable
4. $L_e = \{ <M> \mid L(M) = \emptyset \}$ – Given any TM M, does M accept empty set.
5. Subset: Given two Turing machines is Language accepted by M_1 a subset of language accepted by M_2 - $\{ <M1,M2> \mid L(M_1) \subseteq L(M_2) \}$
6. Blank Tape acceptance: *{ <M> | M halts on blank tape}*
7. Testing printing of symbol: *{ <M> | M prints out specific symbol a on the tape}*

Properties of recursively enumerable languages

- L is r.e. if it is accepted by a TM, L= L(M) for some TM M.
- Let $\mathcal{P}$ be a set of *r.e.* languages, each is a subset of {0,1}* (or any alphabet) – $\mathcal{P}$ is said to be a property of *r.e.* languages.
- a set L has property $\mathcal{P}$ if L is an element of $\mathcal{P}$
- In terms of properties of the language accepted by a turing machine, let $L_{\mathcal{P}} = \{ \langle M \rangle \mid L(M) \text{ is in } \mathcal{P} \}$
- $\mathcal{P}$ is a *trivial property* if $\mathcal{P}$ is empty or $\mathcal{P}$ consists of all *r.e.* languages
  - All languages satisfy this property or none of them satisfy this property
- $\mathcal{P}$ is a *non-trivial property* otherwise.
  - Some languages satisfy this property, and some do not.

Trivial and Non-trivial Properties

- Non-trivial property: refers to a property satisfied by some but not all r.e. languages
- Trivial property: property satisfied by all or none (of r.e. languages)

- More formally:
- $\mathcal{P}$ is a *trivial property* if $\mathcal{P}$ is empty or $\mathcal{P}$ consists of all *r.e.* languages
- $\mathcal{P}$ is a *non-trivial property* otherwise.

Rice's Theorem

- Rice's Theorem: Any non-trivial property **P** of r.e. languages is undecidable.

- So how does one use this result……
  - Observe that this theorem is about r.e. languages -- languages which are accepted by a TM
    - We can give a TM to accept a language in this set of languages
- Ex: Determining if $\{<M> \mid L(M) \text{ is finite}\}$ is undecidable.
- Proof –to apply Rice's theorem, we have to show that this is a non-trivial property..How ?
  - Provide a TM M_1 such that $L(M_1)$ has this property – $L(M_1)$ is finite.
  - Provide a TM M_2 such that $L(M_2)$ does not have this property – $L(M_2)$ is infinite.
  - Therefore the property is non-trivial.

Properties of CFL languages…and another important Undecidable Problem

The Post Correspondence Problem (PCP)

- Given two sequences of n strings on some alphabet Σ, for instance

  $A = w_1, w_2, \ldots, w_n$ and $B = v_1, v_2, \ldots, v_n$

  there is a Post correspondence solution (PC solution) for the pair (A, B) if there is a nonempty sequence of integers

  $i, j, \ldots, k$, such that $w_i w_j \ldots w_k = v_i v_j \ldots v_k$

- Example: assume A,B are

  $w_1 = 11,$ $w_2, = 10111,$ $w_3 = 10$

  $v_1 = 111$ $, v_2, = 10,$ $v_3 = 0$

  solution for this instance of (A, B) exists: sequence 2113

  $w_2 w_1 w_1 w_3 =$ 1011111110

  $v_2 v_1 v_1 v_3 =$ 1011111110

The Undecidability of the Post Correspondence Problem

- The Post correspondence problem is to devise an algorithm that determines, for any (A, B) pair, whether or not there exists a PC solution
- For example, there is no PC solution if A and B consist of

$w_1 = 00,\ w_2, = 001,\ w_3 = 1000$ and $v_1 = 0,\ v_2, = 11,\ v_3 = 011$

- **Theorem:** the Post correspondence problem (PCP) is undecidable
- result is crucial for showing the undecidability of various problems involving context-free languages

Undecidable Problems for Context-Free Languages

- The Post correspondence problem is a convenient tool to study some questions involving context-free languages
- The following questions, among others, can be shown to be undecidable
  - Given an arbitrary context-free grammar G, is G ambiguous?
  - Given arbitrary context-free grammars G_1 and G_2,
    is $L(G_1) \cap L(G_2) = \varnothing$?
  - Given arbitrary context-free grammars G_1 and G_2,
    is $L(G_1) = L(G_2)$?
  - Given arbitrary context-free grammars G_1 and G_2,
    is $L(G_1) \subseteq L(G_2)$?

Computability/Decidability - Summary

- Turing machines are capable of implementing any algorithm that can be implemented on today's von Neumann model of general purpose computers….Turing-Church Thesis
- Non-deterministic and Deterministic Turing machines have the same ”power” in terms of what they can solve.
- It is not known is NDTM and DTM are equivalent in time efficiency….the P=NP problem.
- There are problems that cannot be solved using Turing machines – these are undecidable (unsolvable) problems.
  - Reducibility is a technique to show a problem is undecidable by reducing a known undecidable problem to this problem.

Automata, Grammars, Languages….structure?

- Different models of automata: DFA, PDA, TM
  - With increasing "power"
- Grammars to define languages….
  - Regular grammar = DFA
  - Context Free Grammar = PDA
- How do they relate to each other……Chomsky Hierarchy

Grammars: Definition

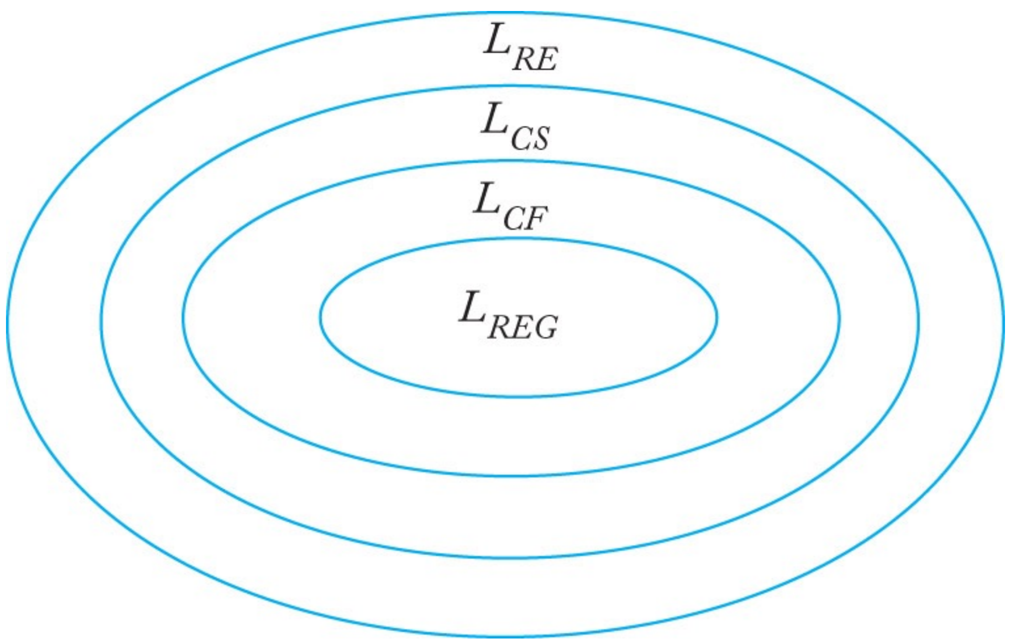
- Definition: A grammar G (V,T,P,S) consists of:

  V – variables, T – terminals, S – start variables

  P – set of production rules
- By placing constraints on the type of production rules we get different classes of grammars
- Unrestricted grammar: Production rule is of the form $x \longrightarrow y$ where $x,y \in (V \cup T)^+$
- Context Sensitive: $|x| < |y|$, $x,y \in (V \cup T)^+$
- Context Free: $x \in V$ and $y \in (VUT)^*$
- Regular grammars: $x \in V$ and at most one variable in y

The Chomsky Hierarchy

- The linguist Noam Chomsky summarized the relationship between language families by classifying them into four language types, type 0 to type 3 -- the *Chomsky Hierarchy*



Automata Models and The Chomsky Hierarchy

- Theorem: If G is a Regular grammar then L(G) is accepted by a DFA.
  - If L is accepted by a DFA then L =L(G) for some regular grammar G.
- Theorem: If G is a context free grammar, then L(G) is accepted by a PDA.
  - If L is accepted by a PDA then L =L(G) for some CFG G
- Theorem: If G is any unrestricted grammar then L(G) is accepted by a Turing machine.
  - All grammars are unrestricted grammars
- Theorem: If G is a context sensitive grammar then L(G) is accepted by a linear bounded automaton
  - A linear bounded automaton is a subclass of Turing machines

Other Mathematical Foundations of CS….?

- Automata and computability theory is a big part of foundations of CS
- Other mathematical abstractions and techniques ?
- Computational Logic – first order logic
  - Theorem provers
  - Semantics of programs……reasoning
- Recursive function theory and Lambda-Calculus
  - Foundations of functional programming languages
- Formal Learning theory
  - Rooted in mathematical logic and how recursive functions “learn”

This is all old stuff…why bother with foundations..

- Consider one of the high impact fields in CS: Machine learning
- Early machine learning - expert systems
  - Ex: Q&A system

    Ques: Has Paul done his Christmas Shopping yet

    Ans: I don’t know if he has bought anything for his mother.
- Today’s learning built on Statistical Machine Learning
  - Logical Form: *A implies B with probability P*
- Problems that are being seen today ?
  - Where is this “probability P” coming from ?
- So what’s the solution…?
  - Researchers are pushing for considering combination of Reasoning and Statistical ML.

CS 3313 Summary: What was it about ?

- Theoretical foundations of Computer Science
  - It's also a look at history of CS…Concept of computing existed before the first computer was built…
- Answer/Ask fundamental (abstract) questions:
  - What is computation –i.e., how do you define what an algorithm is?
  - mathematical models for different types of computing machines ?
    - Why is this an interesting question ?
  - How do you formally define a language
    - Natural language or Programming language
- Study fundamental limits of computing, and properties of languages
  - Mathematical approach
- Above all: about problem solving using math tools
  - Mathematical puzzles which abstract "real" computational questions

Limits of Computation (of automata): Questions

- Can we use a finite state machine to build a compiler ?
- Are all properties of a programming language (C) captured by a CFG (i.e, a PDA) ?
- Can we design a compiler that will determine for any program, whether the program halts on all inputs
  - Or, will the compiler detect any bugs in the program ?
  - Or, are two programs equivalent ? (do they compute the same function)

Determining the simplest machine to solve a problem…

- Given a problem (language), what is the simplest (efficiency and cost) machine that can solve the problem ?
  - DFA (Finite state machine)
  - PDA
  - Turing Machine

- How do you prove it is the simplest ?
  - Prove it cannot be implemented on a simpler machine

Course Logistics: Final Exam May 4th 5:20pm

- Will try to post updated totals (HW, Quiz) before the Final
  - HW8 due tomorrow..
- Final: 40 points (out of 100 total exam points).
- Contents: Comprehensive but clear focus on Material after Exam 2.
- Format: Part A required, and Part B. - Extra Credit
- 90 minutes for Part A (required questions Part A)
- 30 minutes for Part B (extra credit).
- Separate handin for each part.
- Extra Credit is added to your overall exam score – so this an opportunity to improve your exam/course grade.
- Will schedule a review session for late next week.

Where to next…..

- Junior year: “core CS”
  - Systems – Operating systems
    - Project intensive, lots of C and System skills
  - Algorithms
    - Problem solving skills, data structures, theoretical analysis
- These are more intense courses with more depth
  - Require more independent work….
  - Fewer Teaching Assistants (no LAs!)

Where to next…..

- Augmenting your skills/knowledge. (summer)
  - Work on independent projects – or internships
    - Forming a group and building a project provides structure
  - All your peers in CS are doing this…..classes provide a small part of the overall skillset
  - Examples:
    - programming languages (Python or R or more C?)
    - Software packages
    - Databases – learn more MongoDB (or ArangoDB)…you have the skills needed from DB class (Python, AWS,…)
      - Build a simple data analytics application
    - Web development (JavaScript ?)
    - Mobile app dev – iOS (?) …Event driven programming

Where to next…..

- Rest, Recuperate, Recharge…..
  - 2020-21 Academic year has been anything but easy!