Review

• Highest level: 2 main points

1. Decidability
   – problem solvable by an algorithm = problem is decidable
   – some problems are not decidable (e.g. HALT)

2. Tractability
   – problem solvable in polynomial time = problem is tractable
   – some problems are not tractable (EXP-complete problems)
   – huge number of problems are likely not to be tractable (NP-complete problems)

• Important ideas
   – "problem" formalized as language
     • language = set of strings
   – "computation" formalized as simple machine
     • finite automata
     • pushdown automata
     • Turing Machine
   – "power" of machine formalized as the set of languages it recognizes

• Important ideas (continued):
   – simulation used to show one model at least as powerful as another
   – diagonalization used to show one model strictly more powerful than another
   • also Pumping Lemma
   – reduction used to compare one problem to another
Review

• Important ideas (continued):
  – complexity theory investigates the resources required to solve problems
    • time, space, others…
  – complexity class = set of languages
    – language $L$ is C-hard if every problem in C reduces to $L$
    – language $L$ is C-complete if $L$ is C-hard and $L$ is in C.

A complete problem is a surrogate for the entire class.

Summary

Part I: automata

Finite Automata

• Non-deterministic variant: NFA
• Regular expressions built up from:
  – unions
  – concatenations
  – star operations

Main results: same set of languages recognized by FA, NFA and regular expressions (“regular languages”).

Pushdown Automata

New capabilities:
• can push symbol onto stack
• can pop symbol off of stack
Context-Free Grammars

**Start symbol**: $A$
**Non-terminal symbols**: $A, B$
**Terminal symbols**: $0, 1, \#$
**Production**: $A \rightarrow 0A1$, $A \rightarrow B$, $B \rightarrow \#$

Pushdown Automata

**Main results**: same set of languages recognized by NPDA, and context-free grammars ("context-free languages").

- and DPDA's weaker than NPDA's...

Non-regular languages

**Pumping Lemma**: Let $L$ be a regular language. There exists an integer $p$ ("pumping length") for which every $w \in L$ with $|w| \geq p$ can be written as $w = xyz$ such that:
1. for every $i \geq 0$, $xyz \in L$, and
2. $|y| > 0$, and
3. $|xy| \leq p$.

Pumping Lemma for CFLs

**CFL Pumping Lemma**: Let $L$ be a CFL. There exists an integer $p$ ("pumping length") for which every $w \in L$ with $|w| \geq p$ can be written as $w = uvxyz$ such that:
1. for every $i \geq 0$, $uvxyz \in L$, and
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

Summary

Part II: Turing Machines and decidability

- New capabilities:
  - infinite tape
  - can read OR write to tape
  - read/write head can move left and right
**Deciding and Recognizing**

- TM $M$:
  - $L(M)$ is the language it recognizes
  - if $M$ rejects every $x \notin L(M)$ it decides $L$
  - set of languages recognized by some TM is called Turing-recognizable or recursively enumerable (RE)
  - set of languages decided by some TM is called Turing-decidable or decidable or recursive

**Church-Turing Thesis**

- the belief that TMs formalize our intuitive notion of an algorithm is:
  - The Church-Turing Thesis
    - everything we can compute on a physical computer can be computed on a Turing Machine
  - Note: this is a belief, not a theorem.

**The Halting Problem**

- $H : (M, x) :$ does $M$ halt on $x$?
- The existence of $H$ which tells us yes/no for each box allows us to construct a TM $H'$ that cannot be in the table.

**Decidable, RE, coRE...**

- $\{a^n b^n : n \geq 0\}$
- $\{a^n b^n c^n : n \geq 0\}$

**Using reductions**

- Used reductions to prove lots of problems were:
  - undecidable (reduce from undecidable)
  - non-RE (reduce from non-RE)
  - or show undecidable, and RE
  - or show undecidable, and coRE
  - Rice's Theorem: Every nontrivial TM property is undecidable.
The Recursion Theorem

**Theorem:** Let $T$ be a TM that computes $f_n: \Sigma^* \times \Sigma^* \to \Sigma^*$. There is a TM $R$ that computes the fn: $r: \Sigma^* \to \Sigma^*$ defined as $r(w) = t(w, <R>)$.

- In the course of computation, a Turing Machine can output its own description.

Incompleteness Theorem

**Theorem:** Peano Arithmetic is not complete.

(same holds for any reasonable proof system for number theory)

Proof outline:
- the set of theorems of PA is RE
- the set of true sentences (= Th(N)) is not RE

Summary

Part III: Complexity

- Complexity Theory = study of what is computationally feasible (or tractable) with limited resources:
  - running time
  - storage space
  - number of random bits
  - degree of parallelism
  - rounds of interaction
  - others…

main focus: not in this course

Time and Space Complexity

**Definition:** the time complexity of a TM $M$ is a function $t: \mathbb{N} \to \mathbb{N}$, where $t(n)$ is the maximum number of steps $M$ uses on any input of length $n$.

**Definition:** the space complexity of a TM $M$ is a function $s: \mathbb{N} \to \mathbb{N}$, where $s(n)$ is the maximum number of tape cells $M$ scans on any input of length $n$.

Complexity Classes

**Definition:** $\text{TIME}(t(n)) = \{ L : \text{there exists a TM } M \text{ that decides } L \text{ in time } O(t(n)) \}$

$P = \bigcup_{k \geq 1} \text{TIME}(n^k)$

$\text{EXP} = \bigcup_{k \geq 1} \text{TIME}(2^{n^k})$

**Definition:** $\text{SPACE}(t(n)) = \{ L : \text{there exists a TM } M \text{ that decides } L \text{ in space } O(t(n)) \}$

$\text{PSPACE} = \bigcup_{k \geq 1} \text{SPACE}(2^{n^k})$
Complexity Classes

**Definition:** \( \text{NTIME}(t(n)) = \{ L : \text{there exists a NTM M that decides L in time } O(t(n)) \} \)

\( \text{NP} = \bigcup_{k \geq 1} \text{NTIME}(2^{n^k}) \)

- Theorem: \( \text{P} \subseteq \text{EXP} \)
- \( \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \subseteq \text{EXP} \)
- Don’t know if any of the containments are proper.

Alternate definition of NP

**Theorem:** language \( L \) is in NP if and only if it is expressible as:

\( L = \{ x | \exists y, |y| \leq |x|, (x, y) \in R \} \)

where \( R \) is a language in \( \text{P} \).

Poly-time reductions

- Type of reduction we will use:
  - “many-one” poly-time reduction (commonly)
  - “mapping” poly-time reduction (book)

1. \( f \) poly-time computable
2. YES maps to YES
3. NO maps to NO

Hardness and completeness

**Definition:** a language \( L \) is \( C \)-hard if for every language \( A \in C \), \( A \) poly-time reduces to \( L \), i.e., \( A \leq_p L \).

- can show \( L \) is \( C \)-hard by reducing from a known \( C \)-hard problem

**Definition:** a language \( L \) is \( C \)-complete if \( L \) is \( C \)-hard and \( L \in C \)

Complete problems

- \( \text{EXP} \)-complete: \( \text{ATM}_m = \{ <M, x, m> : M \text{ is a TM that accepts } x \text{ within at most } m \text{ steps} \} \)
- \( \text{PSPACE} \)-complete: \( \text{QSAT} = \{ \phi : \phi \text{ is a 3-CNF, and } \exists x_1, x_2, x_3, \ldots x_k. \phi(x_1, x_2, \ldots x_k) \} \)
- \( \text{NP} \)-complete: \( \text{3SAT} = \{ \phi : \phi \text{ is a satisfiable 3-CNF formula} \} \)

Lots of NP-complete problems

- Indenent Set
- Vertex Cover
- Clique
- Hamilton Path (directed and undirected)
- Hamilton Cycle and TSP
- Subset Sum
- \( \text{NAE3SAT} \)
- Max Cut
- Problem sets: max/min Bisection, 3-coloring, subgraph isomorphism, subset sum, (3,3)-SAT, Partition, Knapsack, Max2SAT…
Other complexity classes

- coNP – complement of NP
  - complete problems: UNSAT, DNF-TAUTOLEGY

- NP intersect coNP
  - contains (decision version of) FACTORING

- PSPACE
  - complete problems: QSAT, GEOGRAPHY

Complexity classes

all containments believed to be proper