1

Outline

- Challenges to Extended Church-Turing
  - randomized computation
  - quantum computation
- Course review

2

Extended Church-Turing Thesis

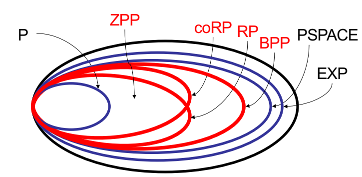
- the belief that TMs formalize our intuitive notion of an efficient algorithm is:

> The "extended" Church-Turing Thesis
>
> everything we can compute in time t(n) on a physical computer can be computed on a Turing Machine in time $t(n)^{O(1)}$ (polynomial slowdown)

- randomized computation challenges this belief

3

RP,coRP, BPP

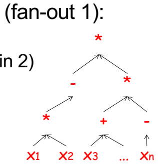


- from definitions: ZPP ⊆ RP, coRP ⊆ BPP

4

Polynomial identity testing

- Given: polynomial $p(x_1, x_2, \ldots, x_n)$ as arithmetic formula (fan-out 1):
  - multiplication (fan-in 2)
  - addition (fan-in 2)
  - negation (fan-in 1)



variables take values in finite field F

5

Polynomial identity testing

- Question: Is p identically zero?
  - i.e., is $p(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbf{F}^n$
  - (assume |**F**| larger than degree…)
- "polynomial identity testing" because given two polynomials p, q, we can check the identity $p \equiv q$ by checking if $(p - q) \equiv 0$

6

Polynomial identity testing

Lemma (Schwartz-Zippel): Let

$p(x_1, x_2, \ldots, x_n)$

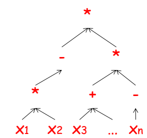
be a total degree d polynomial over a field **F** and let S be any subset of **F**. Then if p is not identically 0,

$$\Pr_{r_1, r_2, \ldots, r_n \in S}[\, p(r_1, r_2, \ldots, r_n) = 0] \le d/|S|.$$

7

Polynomial identity testing

- Given: polynomial $p(x_1, x_2, \ldots, x_n)$ over field **F**
- Is p identically zero?



- Note: degree d is at most the size of input

8

Polynomial identity testing

- randomized algorithm: pick a subset $S \subseteq$ **F** of size 2d
  - pick $r_1, r_2, \ldots, r_n$ from S uniformly at random
  - if $p(r_1, r_2, \ldots, r_n) = 0$, answer "yes"
  - if $p(r_1, r_2, \ldots, r_n) \ne 0$, answer "no"
- if p identically zero, never wrong
- if not, Schwartz-Zippel ensures probability of error at most ½

9

Randomized complexity classes

- We have shown:
  - Polynomial Identity Testing is in coRP
  - note: no sub-exponential time deterministic algorithm know

10

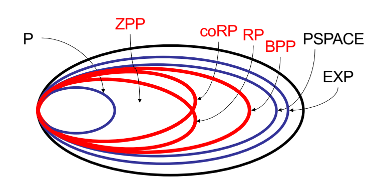
Randomized complexity classes

- How powerful is randomized computation?
- We have seen an example of a problem in **BPP** that we only know how to solve deterministically in **EXP**.

Is randomness a panacea for intractability?

11

Randomized complexity classes



- believed that P = ZPP = RP = coRP = BPP (!)

12

Course Review

Review

- Highest level: 2 main points

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

Review

- Highest level: 2 main points

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)

Review

- 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

Review

- 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.

Review

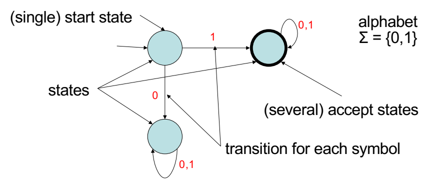
- Important ideas (continued):

A complete problem is a surrogate for the entire class.

Summary

Part I: automata

Finite Automata



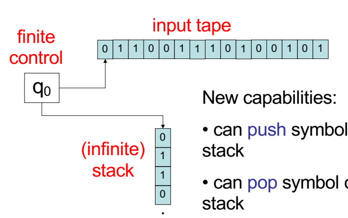
- read input one symbol at a time; follow arrows; accept if end in accept state

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").

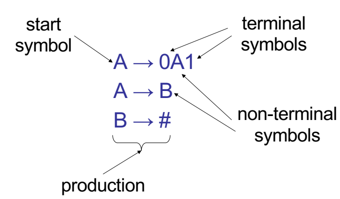
Pushdown Automata



New capabilities:

- can push symbol onto stack
- can pop symbol off of stack

Context-Free Grammars



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 $i \geq 0$, $xy^iz \in L$, 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

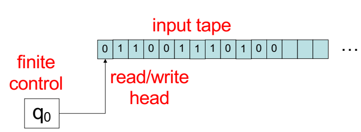
$w = uvxyz$ such that

1. for every $i \geq 0$, $uv^ixy^iz \in L$, and
2. $|vy| > 0$, and
3. $|vxy| \leq p$.

Summary

Part II: Turing Machines and decidability

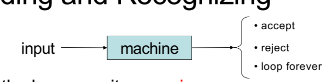
Turing Machines



- 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:

<u>The Church-Turing Thesis</u>

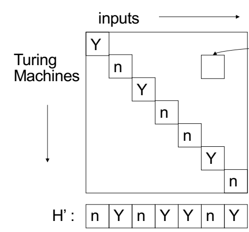
everything we can compute on a physical computer

can be computed on a Turing Machine

- Note: this is a belief, not a theorem.

31

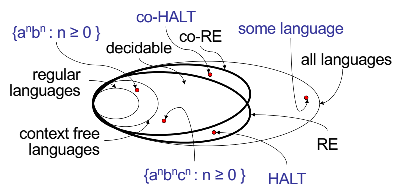
The Halting Problem



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.

32

Decidable, RE, coRE…

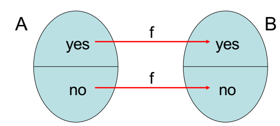


some problems (e.g HALT) have no algorithms

33

Definition of reduction

- More refined notion of reduction:
  - "many-one" reduction (commonly)
  - "mapping" reduction (book)



reduction from language A to language B

34

Using reductions

- Used reductions to prove lots of problems were:
  - undecidable (reduce from undecidable)
  - non-RE (reduce from non-RE)
    - or show undecidable, and coRE
  - non-coRE (reduce from non-coRE)
    - or show undecidable, and RE

<u>**Rice's Theorem**</u>: Every nontrivial TM property is undecidable.

35

The Recursion Theorem

<u>**Theorem**</u>: Let T be a TM that computes fn:

$t: \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$

There is a TM R that computes the fn:

$r: \Sigma^* \rightarrow \Sigma^*$

defined as $r(w) = t(w, \langle R \rangle)$.

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

36

Summary

Part III: Complexity

Complexity

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

(number of random bits, degree of parallelism, rounds of interaction, others: not in this course)

Time and Space Complexity

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

Complexity Classes

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

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

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

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

$$PSPACE = \cup_{k \geq 1} SPACE(n^k)$$

Complexity Classes

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

$$NP = \cup_{k \geq 1} NTIME(n^k)$$

- Theorem: $P \subsetneq EXP$
- $P \subseteq NP \subseteq PSPACE \subseteq EXP$
- Don't know if any of the containments are proper.

Alternate definition of NP

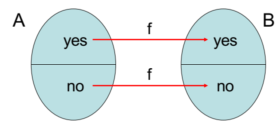
Theorem: language L is in NP if and only if it is expressible as:

$$L = \{ x \mid \exists y, |y| \leq |x|^k, (x, y) \in R \}$$

where R is a language in 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

43

Hardness and completeness

Definition: a language L is C-hard if for every language A ∈ 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 ∈ C

44

Complete problems

- EXP-complete: ATM_B = {<M, x, m> : M is a TM that accepts x within at most m steps}
- PSPACE-complete: QSAT = {φ : φ is a 3-CNF, and $\exists x_1 \forall x_2 \exists x_3 \ldots \forall x_n\ \varphi(x_1, x_2, \ldots x_n)$ }
- NP-complete: 3SAT = {φ : φ is a satisfiable 3-CNF formula}

45

Lots of NP-complete problems

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

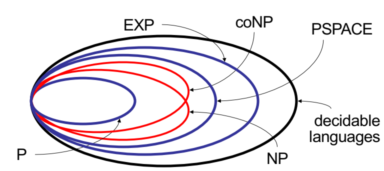
46

Other complexity classes

- coNP – complement of NP
  - complete problems: UNSAT, DNF-TAUTOLOGY
- NP intersect coNP
  - contains (decision version of) FACTORING
- PSPACE
  - complete problems: QSAT, GEOGRAPHY

47

Complexity classes



all containments believed to be proper

48

Quantum Computation

March 6, 2024 CS21 Lecture 26 49

49

Extended Church-Turing Thesis

- the belief that TMs formalize our intuitive notion of an efficient algorithm is:

The "extended" Church-Turing Thesis

everything we can compute in time t(n) on a physical computer can be computed on a (probabilistic) Turing Machine in time $t(n)^{O(1)}$ (polynomial slowdown)

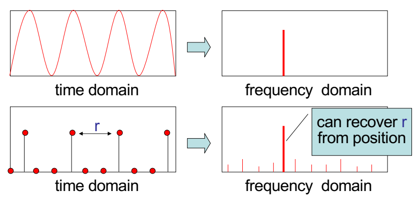
- Quantum computation challenges this belief

March 6, 2024 CS21 Lecture 26 50

50

For use later…

- Fourier transform:



March 6, 2024 CS21 Lecture 26 51

51

A different model

- infinite tape of a Turing Machine is an idealized model of computer
- real computer is a Finite Automaton (!)
  - n bits of memory
  - 2^n states

March 6, 2024 CS21 Lecture 26 52

52

Model of deterministic computation

$$\begin{pmatrix}1\\0\\0\\0\\\vdots\\0\end{pmatrix}\begin{pmatrix}0\\1\\0\\0\\\vdots\\0\end{pmatrix}\begin{pmatrix}0\\0\\1\\0\\\vdots\\0\end{pmatrix}\cdots\begin{pmatrix}0\\0\\0\\0\\\vdots\\1\end{pmatrix}$$

2^n possible basic states

state at time t

state at time t+1

one 1 per column

$$\begin{pmatrix}0&0&0&0\\1&0&0&0\\0&0&1&0\\0&1&0&1\end{pmatrix}\begin{pmatrix}0\\1\\0\\0\end{pmatrix}=\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$$

March 6, 2024 CS21 Lecture 26 53

53

Model of randomized computation

$$\begin{pmatrix}p_0\\p_1\\p_2\\p_3\\\vdots\\p_{2^n-1}\end{pmatrix}$$

possible states at time t: $\sum_i p_i = 1$ $p_i \in R^+$

state at time t

state at time t+1

"stochastic matrix" sum in each column = 1

$$\begin{pmatrix}0&\frac{1}{4}&0&\frac{1}{2}\\\frac{1}{2}&\frac{1}{4}&0&\frac{1}{4}\\\frac{1}{2}&\frac{1}{4}&1&\frac{1}{4}\\0&\frac{1}{4}&0&0\end{pmatrix}\begin{pmatrix}0\\\frac{1}{2}\\0\\\frac{1}{2}\end{pmatrix}=\begin{pmatrix}\frac{3}{8}\\\frac{1}{4}\\\frac{1}{4}\\\frac{1}{8}\end{pmatrix}$$

March 6, 2024 CS21 Lecture 26 54

54

Model of randomized computation

- at end of computation, see specific state
- demand correct result with high probability
- think of as "measuring" system:

$$\begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \\ \vdots \\ p_{2^n-1} \end{pmatrix} \Rightarrow \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

see i^{th} basic state with probability p_i