CS21
Decidability and Tractability

Lecture 25

March 7, 2018

Outline

- challenges to the extended Church-Turing Thesis
  - randomized computation
  - quantum computation

Extended Church-Turing Thesis

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

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

Randomness in computation

- Example of the power of randomness
- Randomized complexity classes

Communication complexity

two parties: Alice and Bob

function $f:\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$

Alice holds $x \in \{0,1\}^n$; Bob holds $y \in \{0,1\}^n$

- Goal: compute $f(x, y)$ while communicating as few bits as possible between Alice and Bob

- count number of bits exchanged (computation free)
- at each step: one party sends bits that are a function of held input and received bits so far

Communication complexity

- simple function (equality):

$$EQ(x, y) = 1 \text{ iff } x = y$$

- simple protocol:
  - Alice sends x to Bob (n bits)
  - Bob sends EQ(x, y) to Alice (1 bit)
  - total: n + 1 bits
  - (works for any predicate f)

Communication complexity

- protocol for EQ employing randomness?
  - Alice picks random prime p in $\{1\ldots 4n^2\}$, sends:
    - p
    - (x mod p)
  - Bob sends:
    - (y mod p)
  - players output 1 if and only if:

$$(x \bmod p) = (y \bmod p)$$

Communication complexity

– O(log n) bits exchanged

– if x = y, always correct

– if x ≠ y, incorrect if and only if:

p divides |x – y|

– # primes in range is ≥ 2n

– # primes dividing |x – y| is ≤ n

– probability incorrect ≤ 1/2

Randomness gives an exponential advantage!!

Communication complexity

two parties: Alice and Bob

function $f:\{0,1\}^n \times \{0,1\}^n \rightarrow \{0,1\}$

Alice holds $x \in \{0,1\}^n$; Bob holds $y \in \{0,1\}^n$

- Goal: compute $f(x, y)$ while communicating as few bits as possible between Alice and Bob

Example: $EQ(x, y) = 1$ iff $x = y$

- Deterministic protocol: no fewer than $n+1$ bits
- Randomized protocol: $O(\log n)$ bits

Extended Church-Turing Thesis

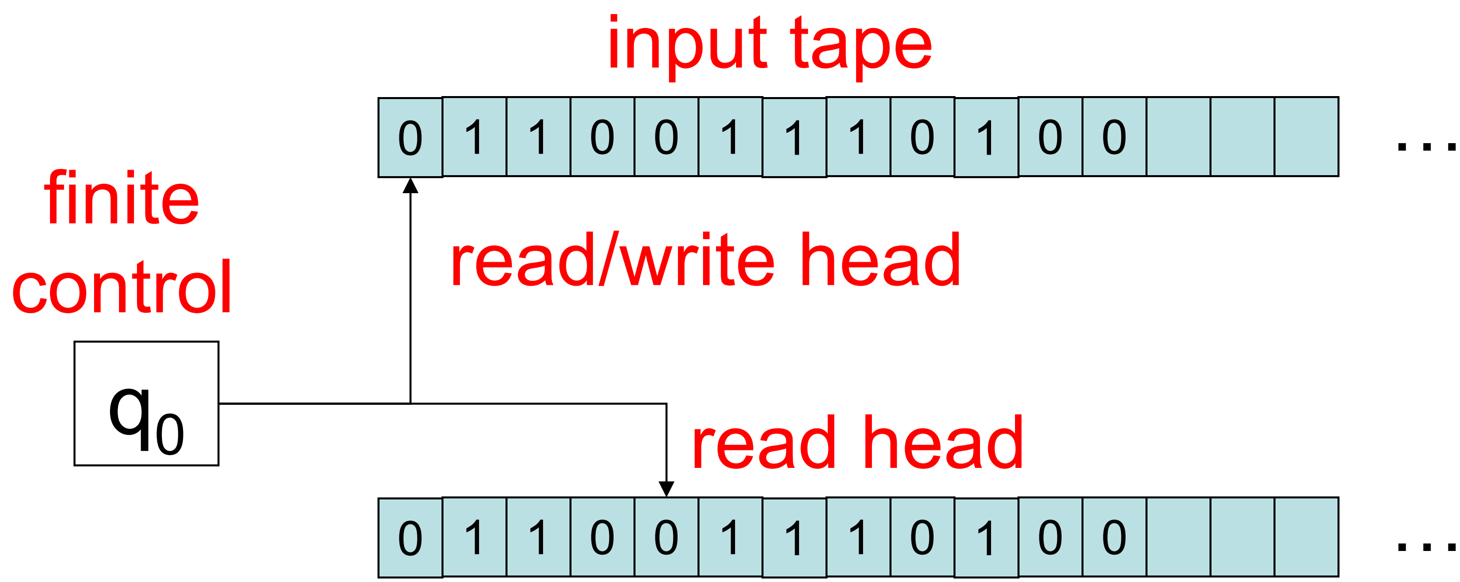
- Common to insert "probabilistic":

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)

Randomized complexity classes

- model: probabilistic Turing Machine
  - deterministic TM with additional read-only tape containing “coin flips”



Randomized complexity classes

- **RP** (Random Polynomial-time)
  - $L \in$ **RP** if there is a p.p.t. TM M:

$$x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq ½$$

$$x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] = 1$$

- **coRP** (complement of Random Polynomial-time)
  - $L \in$ **coRP** if there is a p.p.t. TM M:

$$x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] = 1$$

$$x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] \geq ½$$

"p.p.t" = probabilistic polynomial time

Randomized complexity classes

- **BPP** (Bounded-error Probabilistic Poly-time)
  - L $\in$ **BPP** if there is a p.p.t. TM M:

$$x \in L \Rightarrow \Pr_y[M(x,y) \text{ accepts}] \geq 2/3$$

$$x \notin L \Rightarrow \Pr_y[M(x,y) \text{ rejects}] \geq 2/3$$

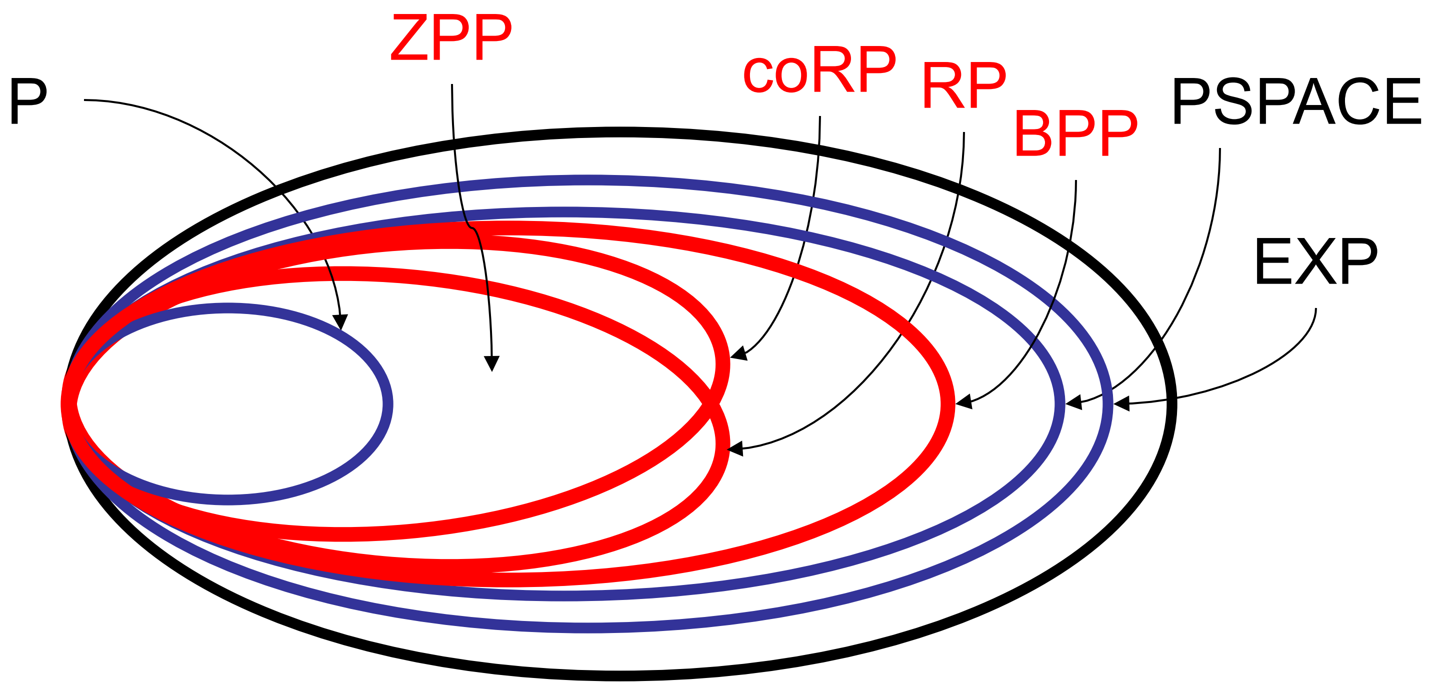
Randomized complexity classes

These classes may capture "efficiently computable" better than **P**.

One more important class:

- **ZPP** (Zero-error Probabilistic Poly-time)
  - **ZPP = RP ∩ coRP**
  - $\Pr_y$[M(x,y) outputs "fail"] ≤ ½
  - otherwise outputs correct answer

RP,coRP, BPP



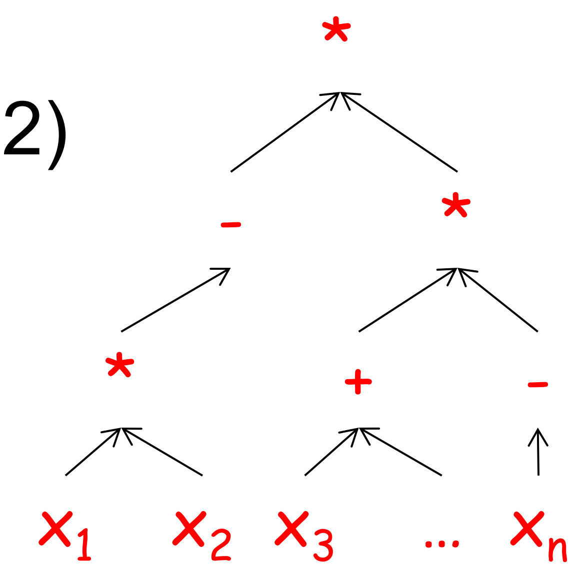
- from definitions: ZPP ⊂ RP, coRP ⊂ BPP

Relationship to other classes

- all these classes contain **P**
  - they can simply ignore the tape with coin flips
- all are in **PSPACE**
  - can exhaustively try all strings y
  - count accepts/rejects; compute probability
- **RP** $\subset$ **NP** (and **coRP** $\subset$ **coNP**)
  - multitude of accepting computations
  - **NP** requires only one

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)



Polynomial identity testing

- Question: Is p identically zero?
  - i.e., is $p(\mathbf{x}) = 0$ for all $\mathbf{x} \in \mathbf{F}^n$
  - (assume $|\mathbf{F}|$ larger than degree…)

- "polynomial identity testing" because given two polynomials p, q, we can check the identity $p \equiv q$ by checking if $(p - q) \equiv 0$

Polynomial identity testing

- try all $|\mathbf{F}|^n$ inputs?
  - may be exponentially many
- multiply out symbolically, check that all coefficients are zero?
  - may be exponentially many coefficients
- Best known deterministic algorithm places in EXP

Polynomial identity testing

Lemma (Schwartz-Zippel): Let

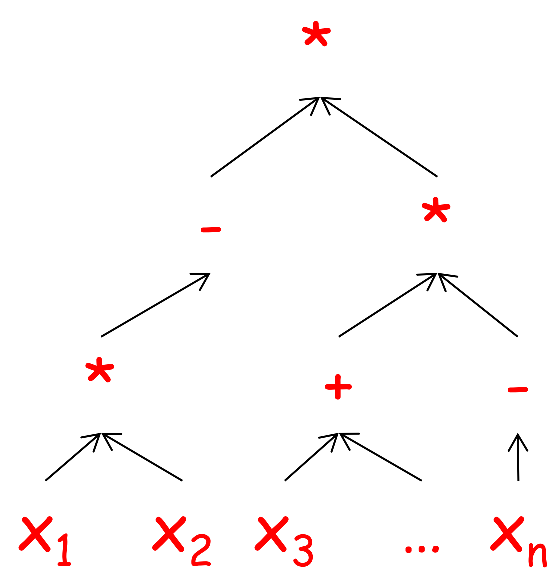
$$p(x_1, x_2, \ldots, x_n)$$

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,

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

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

Polynomial identity testing

- randomized algorithm: pick a subset $S \subset \mathbf{F}$ 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 $p(r_1, r_2, \ldots, r_n) \neq 0$, answer "no"

- if p identically zero, never wrong
- if not, Schwartz-Zippel ensures probability of error at most ½

Randomized complexity classes

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

Randomized complexity classes

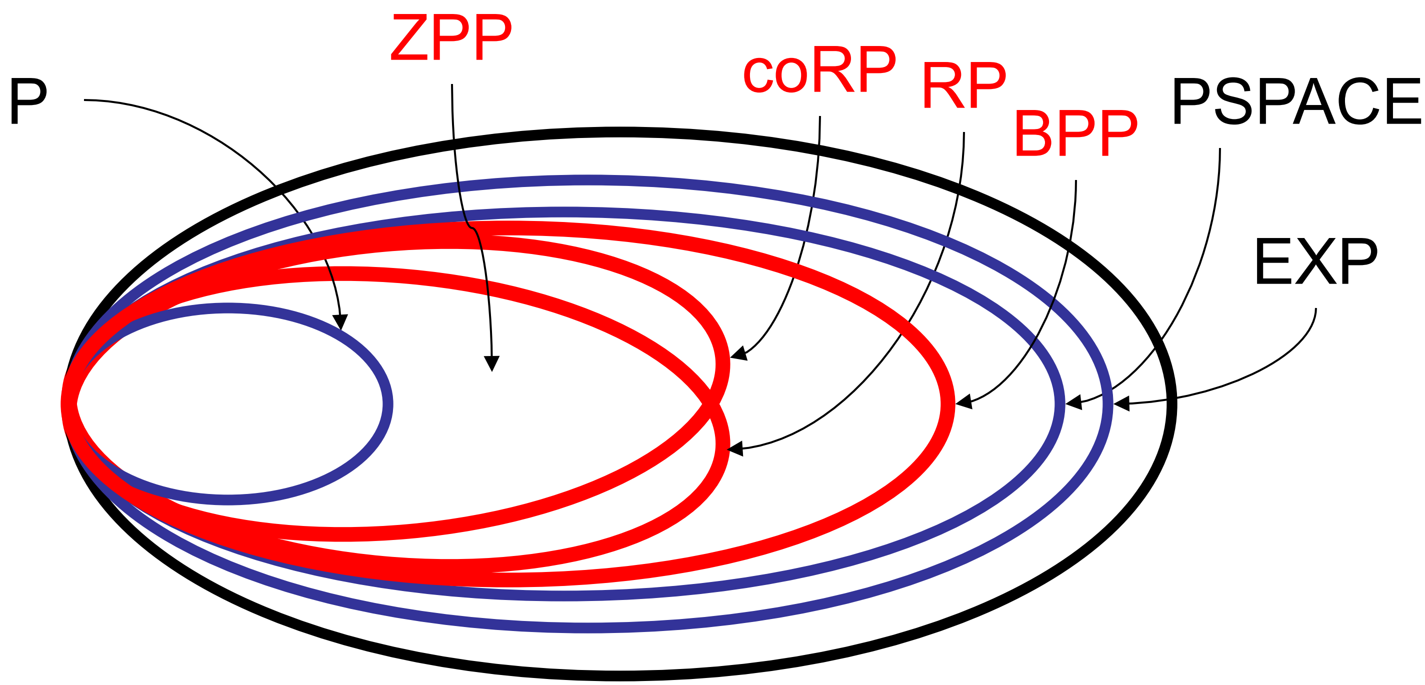
- How powerful is randomized computation?
- We have seen an example of a problem in

  BPP

  that we only know how to solve deterministically in **EXP**.

Is randomness a panacea for intractability?

Randomized complexity classes



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