



Review of Complexity Theory

Breno de Medeiros

Department of Computer Science

Florida State University

Turing Machines

- ⑥ A traditional way to model a computer mathematically is the *Turing Machine* model (TM). A Turing Machine is characterized by that:
 - △ *TM* recognizes a set of symbols from a finite alphabet Σ . Assume that $\Sigma = \{0, 1, \sqcup\}$. By a *string* understand any finite sequence of 0's and 1's followed by a \sqcup .
 - △ *TM* has an half-infinite tape containing cells. Each cell may store an element of Σ .
 - △ *TM* has a set of states S with a special state called start.
 - △ At the beginning of a computation, *TM* is at state start (an element of S), and the tape contains a string.
 - △ There is a special state Halt. (Technically, NOT an element of S .)
 - △ If the *TM* solves a *decisional problem*, it has also special states YES and NO (not part of S either).
 - △ If the *TM* solves a computational problem, it has an (infinite) output tape, where it may write a string.

Turing Machine Computations

TM has a tape-head. At start, it lies in the first tape cell. TM 's computations are described by a state-transition function δ .

- ⑥ δ is a function from $\Sigma \times S$ to $\Sigma \times S \cup \{ \text{halt, YES, NO} \} \times \{ \rightarrow, \leftarrow, \downarrow \}$.
- ⑥ At each step, the TM reads the symbol σ_{curr} in the tape cell at the current tape-head position, and its current state s_{curr} . It then computes $\delta(\sigma_{curr}, s_{curr}) = (\sigma_{next}, s_{next}, pos_{next})$. Then:
 - △ σ_{next} is written to the current tape cell, and the tape head is moved one cell right, left, or stays in the same position according to pos_{next} being equal to \rightarrow , \leftarrow , or \downarrow .
 - △ The current state is changed to s_{next} . If s_{next} is one of halt, YES, NO, then the computations is concluded.
- ⑥ If TM has an output tape, δ also specifies an optional symbol to be written to output, and the movement of the output tape-head.

Example of a Turing Machine Computation



Terminology

- ⑥ Each step of a Turing Machine is given a unitary time cost.
- ⑥ Let $t_{TM}(\delta)$ be the number of steps taken by a Turing Machine TM to complete processing an input string δ .
- ⑥ $|\delta|$, the length of δ , is the number of 0's and 1's in the representation of δ .

Language Accepted by a Turing Machine

- ⑥ A language L is a subset of $\{0, 1\}^*$. (Considered strings by implicitly appending a terminating \sqcup .)
 - △ The language $L(TM)$ accepted by a turing machine TM is the set of strings that lead to TM terminate in finitely many steps, in the state YES.
- ⑥ The class P (polynomial-time) contains all languages L such that:
 - △ Given L , there is a Turing Machine TM_L such that L is the language accepted by TM_L , and moreover a polynomial $p_{TM_L}(n)$ such that, if ℓ is in L and $|\ell| \leq n$, then $t_{TM_L}(\ell) \leq p_{TM_L}(n)$.

Example of a Polynomial Language

[Redacted content]

Non-deterministic Turing Machines

- ⑥ A non-realistic computational model that is nonetheless important from a theoretical standpoint is that represented by a non-deterministic Turing Machine (*NTM*).
- ⑥ An *NTM* is similar to a (deterministic) *TM* except that instead of a single-valued transition function δ , it has a multi-valued one $\tilde{\delta}$. At any given step in the computation, the machine may choose which of the possible values to take for the transition.
- ⑥ A string ℓ is accepted by an *NTM* if there is a valid *computational path* (i.e., set of choices at each step compatible with $\tilde{\delta}$) which lead to *NTM* terminate in state YES, when given ℓ as its input at state start, and if no valid computational paths lead to *NTM* terminating in state NO, when given ℓ as input.

Non-deterministic time

- ⑥ If ℓ is accepted by NTM , the time $\tilde{t}_{NTM}(\ell)$ to process string ℓ is, by definition, the shortest valid computational path leading to NTM accepting ℓ .
- ⑥ The complexity class NP is defined as the set of languages that are accepted by a non-deterministic machine in polynomial time. More precisely:
 - △ Given L , there is a non-deterministic Turing machine NTM_L such that L is the language accepted by NTM_L , and a polynomial p_{NTM_L} , such that if ℓ is in L , and $|\ell| \leq n$, then:
 - △ $\tilde{t}_{NTM_L}(\ell) \leq p_{NTM_L}(\ell)$.

Example of an NP language



Introducing randomization

- ⑥ We have considered deterministic computations, as well as (non-realistic) non-deterministic ones. However, in practice one often considers *probabilistic* computations.
- ⑥ These computations are like non-deterministic ones in the sense that their transition function is many-valued, but each type of transition is associated with a probability value.
- ⑥ Given a probabilistic machine PTM , each input string ℓ may lead to both accepting and rejecting computational paths.
- ⑥ The probability of acceptance is the total sum of the probabilities associated to all accepting computational paths.
- ⑥ Randomized complexity classes are defined in terms of the existence of PTM 's that accept their languages with specified acceptance and rejection probabilities.

The class RP

⑥ RP is the class of languages L such that:

- △ There is a probabilistic turing machine PTM_L and a polynomial p_{PTM_L} .
- △ For every string ℓ with $|\ell| \leq n$, all computational paths of PTM_L terminate in at most $p_{PTM_L}(n)$ steps.
- △ If ℓ does not belong to L , then PTM_L always rejects ℓ .
- △ If ℓ belongs to L , then the probability that PTM_L accepts is at least $1/2$.

⑥ Observations:

- △ RP contains P , as any (deterministic) Turing Machine TM is also a probabilistic (non-deterministic) TM satisfying the above probability restrictions.
- △ NP contains RP , as eliminating the probabilities from PTM gives a non-deterministic machine NTM .

RP language example



The classes co-RP and ZPP

- ⑥ The definition of RP is not symmetric, as the behavior when ℓ is in L or not differ.
- ⑥ If we invert the definition, we have the co-RP class, containing the languages L such that there is a Turing machine that always accepts if ℓ is in L but rejects with probability at least $1/2$ if ℓ is not in L .
- ⑥ As before, $P \subset \text{co-RP}$.
- ⑥ The class ZPP is defined as $RP \cap \text{co-RP}$.
- ⑥ $P \subset ZPP$, and many believe that $P = ZPP$, but that has not been proven.

Another characterization of ZPP

- ⑥ ZPP can also be characterized by the class of languages that have PTM such that with probability 1 the algorithm will terminate with the correct answer. Note that here the answer is guaranteed to be correct always, but the running time is not guaranteed to be polynomial, only expected to, in a probabilistic sense. Show the equivalence.

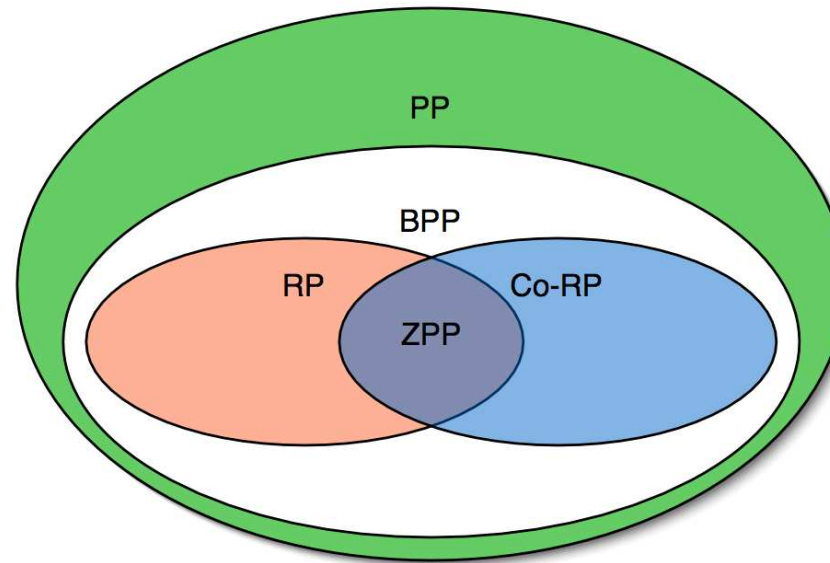
Bounded-error, Probabilistic Polynomial (BPP) Class

- ⑥ BPP is the class on languages decidable by PTM that finish all instances in polynomial time, and have an error at most $1/3$ in all instances. Formally:
 - △ A language L is in BPP , if there are a probabilistic Turing machine PTM_L and polynomial p_{PTM_L} s.t. PTM_L takes at most $p_{PTM_L}(n)$ steps to process any strings of length n , and:
 - △ If $\ell \in L$, then PTM_L accepts with probability at least $2/3$.
 - △ If $\ell \notin L$, it accepts with probability at most $1/3$.
- ⑥ The bound $1/3$ is arbitrary. By simulating the PTM on the same input a polynomial number of times and using majority voting, one can show that is sufficient to start with error probability $1/2 - 1/p_1(n)$, where $p_1(\cdot)$ is any polynomial and then the error can be reduced to ϵ , such that $-\log_2 \epsilon < p_2(n)$, for any polynomial $p_2(\cdot)$. In particular, any constant $C \in (0, 1/2)$ can be used instead of $1/3$.

The Complexity Class PP

- ⑥ PP is the class of all decision problems solvable in (deterministic) polynomial time by a PTM , with the following error probabilities:
 - △ If ℓ belongs to the language, (is a YES instance), then the PTM accepts with a probability strictly larger than $1/2$.
 - △ On a NO instance, the algorithm replies YES with probability at most equal to $1/2$.
- ⑥ For instance, SAT is a problem in PP . To see this, consider a Boolean formula ϕ in n variables, and the following PTM :
 - △ Choose a random assignment $\{\text{true}, \text{false}\} \leftarrow x_i$ to each of the boolean variables, independently.
 - △ Evaluate ϕ on the above assignment. If true, answer YES. If false, answer YES or NO with equal probability $1/2$.

Summary



Relationships between the randomized complexity classes.

The class PP is not efficiently decidable because the probabilities cannot be amplified by repeated application of the protocol. BPP and the other classes are efficiently recognizable via probabilistic algorithms.