2nd Week

NP-Completeness, Probabilistic and Counting Complexity Classes

Synopsis.

- P, NP, co-NP
- NP-Complete Problems
- Kolmogorov Complexity
- Probabilistic Computation
- Counting Complexity Classes

April 16, 2018. 23:59

Course Schedule: 16 Weeks

Subject to Change

- Week 1: Basic Computation Models
- Week 2: NP-Completeness, Probabilistic and Counting Complexity Classes
- Week 3: Space Complexity and the Linear Space Hypothesis
- Week 4: Relativizations and Hierarchies
- Week 5: Structural Properties by Finite Automata
- Week 6: Stype-2 Computability, Multi-Valued Functions, and State Complexity
- Week 7: Cryptographic Concepts for Finite Automata
- Week 8: Constraint Satisfaction Problems
- Week 9: Combinatorial Optimization Problems
- Week 10: Average-Case Complexity
- Week 11: Basics of Quantum Information
- Week 12: BQP, NQP, Quantum NP, and Quantum Finite Automata
- Week 13: Quantum State Complexity and Advice
- Week 14: Quantum Cryptographic Systems
- Week 15: Quantum Interactive Proofs
- Week 16: Final Evaluation Day (no lecture)

YouTube Videos

- This lecture series is based on numerous papers of T. Yamakami. He gave conference talks (in English) and invited talks (in English), some of which were video-recorded and uploaded to YouTube.
- Use the following keywords to find a playlist of those videos.
- YouTube search keywords:

  Tomoyuki Yamakami conference invited talk playlist

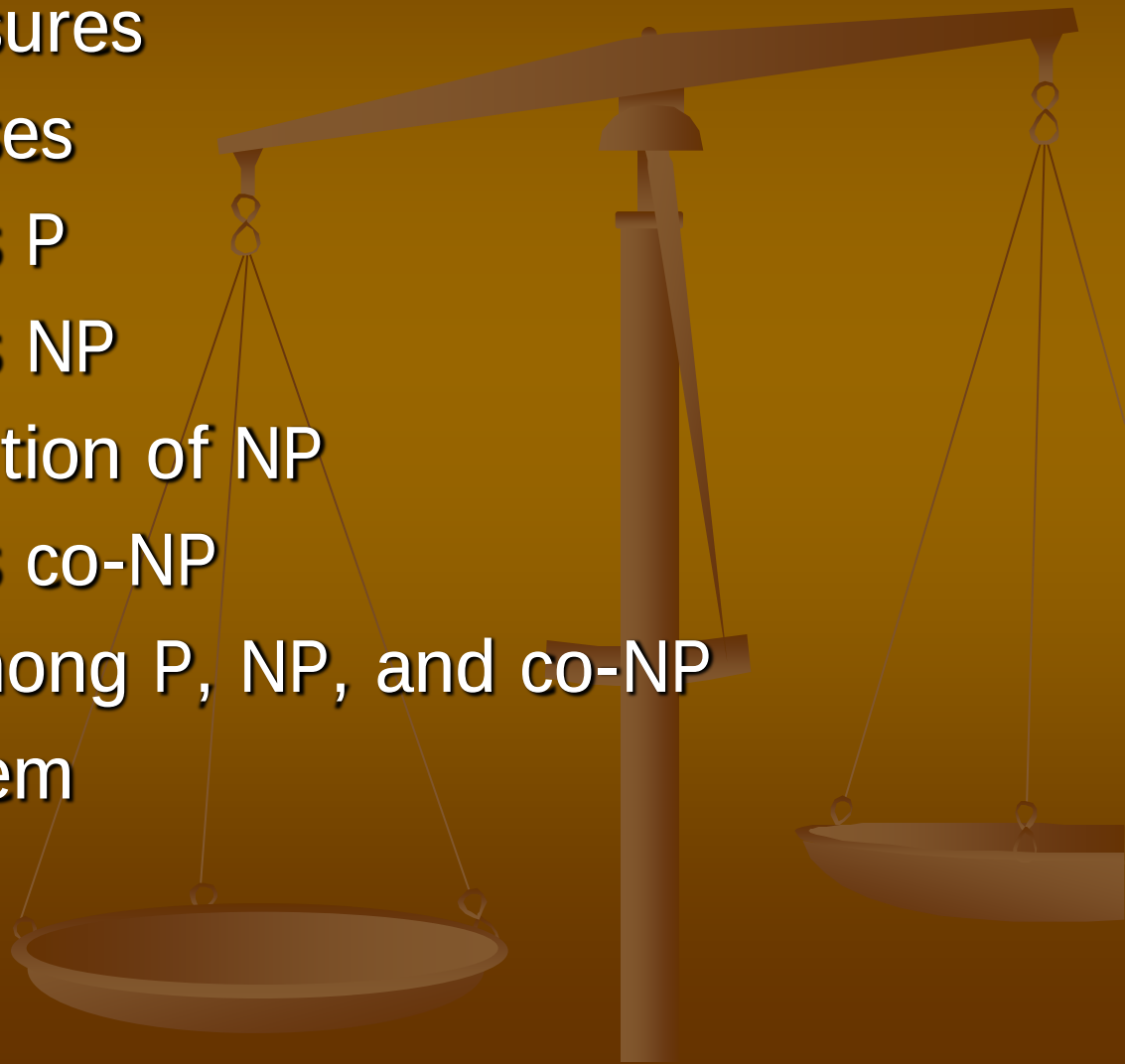
Conference talk video

Main References by T. Yamakami

- T. Yamakami. Oracle pushdown automata, nondeterministic reducibilities, and the hierarchy over the family of context-free languages. In Proc. of SOFSEM 2014, Lecture Notes in Computer Science, vol. 8327, pp. 514-525 (2014). A complete version is available at arXiv:1303.1717.
- T. Yamakami. One-way bounded-error probabilistic pushdown automata and Kolmogorov complexity (preliminary report). In Proc. of DLT 2017, Lecture Notes in Computer Science, vol. 10396, pp. 353-364 (2017). A complete and corrected version will be posted at arXiv.org shortly.

I. Basic Complexity Classes

1. Complexity Measures
2. Complexity Classes
3. Complexity Class P
4. Complexity Class NP
5. Another Formulation of NP
6. Complexity Class co-NP
7. Relationships among P, NP, and co-NP
8. The P=NP Problem



Complexity Measures

- A notion of complexity measure is used to classify various "problems" (i.e., languages and functions).
- Basic complexity measures of algorithms include the running time and the usage of memory space.
- We say that problem A is of time complexity t(n) if there exists an algorithm that solves A in time t(n) for all length-n inputs.
- Similarly, problem A is of space complexity s(n) if there exists an algorithm that solves A in space s(n) for all length-n inputs.
- (*) Other complexity measures, including circuit complexity and state complexity, will be discussed in Weeks 3 and 6.

Complexity Classes

- Assume that a specific complexity measure is given.
- Informally, we define a complexity class as a collection of decision problems, solutions of which is measured by a complexity measure of an algorithm.
- Namely, a complexity class is a set of problems, which can be solved by algorithms of the given complexity measure.
- In particular, a complexity class of decision problems is also called a family of languages because decision problems are identified with languages (see Week 1).

Complexity Class P

- The complexity class P is the set of decision problems (or languages) that are polynomial-time solvable.
- More precisely, a decision problem (or a language) L is in P if there exist a constant $k \geq 1$ and a multi-tape DTM (deterministic Turing machine) M s.t., for any input x,
  1. $x \in L \rightarrow$ M accepts x in $O(n^k)$ time, and
  2. $x \notin L \rightarrow$ M rejects x in $O(n^k)$ time.
- Many natural problems belong to this complexity class P.
- (Example) The problem PRIMES of determining whether a given positive integer is a prime number belongs to P. [Agrawal-Kayal-Saxena (2002)].

Who Introduced Class P?

- The class P was introduced in 1964 by Alan Cobham, and independently, in 1965 by Jack Edmonds.
  - Alan Cobham. The intrinsic computational difficulty of functions. In Proceedings of the 1964 Congress for Logic, Methodology, and the Philosophy of Science, pp. 24-30, 1964.
  - Jack Edmonds. Paths, trees, and flowers. Canadian Journal of Mathematics, Vol. 17, pp. 449-467, 1965.

A. Cobham

J. Edmonds

Closure Properties of P

- The complexity class P is closed under Boolean operations, concatenation, and Kleene star.

- (Claim) If $L_1, L_2 \in P$, then

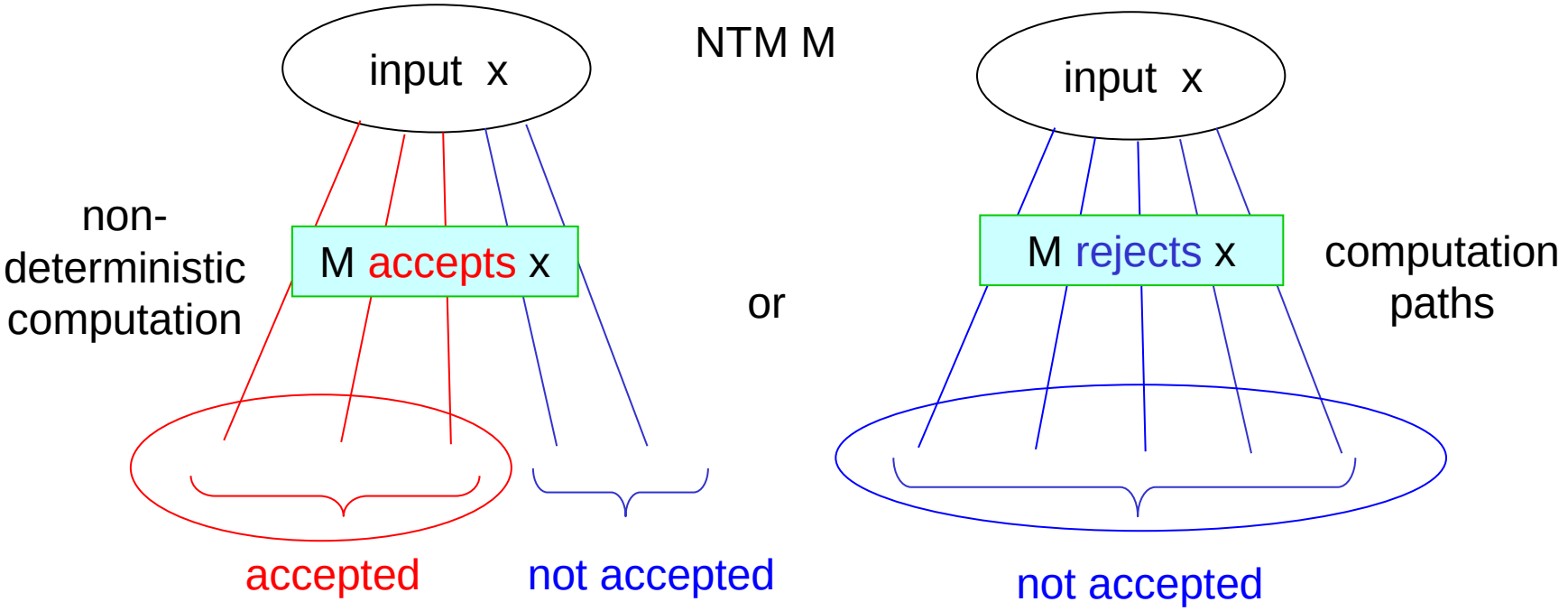
  $L_1 \cup L_2 \in P$, $L_1 \cap L_2 \in P$, $L_1^c \in P$, $L_1 L_2 \in P$, and $L_1^* \in P$.

  - $L_1 \cup L_2$: union
  - $L_1 \cap L_2$: intersection
  - L_1^c : complement
  - $L_1 L_2$: concatenation
  - L_1^* : Kleene star (or Kleene closure)

  } Boolean operations

Acceptance vs Rejection for NTMs (revisited)

- On input x, an NTM M is said to accept x if M enters an accepting state along a certain computation path.
- An input that is not accepted is said to be rejected.

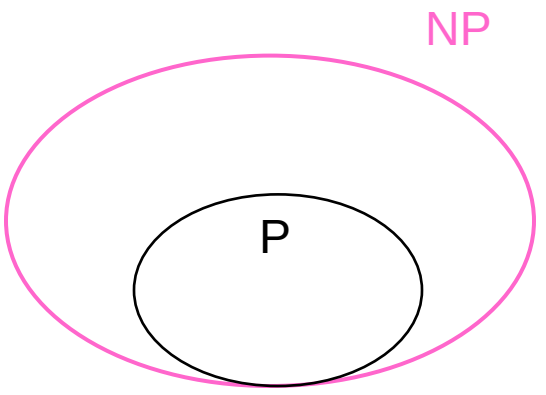


Complexity Class NP

- A decision problem (or a language) L is in NP if there is an NTM (nondeterministic Turing machine) M such that, for any input x,
  1. $x \in L \leftrightarrow$ there exists an accepting computation path of M on x (or x is accepted by M), and
  2. M halts in polynomial time.

- (Claim) $P \subseteq NP$

❑ Proof: This is because every deterministic computation is a special case of a nondeterministic computation.



Many believe in this way

Natural NP Problems I

This problem will be explained later.

- There are many natural decision problem the complexity class NP. For example:

- Boolean Formula Satisfiability Problem (SAT)
  - instance: a Boolean formula ϕ
  - question: Is there any satisfying assignment for ϕ?

- Traveling Salesperson Problem (TSP)
  - instance: a set of cities, a table of traveling cost between two cities, and a budget k
  - question: Is there any tour (i.e., visiting each city exactly once and finishing at the starting city) with cost at most k?

Natural NP Problems II

- Here are more examples of NP problems.

- 0-1 Knapsack Problem (KNAPSACK)
  - instance: a finite set U of items, size $s(u) \in N^+$, value $v(u) \in N^+$ for each $u \in U$, bounds $B \in N^+$, and $k \in N^+$
  - question: Is there a subset $A \subseteq U$ s.t. $\Sigma_{u \in A} s(u) \leq B$ and $\Sigma_{u \in A} v(u) \geq k$?
- Graph 3-Colorability Problem (3-COLOR)
  - instance: a graph G=(V,E)
  - question: Is G 3-colorable?

"3-colorable" means that there exist a function $f : V \rightarrow \{1,2,3\}$ s.t. $f(u) \neq f(v)$ whenever $\{u,v\} \in E$?

Another Formulation of NP

- Here is a quite different formulation of NP.
- A language L belongs to NP iff there exists a two-input polynomial-time algorithm A and constant $c \geq 1$ such that

$$L = \{ x \in \{0,1\}^* \mid \exists\, y \text{ s.t. } |y| = O(|x|^c) \text{ and } A(x,y) = 1 \}.$$

- In this case, "y" is called a certificate.
- Moreover, this algorithm A is said to verify the language L in polynomial time.

- In other words, the complexity class NP is the class of languages that can be verified by a polynomial-time algorithm.

Who Introduced Class NP?

- The class NP was introduced in 1965 by Jack Edmonds, who also conjectured that P≠NP.

  - Jack Edmonds. Paths, trees, and flowers. Canadian Journal of Mathematics, Vol.17, pp.449—467, 1965.

Complexity Class co-NP

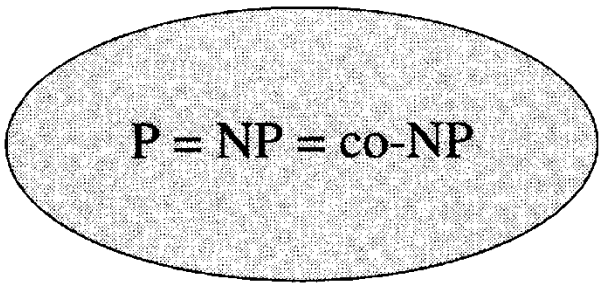
- For any language L, the complement L^c of L is the difference Σ^* - L.
- That is, L^c is the problem obtained from L by exchanging its outcomes of 0 and 1; namely,

$$L^c = \left\{ x \in \Sigma^* \mid x \notin L \right\}$$

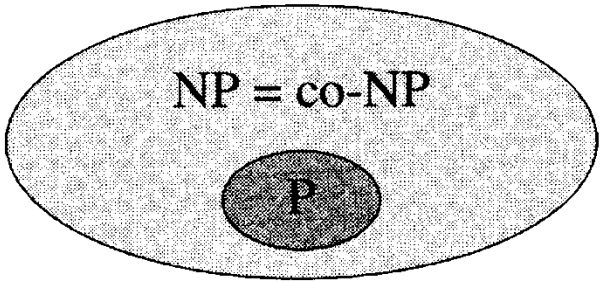
- (Claim) If $L \in P$ then $L^c \in P$. In other words, P = co-P.
- We define the complexity class co-NP as the set of decision problems (or languages) L such that $L^c \in NP$.
- In other words, $L \in NP \Leftrightarrow L^c \in$ co-NP.
- (Claim) CFL $\cup$ co-CFL $\subseteq$ P $\subseteq$ NP $\cap$ co-NP $\subseteq$ NP.

Relationships among P, NP, and co-NP

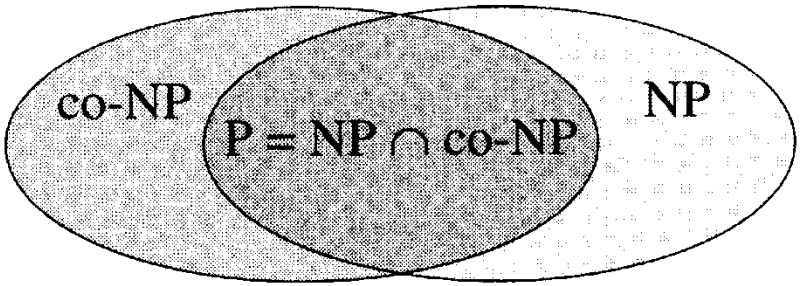
Four possible scenarios



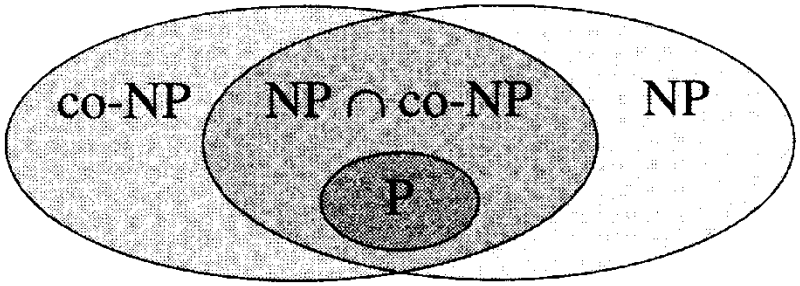
(a)



(b)



(c)



(d)

The P = NP Problem

- The P=NP Problem is one of the most famous open problems in our time.
- This problem asks if all NP problems are solvable in polynomial time. That is,

  Does $L \in NP$ imply $L \in P$?

- Clay Mathematics Institute would award anyone who solves the P=NP problem with $1,000,000 prize.

  (See the next slide.)

P vs NP Problem

Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving

- The Millennium Problems
- Official Problem Description — Stephen Cook
- Lecture by Vijaya Ramachandran at University of Texas (video)
- Minesweeper

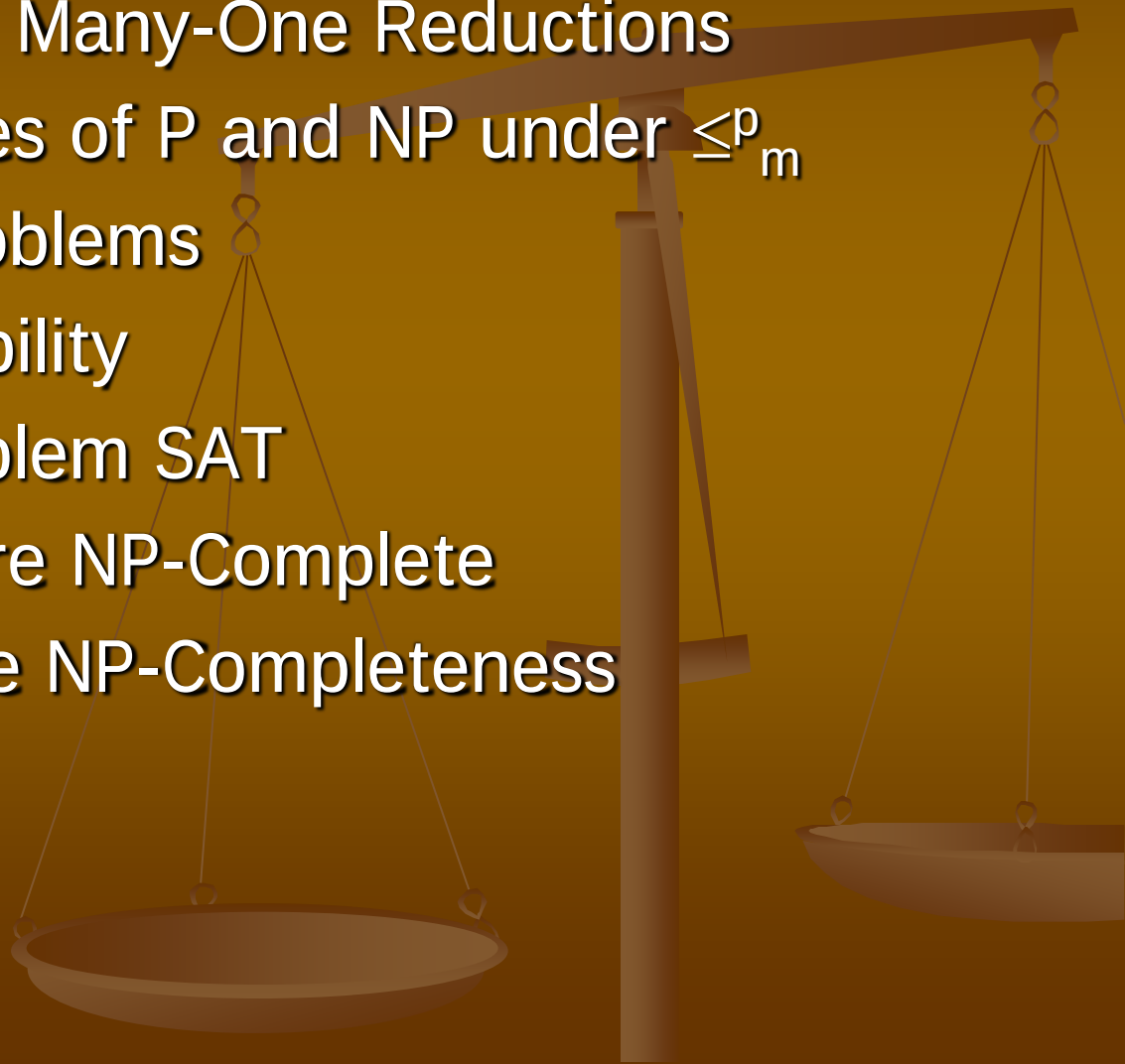
Open Problems

- Associated with P, NP, and co-NP, there are many questions that we do not know their answers at present.
- Here are some of the important open questions.

1. Does $L \in NP$ imply $L \in$ co-NP? (Equivalently, is NP = co-NP?)
2. Does $L \in NP \cap$ co-NP imply $L \in P$? (Equivalently, is P = $NP \cap$ co-NP?)

II. NP-Complete Problems

1. Polynomial-Time Many-One Reductions
2. Closure Properties of P and NP under $\leq^p_m$
3. NP-Complete Problems
4. Formula Satisfiability
5. Satisfiability Problem SAT
6. SAT and 3SAT are NP-Complete
7. How to Prove the NP-Completeness



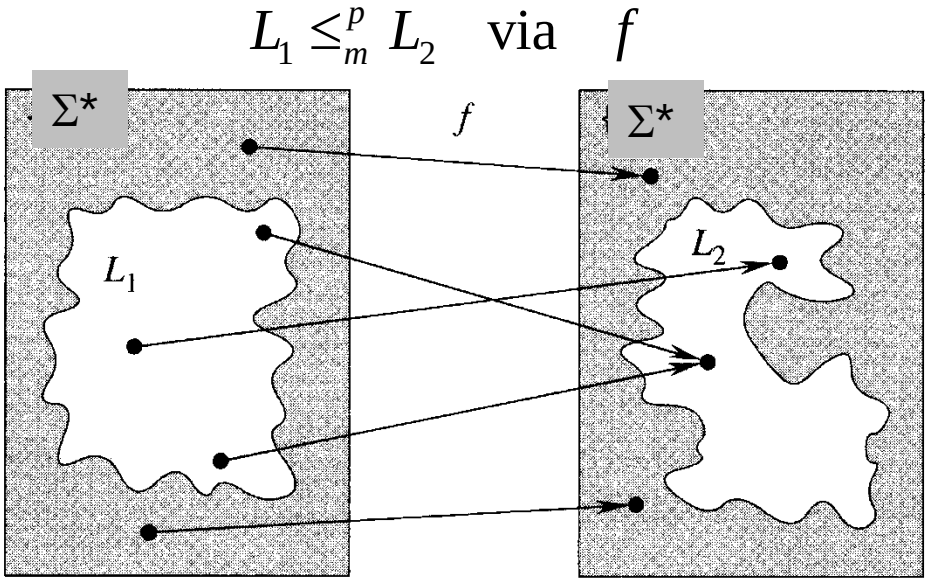
Polynomial-Time Many-One Reductions

- Recall from Week 1 the function class FP of polynomial-time computable functions.
- We say that problem A is polynomial-time (many-one) reducible to problem B if there exists a function f∈FP such that, for every x,

  $x \in A \leftrightarrow f(x) \in B.$

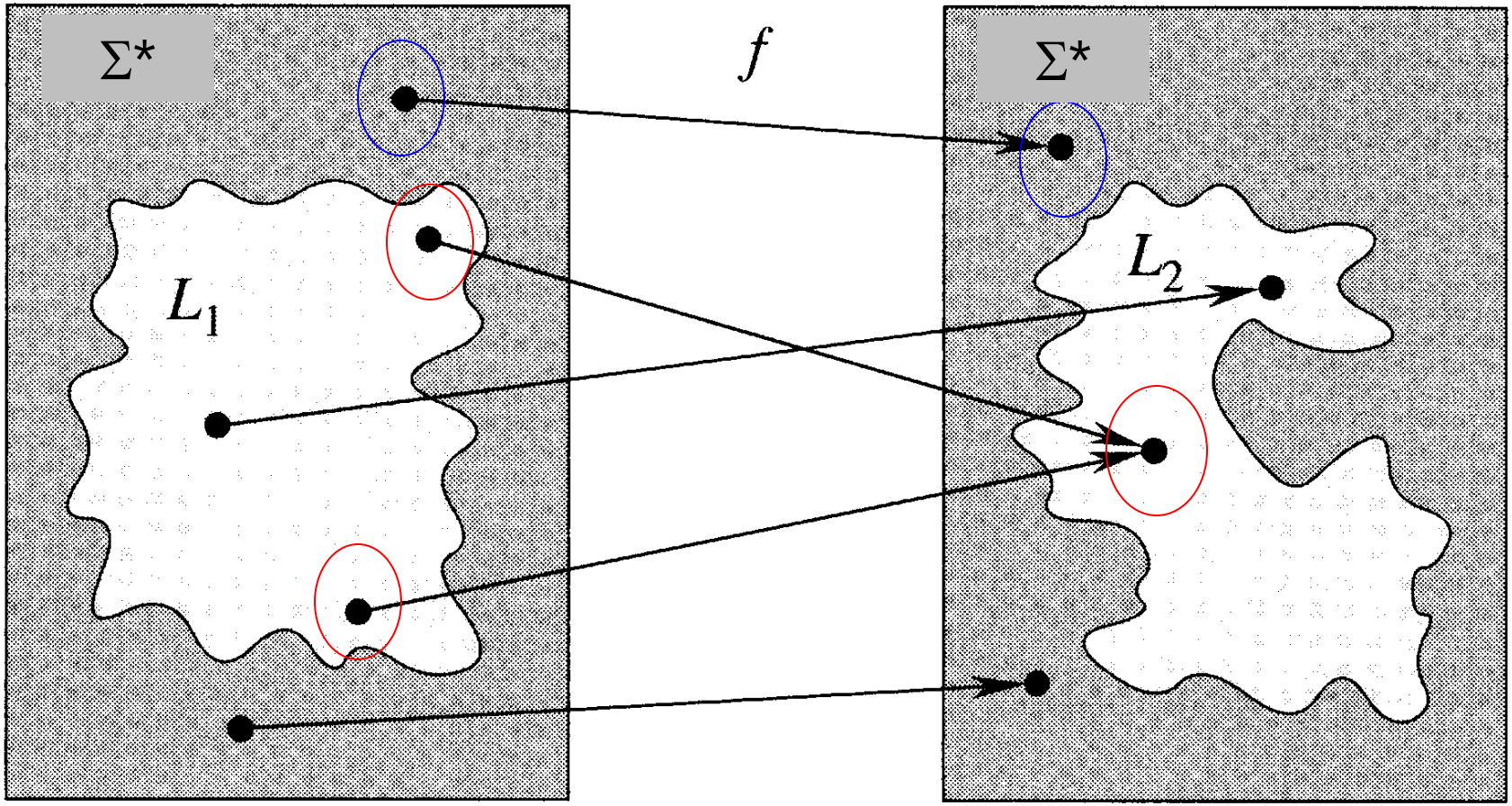
- In this case, we write:

$$A \leq_m^p B$$



(See the next slide.)

$L_1 \leq_m^p L_2$ via $f \iff \forall x[x \in L_1 \leftrightarrow f(x) \in L_2]$



Closure Properties of P and NP under $\leq^p_m$

- Consider closure properties under $\leq^p_m$.
- (Claim) If $L_1 \leq^p_m L_2$ and $L_2 \in P$, then $L_1 \in P$.
- (Claim) If $L_1 \leq^p_m L_2$ and $L_2 \in NP$, then $L_1 \in NP$.
- In other words, P and NP are closed under $\leq^p_m$-reductions.
- These closure properties are critical for the introduction of a completeness notion.

NP-Complete Problems I

- Polynomial-time reductions provide a formal means for showing that one problem is at least as hard as another, to within a polynomial-time factor.
- That is, if $L_1 \leq^p_m L_2$, then L_1 is not more than a polynomial factor harder than L_2.

- A language $L \subseteq \{0,1\}^*$ is called NP-hard (or many-one hard for NP) if

  for every language $A \in NP$, $A \leq^p_m L$.

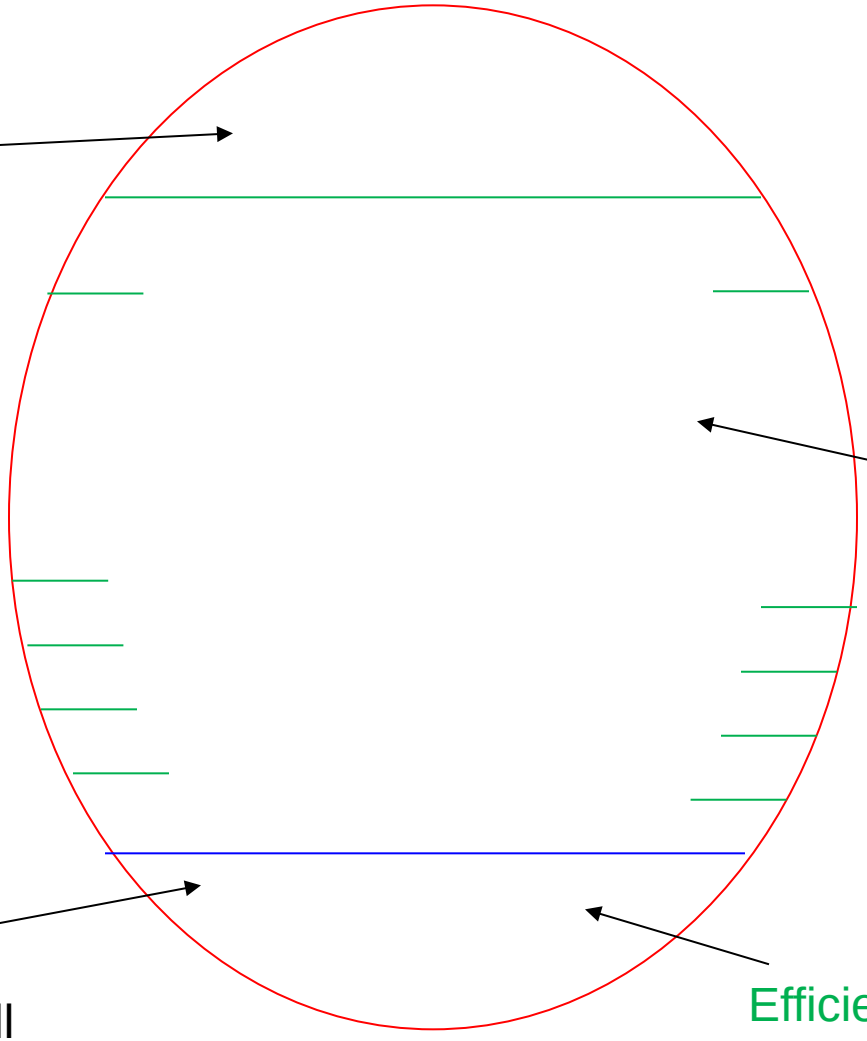
- A language L is called NP-complete (polynomial-time many-one complete for NP, or $\leq^p_m$-complete for NP) if
  1. $L \in NP$ and
  2. L is NP-hard.

NP-Complete Problems II

- In other words, a language L is called NP-complete if
  1. $L \in NP$ and
  2. for every language $A \in NP$, $A \leq^p_m L$.

- All NP-complete problems are the hardest problems in NP to solve in polynomial time.
- We sometimes write NPC to denote the class of all NP-complete languages (or NP-complete problems).
- There are hundreds of NP-complete problems discovered so far. (See, e.g., [Garey-Johnson (1979)].)

Inside of NP


NP
NPC
The set of all NP complete problems
NPI
The set of all problems having intermediate difficulty
P
The set of all P problems
Efficiently solvable problems

Formula Satisfiability

- Here, we formulate the (formula) satisfiability problem (SAT) in the form of language.
- An instance of SAT is a Boolean formula φ composed of
  1. n Boolean variables: $x_1, x_2, ..., x_n$;
  2. m Boolean connectives: $\wedge$ (AND), $\vee$ (OR), $\neg$ (NOT); and
  3. parentheses ("(" and ")").
- It is possible to encode any Boolean formula φ into a certain binary string of length that is polynomial in n+m.
- Hereafter, we always assume such an encoding.

Satisfiability Problem SAT

- A truth assignment for a Boolean formula φ is a set of values assigned to all variables of φ.
- A satisfying assignment for a Boolean formula φ is a truth assignment that causes φ to evaluate to 1.
- A formula with a satisfying assignment is a satisfiable formula. (See the next slide.)

- The satisfiability problem (SAT) is a decision problem:
  - instance: a Boolean formula φ;
  - question: is φ satisfiable?

Example: Satisfying Assignments

- Here is an example of a satisfiable formula.

$$\varphi \equiv \neg(\neg x_1 \vee x_2) \wedge \neg((\neg x_1 \wedge x_3) \vee x_2) \wedge x_4$$

- satisfying assignment $(x_1, x_2, x_3, x_4) = (1,0,0,1)$

$$\begin{aligned}
\varphi(1,0,0,1) &\equiv \neg(\neg 1 \vee 0) \wedge \neg((\neg 1 \wedge 0) \vee 0) \wedge 1 \\
&= \neg(0 \vee 0) \wedge \neg((0 \wedge 0) \vee 0) \\
&= \neg 0 \wedge \neg(0 \vee 0) \\
&= 1 \wedge 1 \\
&= 1
\end{aligned}$$

SAT and 3SAT are NP-Complete

- SAT is the first problem to be shown as an NP-complete problem.
- We restrict formulas to have 3-conjunctive normal form (3CNF), which has at most 3 literals in each clause.
- 3-Satisfiability Problem (3SAT)
  - instance: a 3CNF formula φ
  - question: is φ satisfiable?

A literal is either a variable or the negation of a variable.

- E.g., 3CNF: $\varphi \equiv (x_1 \vee x_2 \vee \neg x_3) \wedge (x_1 \vee \neg x_3) \wedge (\neg x_2 \vee x_3)$
- (Claim) SAT and 3SAT are NP-complete. [Cook (1971)]

How to Prove the NP-Completeness

- Once we find some NP-complete problems, it is rather easy to prove that other NP problems are also NP-complete.
- Her is a way to prove the NP-completeness of other problems.

- (Claim) Assume that A is a known NP-complete problem. If B is an NP problem and $A \leq^p_m B$, then B is NP-complete.

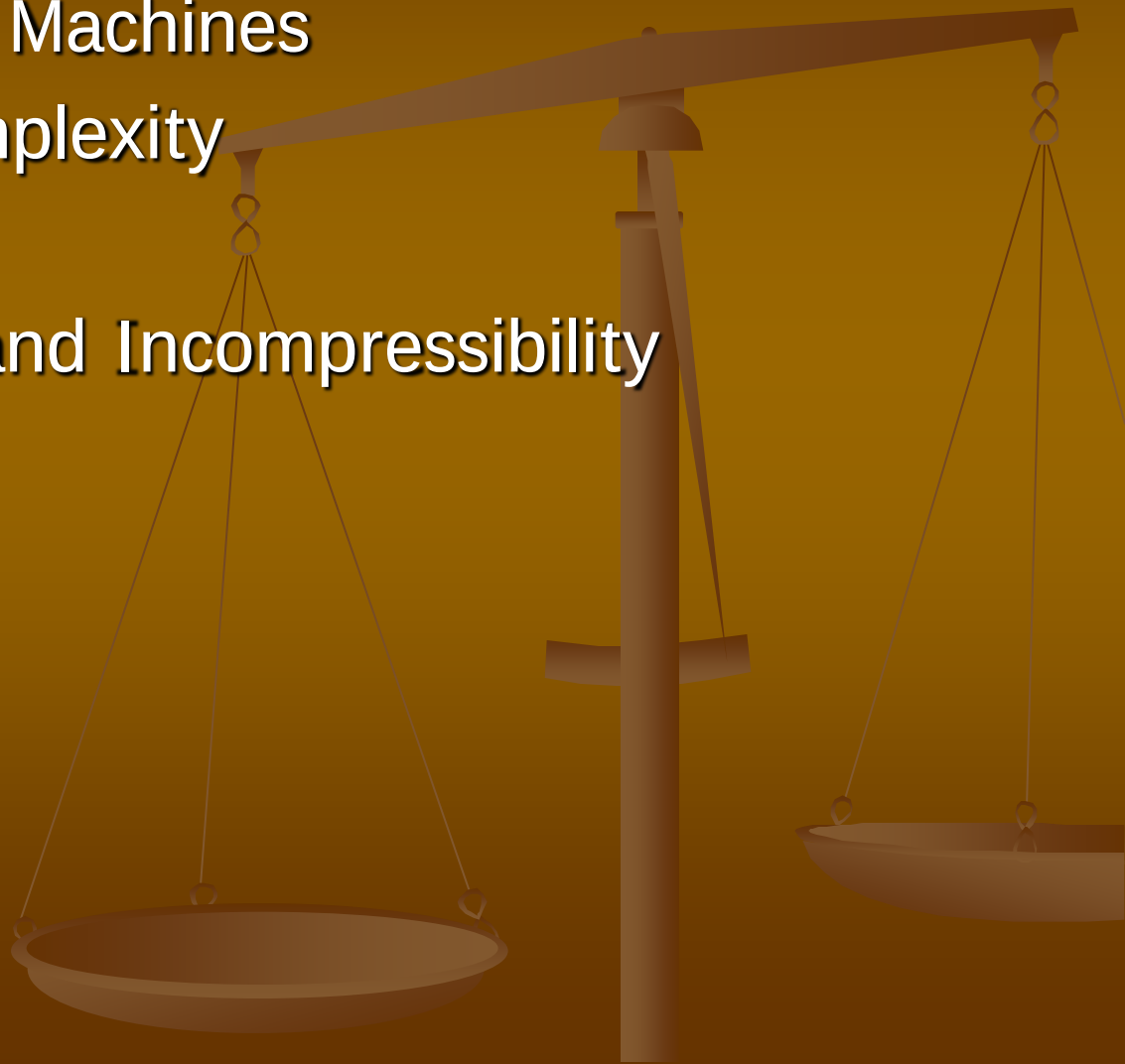
❑ Proof Sketch:

- Let C be any NP problem. Since A is NP-complete, it follows that $C \leq^p_m A$.
- If B satisfies $A \leq^p_m B$, then the transitivity property of $\leq^p_m$ implies that $C \leq^p_m B$.
- Hence, B is also NP-complete by definition.

QED

III. Kolmogorov Complexity

1. Universal Turing Machines
2. Kolmogorov Complexity
3. Basic Properties
4. Compressibility and Incompressibility



Universal Turing Machines

- Let us consider a universal Turing machine, which can simulates, on any input, any 1DTM equipped with an output tape and produces the same outputs whenever the original 1DTM halts.
- More precisely, a universal Turing machine is a DTM with an output tape that takes inputs of the form $\langle e(M),x \rangle$ and simulates M on input x, where e(M) denotes an appropriate binary encoding of a 1DTM M.
- We write U for a fixed universal Turing machine.
- Clearly, it follows that $U(\langle e(M),x \rangle) = M(x)$ for any 1DTM M and any input x whenever M(x) halts.
- Note that U takes a standard input x and any binary input p, which is considered to be a program (that is e(M)).

Kolmogorov Complexity

- Roughly, the Kolmogorov complexity of string x is the minimal size |y| of any binary string y such that U(y) = x.
- In other words, the Kolmogorov complexity of x means the size of the smallest program that produces x.
- bin(n) = binary representation of $n \in N$
- $x=x_1x_2...x_n \in \{0,1\}^n$.
- self-delimiting code of x : $x^{sdc} = 1^{|bin(|x|)|}0bin(|x|)x$.

- Conditional Kolmogorov complexity of x conditioned to y:

  $C(x|y) = \min\{ |p| : U(p^{sdc}y) = x, p \in \{0,1\}^* \}$
- Kolmogorov complexity of x:

  $C(x) = C(x|\lambda)$

Basic Properties

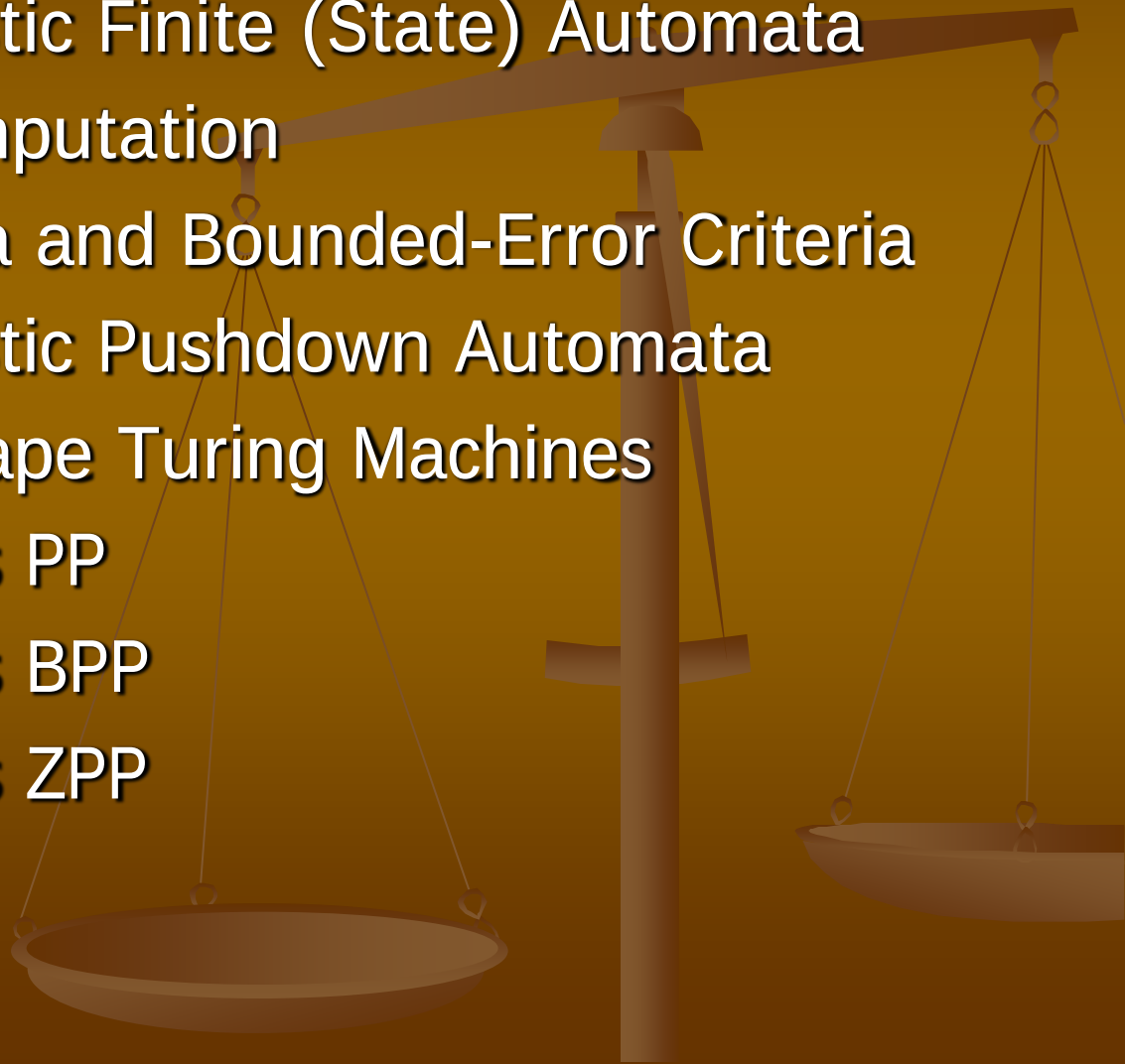
- Here are known properties of Kolmogorov complexity.
  a. $C(x|y) \leq C(x) \leq |x| + O(1)$
  b. $C(f(x)|y) \leq C(x|y) + O(1)$ for any recursive function f
  c. $C(x) \leq C(x|y) + C(y) + O(\min\{ \log|x|, \log|y| \})$
- Examples:
  - Let $x = 1^n$.
  - $C(1^n) = O(\log(n))$, compared to $|1^n|=n$.
  - To see this, consider the following program:
  - on input λ, retrieve "n" (in binary) from CPU memory (O(log(n) bits), and repeatedly output 1 for n times.

Compressibility and Incompressibility

- Let x be any binary string and let $n \in N$.
- x is compressible $\Leftrightarrow$ $C(x) < |x|$.
  Otherwise, x is incompressible.
- n is compressible $\Leftrightarrow$ $C(bin(n)) < \log(n)$.
  Otherwise, n is incompressible.
- (Claim) For any (sufficiently) large n, there exists an incompressible string of length n.
- An incompressible string is sometimes called algorithmically random, which is different from “statistical randomness.”
- (*) Kolmogorov complexity will be used shortly.

IV. Probabilistic Complexity Classes

1. 2-Way Probabilistic Finite (State) Automata
2. Probabilistic Computation
3. Cut-Point Criteria and Bounded-Error Criteria
4. 1-Way Probabilistic Pushdown Automata
5. Probabilistic 1-Tape Turing Machines
6. Complexity Class PP
7. Complexity Class BPP
8. Complexity Class ZPP



2-Way Probabilistic Finite Automata

Let us review a model of 2-way probabilistic finite automaton (or simply, 2pfa) with endmarkers.

$M = (Q, \Sigma, \{¢, \$\}, \delta, q_0, Q_{acc}, Q_{rej})$

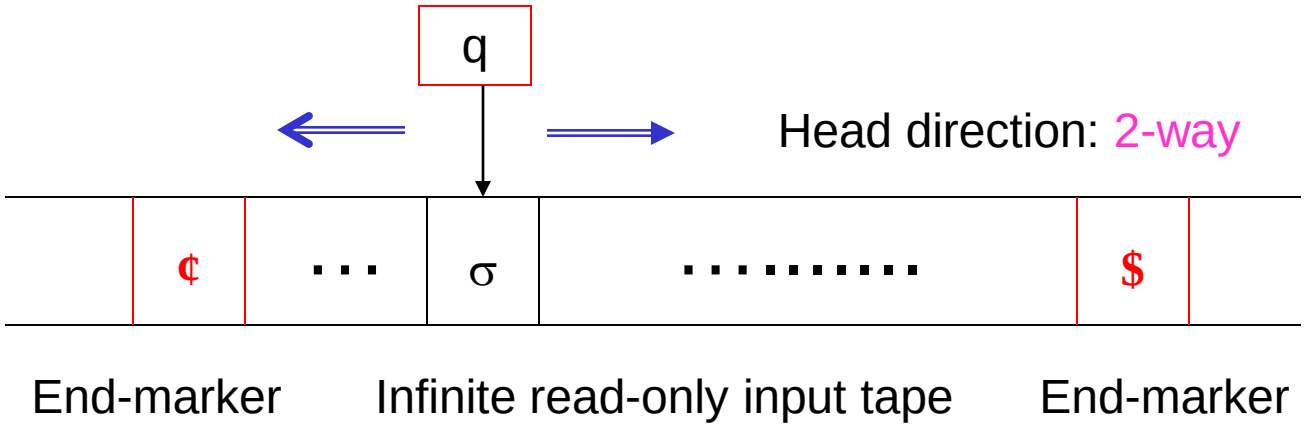
Σ = input alphabet

$Q_{acc} \cup Q_{rej} \subseteq Q$

This is quite different

δ : a probabilistic transition function

Inner state $q \in Q$



Formal Definition of 2pfa's

A 2pfa M = $(Q,\Sigma,\{¢,\$\},\delta,q_0,Q_{acc},Q_{rej})$ has a read-only input tape and a probabilistic transition function δ of the form:

$$\delta : Q \times \breve{\Sigma} \times Q \times D \rightarrow [0,1]$$

$\breve{\Sigma} = \Sigma \cup \{ ¢, \$ \}$ D = { -1, 0, +1 }

- Stochastic Requirement: $\forall(q,\sigma)\left[\sum_{(p,d)} \delta(q,\sigma,p,d) = 1\right]$
- Endmarker condition:
  - ✓ No tape head should move out of the region marked between ¢ and $.

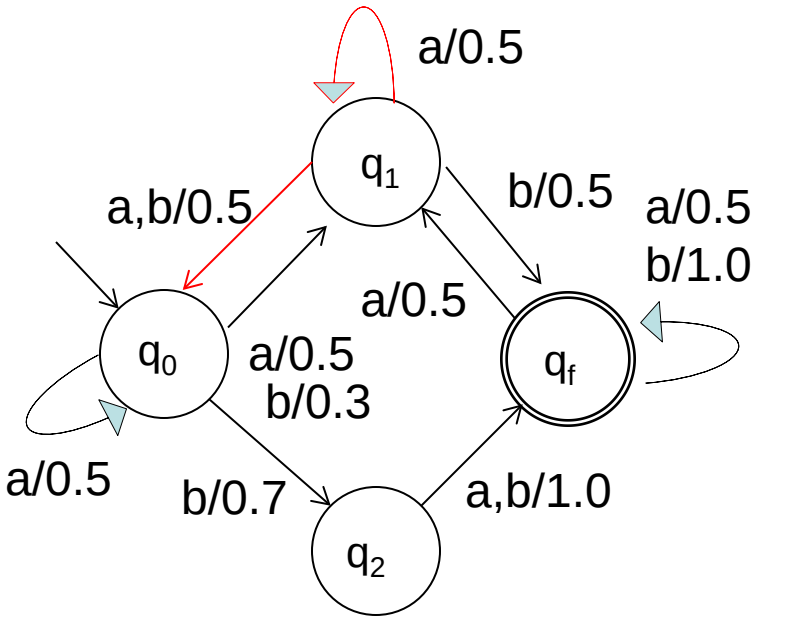
All probabilities sum up to 1.

- Similarly, we can define 1pfa's.

(See the next slide.)

Examples of 1pfa's (one-way case)

- As an example of 1pfa, let us consider the following simple 1pfa and its transition function (expressed as matrices).



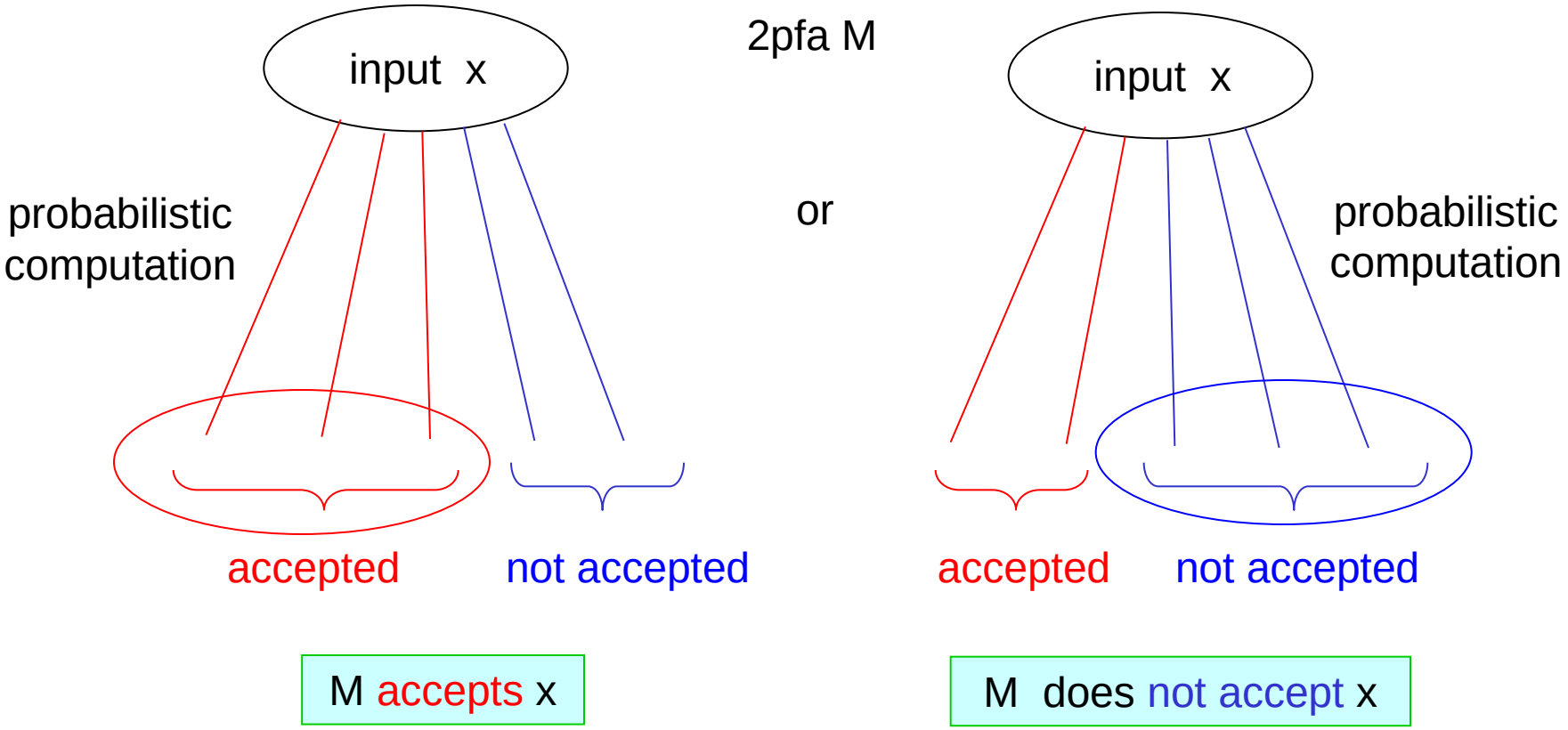
$$\begin{matrix} & A & & & x \\ \left[\begin{matrix} 0.5 & 0.5 & 0 & 0 \\ 0.5 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0.5 \end{matrix}\right]_a & & & & \begin{matrix} q_0 \\ q_1 \\ q_2 \\ q_f \end{matrix} \end{matrix}$$

=1

$$\left[\begin{matrix} 0 & 0.5 & 0 & 0 \\ 0.3 & 0 & 0 & 5 \\ 0.7 & 0 & 0 & 0 \\ 0 & 0.5 & 1.0 & 1.0 \end{matrix}\right]_b$$

Probabilistic Computation

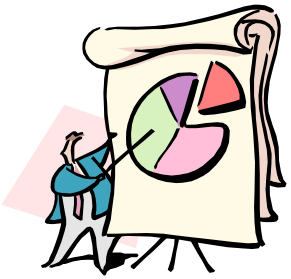
- A 2pfa produces **accepting/non-accepting computation paths** (which may or may not halt).



Cut-Point Criteria

- Rabin (1963) introduced a notion of "cut point".

- Let M be a 2pfa, let $\eta \in [0,1]$, and let $L \subseteq \Sigma^*$.
- $p_{M,acc}(x)$ = acceptance probability of M on input x
- A 2pfa recognizes language L with cut point η $\Leftrightarrow$ for all $x \in \Sigma^*$, $x \in L \leftrightarrow p_{M,acc}(x) \geq \eta$
- A 2pfa M is said to have an isolated cut point η for language L $\Leftrightarrow$ there exists a constant $\varepsilon \in [0,1/2)$ s.t., for all $x \in \Sigma^*$, (1) $x \in L \rightarrow p_{M,acc}(x) \geq \eta+\varepsilon$ and (2) $x \notin L \rightarrow p_{M,acc}(x) \leq \eta-\varepsilon$
- A 2pfa M is said to have an exact cut point η for language L $\Leftrightarrow$ for all $x \in \Sigma^*$, $x \in L \leftrightarrow p_{M,acc}(x) = \eta$



Bounded-Error Criteria

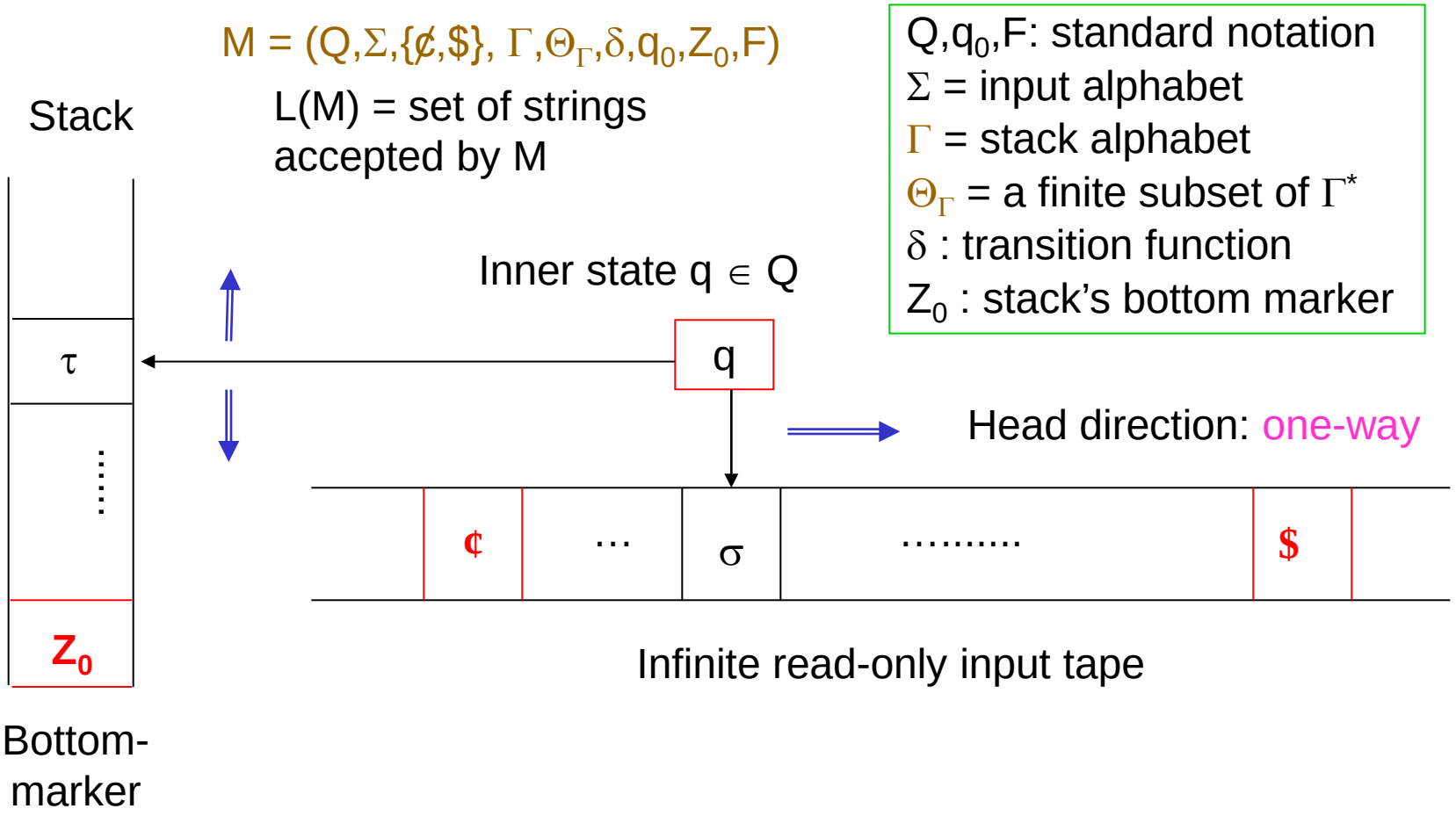
- Let M be a 2pfa, let $\eta \in [0,1]$, and let $L \subseteq \Sigma^*$.
- $p_{M,rej}(x)$ = rejection probability of M on input x
- A 2pfa M is said to have a bounded-error probability for language L $\Leftrightarrow$ there exists a constant $\varepsilon \in [0,1/2)$ s.t., for all $x \in \Sigma^*$, (1) $x \in L \rightarrow p_{M,acc}(x) \geq 1/2+\varepsilon$ and (2) $x \notin L \rightarrow p_{M,rej}(x) \geq 1/2+\varepsilon$
- A 2pfa recognizes language L with unbounded-error probability $\Leftrightarrow$ for all $x \in \Sigma^*$, (1') $x \in L \rightarrow p_{M,acc}(x) > 1/2$ and (2') $x \notin L \rightarrow p_{M,rej}(x) \geq 1/2$
- "Bounded-error probability" is, in essence, equivalent to "isolated cut point," but "unbounded-error probability" slightly deviates from "cut point."

Probabilistic Language Families

- rat-1pfa = one-way rational probabilistic finite automaton
- SL_{rat} = collection of all languages recognized by rat-1pfa's with cut point ½. Such languages are called stochastic languages.
- $SL^{=}_{rat}$ = collection of all languages L recognized by rat-1pfa's s.t. $\forall x$ [$x \in L \leftrightarrow$ M accepts x with exact cut point ½]
- (Claim) $REG \subseteq SL^{=}_{rat} \subseteq SL_{rat}$
- (Claim) SL_{rat} is also defined by rat-2pfa's with cut point ½. [Kaŋeps (1989)]
  - ✓ This means that there is no difference between 1pfa's and 2pfa's in case of cut point ½.
- Later, we will connect them to 1-tape linear-time classes.

1-Way Probabilistic Pushdown Automata

Let us review a model of 1-way (one head) probabilistic pushdown automaton (or 1ppda).



Probabilistic Transition Functions

- A 1ppda M uses a probabilistic transition function δ of the form:

$$\delta : Q \times (\breve{\Sigma} \cup \{\lambda\}) \times \Gamma \times Q \times \Theta_{\Gamma} \to [0,1]$$

  where $\breve{\Sigma} = \Sigma \cup \{¢,\$\}$.

- The notation $\sigma(q,\sigma,a|p,u) = \gamma$ means the following:
  - γ is the transition probability that M is currently in state q, scanning σ on an input tape and symbol a at the top of a stack, and M makes a move of replacing a by u with entering state p.

Formal Definition of 1ppda's

A 1ppda M = $(Q,\Sigma,\Gamma,\Theta_\Gamma,\delta,q_0,Q_{acc})$ has a read-only input tape, a stack, and a probabilistic transition function δ of the form:

$$\delta : Q \times (\breve{\Sigma} \cup \{\lambda\}) \times \Gamma \times Q \times \Theta_\Gamma \to [0,1] \quad \delta(q,\sigma,a \mid p,u) \in [0,1]$$

$\breve{\Sigma} = \Sigma \cup \{ ¢, \$ \}$

- Let $\delta[q,\sigma,a] = \sum_{(p,u)\in Q\times\Theta_\Gamma} \delta(q,\sigma,a \mid p,u)$
- Probability Requirement: $\forall(q,\sigma,a)\left[\delta[q,\sigma,a] + \delta[q,\lambda,a] = 1\right]$
- This extends the deterministic requirement for 1dpda's.

All probabilities sum up to 1.

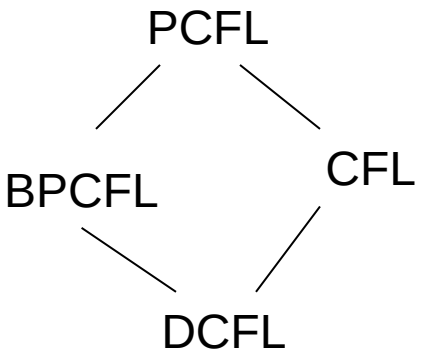
Probabilistic Language Families

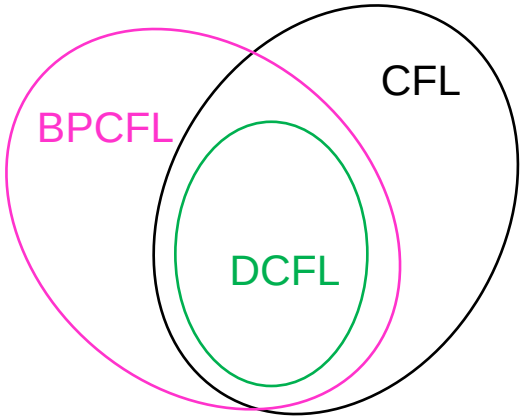
- Similarly to CFL, we define PCFL and BPCFL.
- PCFL = collection of all languages recognized by 1ppda's with unbounded-error probability
- BPCFL = collection of all languages L recognized by 1ppda's with bounded-error probability
- Let $\varepsilon \in [0,1/2)$ be any error bound.
- $BPCFL_\varepsilon$ = class of languages recognized by 1ppda's with error probability at most ε
- In particular, $BPCFL_0 = DCFL$
- $BPCFL = \cup_{\varepsilon \in [0,1/2)} BPCFL_\varepsilon$

Basic Relationships

- Here are simple known relationships among DCFL, BPCFL, and PCFL.
- (Claim) DCF $\subseteq$ BPCFL $\subseteq$ PCFL
- (Claim) BPCFL $\not\subset$ CFL and CFL $\not\subset$ BPCFL

  [Hromkovič-Schnitger (2010)]

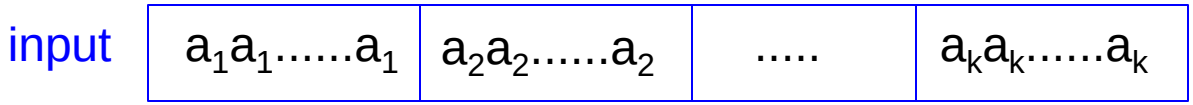




Many believe in this way

Example L_{keq} I

- We see a simple example of BPCFL-languages.
- Let $\Sigma_k = \{ a_1, a_2, ..., a_k \}$ for a constant $k \geq 1$.
- $L_{keq} = \{ a_1^n a_2^n ... a_k^n \mid n \geq 1 \}$. (bounded language)



(Claim) $L_{keq} \notin$ CFL for any $k \geq 3$ (by the pumping lemma or the swapping lemma (see Week 5)).

- (Claim) L_{keq} is in BPCFL for $k \geq 1$. [Hromkovič-Schnitger (2010)]

❑ Proof Sketch:

- Case k=1,2: Trivial because $L_{1eq} \in$ REG and $L_{2eq} \in$ DCFL.

Example L_{keq} II

- Case k=3: The following algorithm places L_{3eq} into BPCFL. The case of $k \geq 4$ is similar.
  1. Fix a sufficiently large constant $t \geq 1$.
  2. Let w be any nonempty input (i.e., $w \neq \lambda$).
  3. Check if $w = a_1^i a_2^j a_3^k$ for certain $i,j,k \geq 1$. If not, reject w. Otherwise, proceed with $i,j,k \geq 1$.
  4. Pick $s \in \{1,2,...,t\}$ uniformly at random.
  4. While reading one a_1, push s+1 0's.
  5. While reading one a_2, pop s 0s.
  6. While reading one a_3, pop one 0.
  7. If w is completely read and the stack is empty, then accept; otherwise, reject.

Example L_{keq} III

- Analysis:
  1. If i=j=k≥1, then M accepts for all s∈[t].
  2. Assume that i≠j or i≠k. If, for example, M accepts w for a pair s_1,s_2 ($s_1 \neq s_2$), then we obtain $(s_a+1)i-s_aj-k=0$ for a=1,2; that is,

$$\begin{cases} s_1(i-j)+(i-k)=0 \\ s_2(i-j)+(i-k)=0 \end{cases} \qquad (*)$$

  3. If i=j, then we obtain i=k, a contradiction. Thus, i≠j.
  4. Since i≠j, (*) then leads to $s_1=s_2$, a contradiction.
  5. Hence, there is no such pair s_1,s_2 ($s_1 \neq s_2$).
  6. This implies that M accepts w with prob. ≤ 1/t.

QED

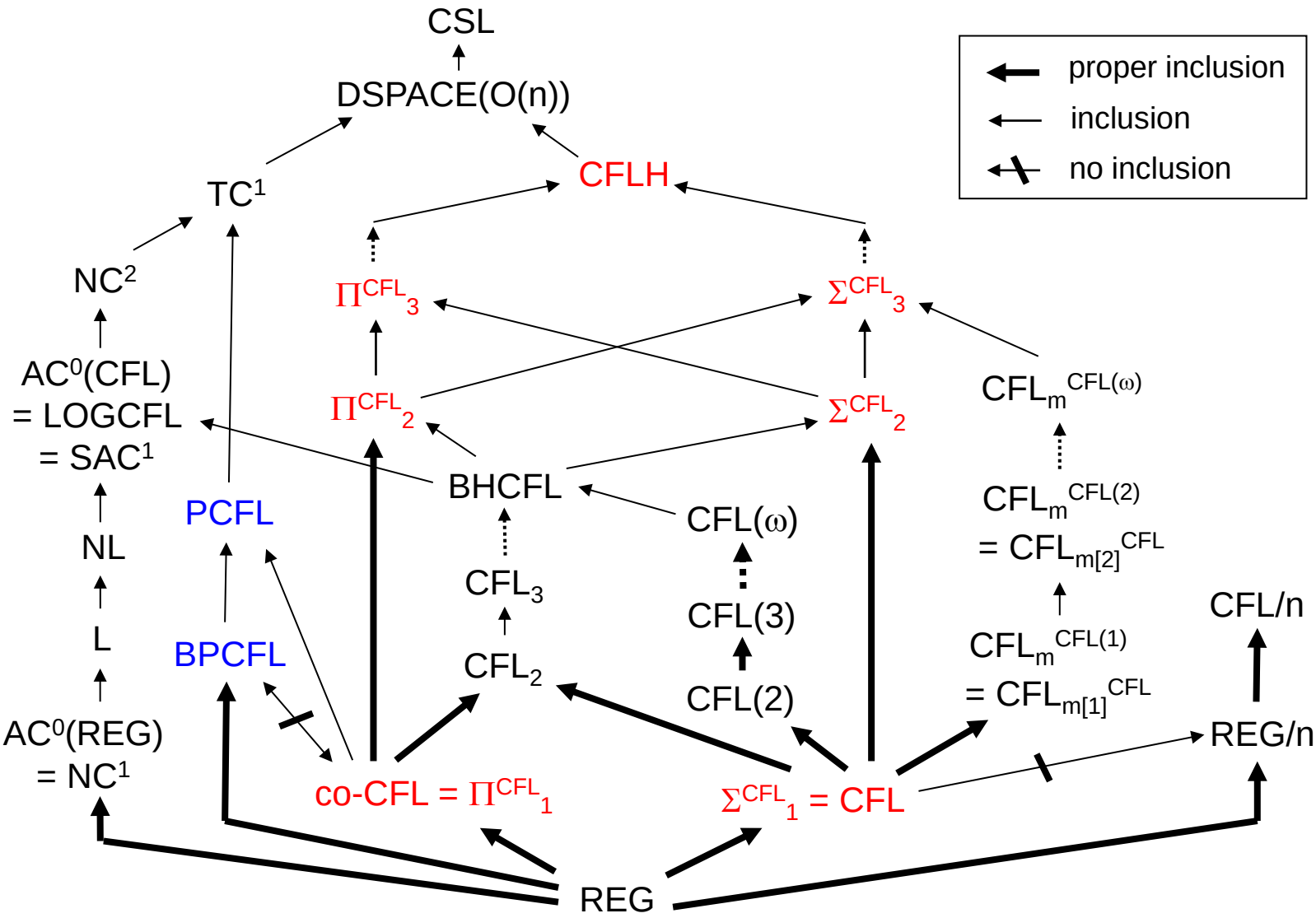
Known Results

- Freivalds
  - $\Sigma_k = \{ a_1, a_2, ..., a_k, b_1, b_2, ..., b_k \}$ for each $k \geq 1$
  - $\#_a(w)$ = # of occurrences of a in w
  - kEqual = $\{ w \in \Sigma^* \mid \forall i \in [k] \ \#_{ai}(w) = \#_{bi}(w) \}$
  - kEqual $\in$ BPCFL for all $k \geq 3$.
- Kaņeps, Geidmanis, Freivalds (1997)
  - TALLY $\cap$ BPCFL $\subseteq$ REG
- Yamakami (2014)
  - BPCFL $\not\subset$ CFL/n (with advice)
  - $\exists$ A: oracle s.t. $\text{BPCFL}^A \not\subset \Sigma^{\text{CFL},A}_2$ (see next slide)

2nd level of the CFL hierarchy (see Week 4)

TALLY = class of languages over single-letter alphabets

Inclusion Relations among Language Families



Complexity of Palindromes

- Theorem: [Yamakami (2017)]

  Pal = { w $\in\{0,1\}^*$ | w = w^R } is not in BPCFL.

❑ Proof Idea:

- The proof of the theorem uses Kolmogorov complexity.
- Li and Vitányi (1995) first proposed Kolmogorov complexity versions of the pumping lemmas for 1dfa's and 1dpda's.
- Glier (2003) gave a (corrected form of) Kolmogorov complexity version of the pumping lemma for 1dpda's.
- We extend Glier's result to handle 1ppda's and obtain a new pumping lemma for 1ppda's.

QED

Open Problems

- There are a number of problems left unsolved.

- Here is an open problem given by Hromkovič and Schnitger (2010).
  - Question: DISJ = { x#y | x ∩ y = ∅ } ∉ BPCFL?
  - Here, x and y are seen as sets of indices of “1”. For example, y=0100101 means {2,5,7}.

- We can ask the following question.
  - Let Center = { u1w | u,w ∈{0,1}*, |u|=|w| }.
  - Question: Is it true that Center ∉ BPCFL?

Probabilistic 1-Tape Turing Machines

- 1PTM = 1-tape probabilistic Turing machine using the strong definition for its running time
- 1-BPLIN = collection of all languages recognized by linear-time 1PTMs with bounded error (i.e., error < ½ -ε)
- 1-PLIN = collection of all languages recognized by linear-time 1PTMs with unbounded error (i.e., error < ½)
- 1-$C_{=}$LIN = collection of all languages L that are recognized by linear-time 1PTMs such that

  $\forall x$ [$x \in L \leftrightarrow$ M accepts x with probability exactly ½].

- (Claim)

  1. 1-BPLIN $\cup$ 1-$C_{=}$LIN $\subseteq$ 1-PLIN.
  2. 1-DLIN $\subseteq$ 1-BPLIN $\cap$ 1-$C_{=}$LIN.

Typical Examples

- The complexity classes 1-PLIN, 1-BPLIN, and 1-$C_{=}$LIN contain the following problems.

- Problems in 1-PLIN
  - Let $\text{Diff}_{<} = \{ a^n b^m \mid 1 \leq n < m \}$.
  - (Claim) $\text{Diff}_{<} \in \text{CFL} - \text{REG}$.
- Problems in 1-$C_{=}$LIN
  - Let $\text{Equal} = \{ a^n b^n \mid n \geq 1 \}$.
  - (Claim) $\text{Equal} \in \text{DCFL} - \text{REG}$.

- We can use the pumping lemma for regular languages to show that $\text{Diff}_{<}$ and Equal are not in REG.

Relationships among Complexity Classes

- Here is a short list of known results regarding the aforementioned complexity classes.

- Collapse results
  - 1-DLIN = 1-NLIN = 1-BPLIN = REG [Hennie65,Kobayashi85,Tadaki-Yamakami-Lin04]
  - $1\text{-}C_{=}LIN = SL^{=}_{rat}$ [Tadaki-Yamakami-Lin (2004)]
  - $1\text{-}PLIN = SL_{rat}$ [Tadaki-Yamakami-Lin (2004)]

- Separation results
  - $1\text{-}C_{=}LIN \neq 1\text{-}PLIN$ [Turakainen (1969)]
  - $1\text{-}C_{=}LIN \neq co\text{-}1\text{-}C_{=}LIN$ [Dieu (1971)]

Complexity Class PP

- We introduce a complexity class defined by probabilistic Turing machines (or PTMs).
- A decision problem (or a language) L is in PP if there is a probabilistic Turing machine M such that, for any input x,
  1. $x \in L \rightarrow$ M accepts x with probability > 1/2,
  2. $x \notin L \rightarrow$ M rejects x with probability $\geq$ 1/2, and
  3. M halts in polynomial time.
- When M satisfies Conditions 1-2, we say that M makes unbounded-error probability.

Natural Problems in PP

- Complexity class PP contains the following problems.

- Majority Satisfiability Problem (Majority-SAT)
  - instance: a Boolean formula φ
  - question: YES if more than half of all assignments make φ true; NO otherwise.

- E.g., $\varphi \equiv (x_1 \vee x_2) \wedge (x_1 \vee x_3 \vee \neg x_4) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$

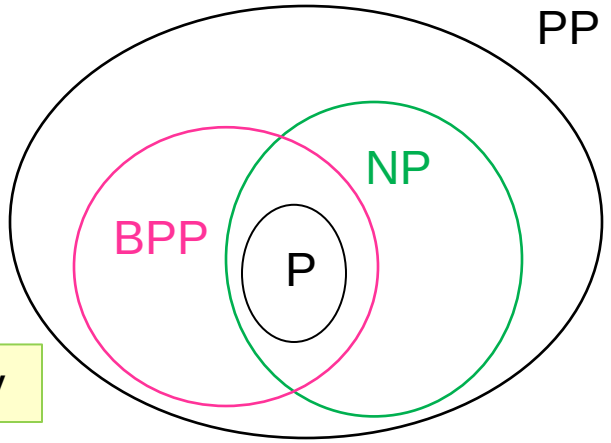
  question: $\left|\left\{(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \mid \varphi(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \equiv 1\right\}\right| > 2^4/2$?

- (Claim) PP is closed under union, intersection, and complementation. [Beigel-Reinold-Spielman (1991)]

Complexity Class BPP

- A decision problem (or a language) L is in BPP if there are a PTM M and a constant (an error bound) $\varepsilon \in [0,1/2)$ such that, for any input x,
  1. $x \in L \rightarrow$ M accepts x with probability $\geq 1-\varepsilon$,
  2. $x \notin L \rightarrow$ M rejects x with probability $\geq 1-\varepsilon$, and
  3. M halts in polynomial time.
- When M satisfies Conditions 1-2, we say that M makes bounded-error probability.

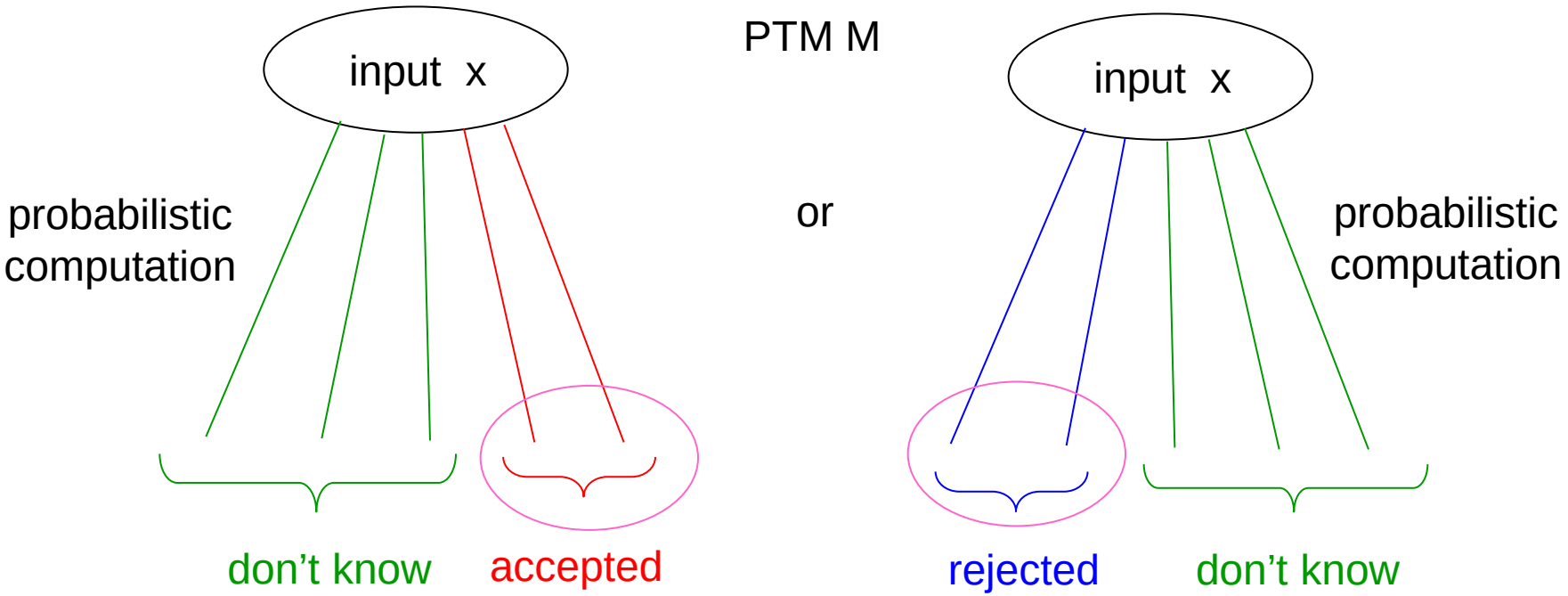
- (Claim) $P \subseteq BPP \subseteq PP$.
- (Claim) $P \subseteq NP \subseteq PP$.



Many believe in this way

Zero-Error Probabilistic Computation

- Here, we consider a slightly different probabilistic model.
- We allow PTMs to reach three distinguished outcomes along each computation path: “accept,” “reject,” and “don’t know.”
- The “don’t know” state is treated as a halting state but neither accepting states nor rejecting states.



Complexity Class ZPP

- A decision problem L is in ZPP if there are a PTM M and a constant $\varepsilon \in [0,1/2)$ such that, for any input x,
  1. $x \in L \rightarrow$ M outputs either "accept" or "don't know,"
  2. $x \notin L \rightarrow$ M outputs either "reject" or "don't know,"
  3. The probability of producing "don't know" on each input is at most 1/2, and
  4. M terminates in polynomial time.
- When M satisfies Conditions 1-3, we say that M makes zero-error probability.

- (Claim) $P \subseteq ZPP \subseteq BPP$.
- (Claim) $ZPP \subseteq NP \cap$ co-NP.

Other Well-Known Complexity Classes

- There are a number of complexity classes that are well-known in use. Here is two of them.
  - RP = one-sided version of PP
  - co-RP = complement class of RP
  - Note that ZPP = RP $\cap$ co-RP.
- For more complexity classes, see Complexity Zoo:
  https://complexityzoo.uwaterloo.ca/Complexity_Zoo

Known Results

- There are numerous results known for probabilistic complexity classes.

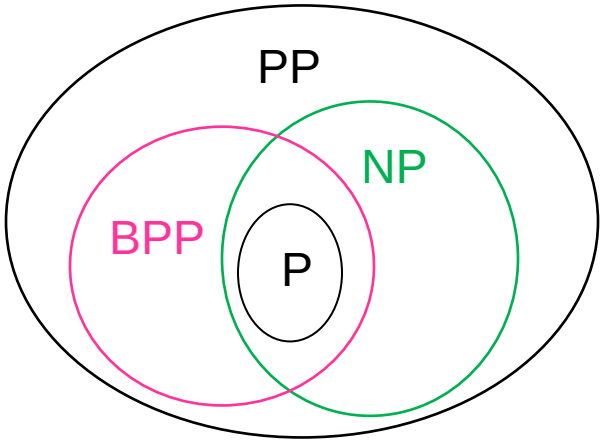
- (Claims)
  1. If NP $\subseteq$ BPP, then RP = NP. [Ko (1982)]
  2. If NP $\subseteq$ BPP, then PH $\subseteq$ BPP. [Zachos (1988)]
  3. BPP $\subseteq \Sigma_2^p \cap \Pi_2^p$. [Sipser-Gacs (1983)]
  4. BPP^{BPP} = BPP. [Ko (1982), Zachos (1982)]
  5. $PP^{PH} \subseteq BPP^{C=P} \subseteq P^{PP}$. [Toda (1991)]

- (*) Relativizations and the polynomial hierarchy will be discussed in Week 4.

Open Problems

- There are a number of problems that have not been solved in the past literature.
- We list some of them below.

Many believe in this way.



  - Is P = BPP?
  - Is NP $\subseteq$ BPP?
  - Is P = PP?
  - Is BPP = PP?

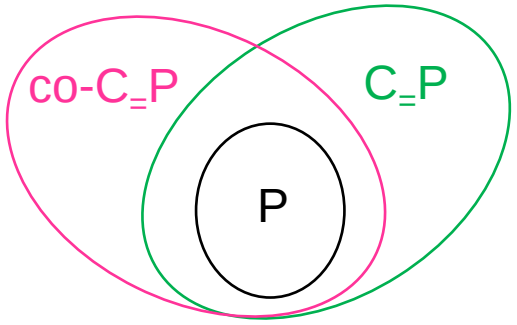
- A certain number of researchers nowadays believe that P = BPP, that is, the use of probabilistic computation does not help.

V. Counting Complexity Classes

1. Complexity Class $C_{=}P$
2. PP as a Counting Complexity Class
3. Simple Inclusion Relationships

Complexity Class $C_{=}P$

- A decision problem (or a language) L is in $C_{=}P$ if there are an NTM M and a function $f : \Sigma^* \to N$ in FP such that, for any input x,
  1. $x \in L \leftrightarrow$ the number of accepting computation paths of M on x is f(x), and
  2. M halts in polynomial time.
- In other words, $L = \{ x \mid \#M(x) = f(x) \}$, where $\#M(x)$ denotes the number of accepting computation paths of M on input x.
- Surprisingly, it is possible to fix f as $f(x) = \#M(x)/2$.



Natural Problems in $C_{=}P$

- Complexity class $C_{=}P$ was first defined by Wagner (1986).
- This complexity class contains the following problems.

- Equality Satisfiability Problem (Equal-SAT)
  - instance: a Boolean formula φ
  - question: YES if exactly half of all assignments make φ true; NO otherwise.

- E.g., $\varphi \equiv ((x_1 \wedge x_2) \vee ((\neg x_1 \vee x_3) \vee \neg x_4)) \wedge (x_2 \vee \neg x_3 \vee \neg x_4)$

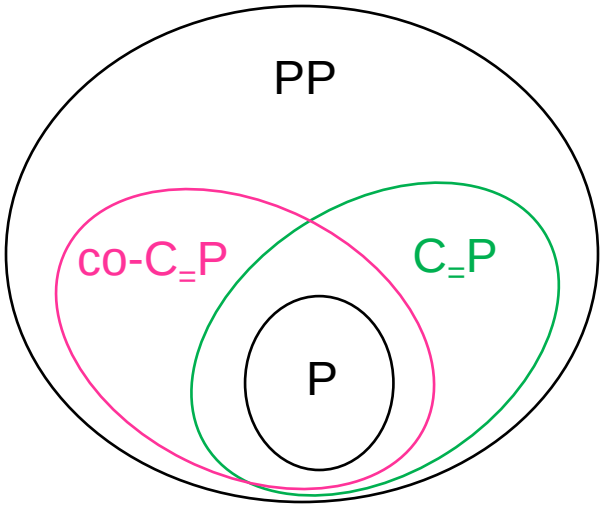
  question: $\left|\{(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \mid \varphi(\alpha_1, \alpha_2, \alpha_3, \alpha_4) \equiv 1\}\right| = 2^4/2$?

PP as a Counting Complexity Class

- We have already seen the complexity class PP.
- This complexity class PP is also considered as a counting complexity class.

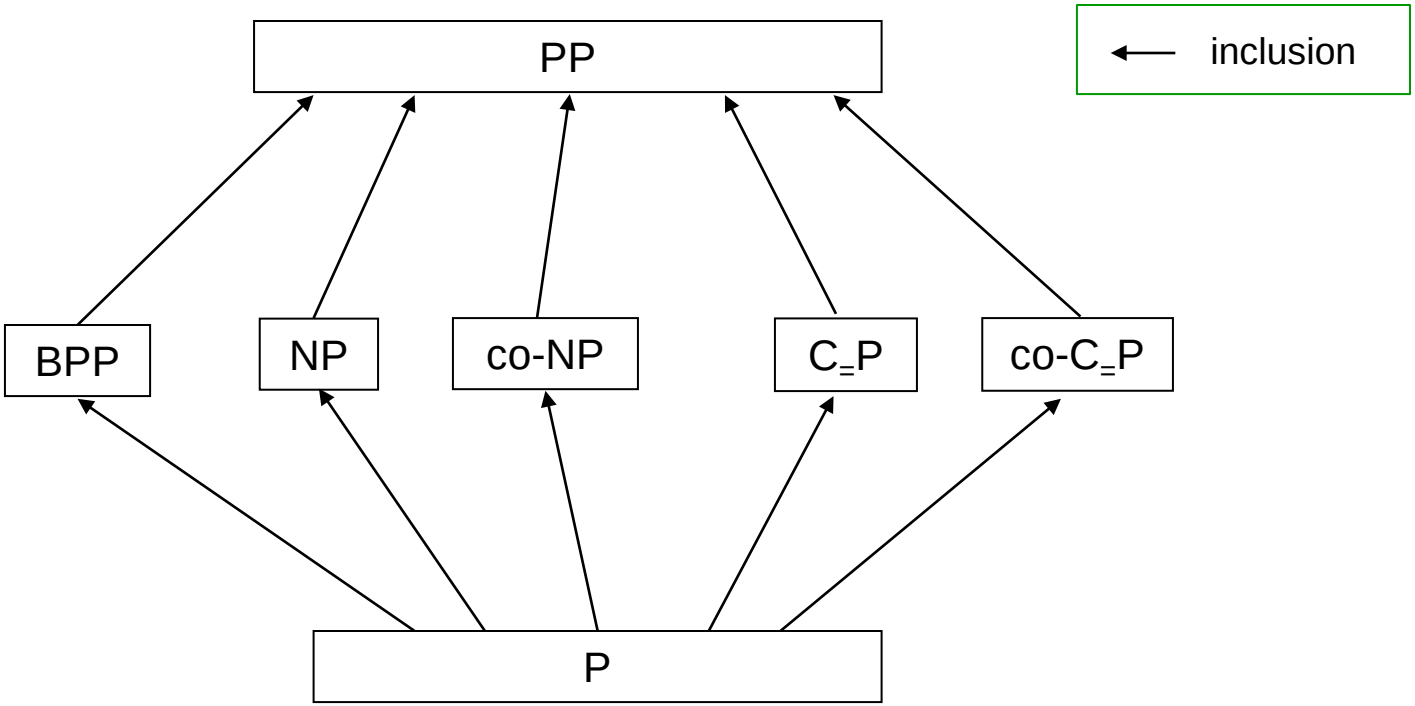
- (Claim) $P \subseteq C_{=}P \cap \text{co-}C_{=}P$.
- (Claim) $C_{=}P \cup \text{co-}C_{=}P \subseteq PP$.
  [Simon (1975)]

Many believe in this way.



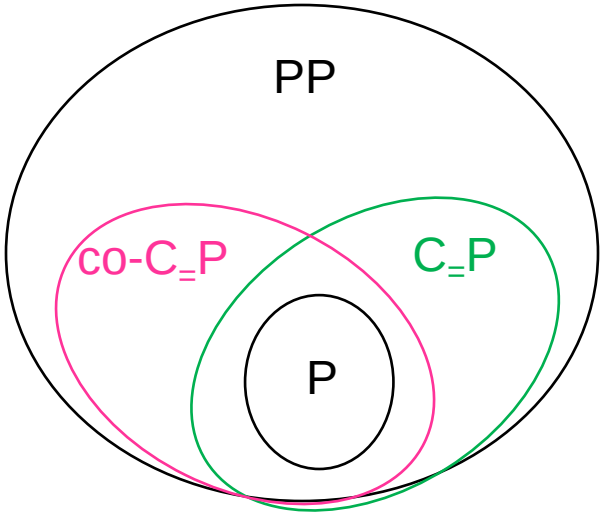
Simple Inclusion Relationships

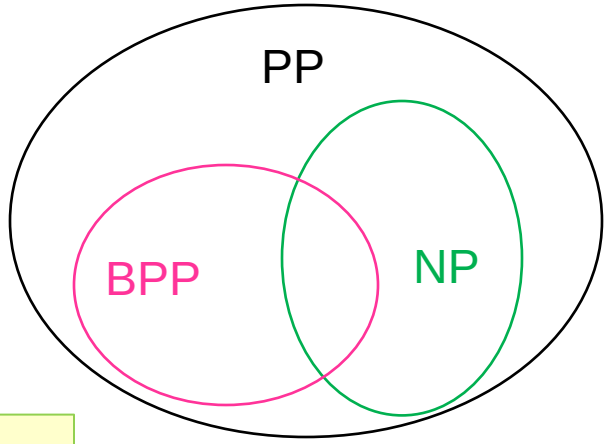
- Here are class inclusions among the aforementioned complexity classes.



Open Problems

- The following questions are not yet answered.
  - Is $P = C_{=}P$ or $P = \text{co-}C_{=}P$?
  - Is $C_{=}P \cup \text{co-}C_{=}P = PP$?
  - Is $C_{=}P = \text{co-}C_{=}P$?
  - Is $NP = C_{=}P$?
  - Is $P = BPP$?
  - Is $NP \subseteq BPP$?





Many believe in this way.

Other Well-Known Complexity Classes

- There are a number of complexity classes that are well-known for use and analysis.
  - US, FewP, SPP, $\oplus$P
  - IP, MIP, P-sel, AM, MA
  - OptP
- For more complexity classes, see Complexity Zoo:
  https://complexityzoo.uwaterloo.ca/Complexity_Zoo

Thank you for listening

Q & A

I’m happy to take your question!

END