


Probability and Computing – Randomised Complexity Classes

Stefan Walzer, Maximilian Katzmann, (Thomas Worsch) | WS 2023/2024



The Second Half of the Semester

- Randomised Complexity Classes
↔ full lecture notes by Thomas Worsch 
- Game Theory and Yao's Principle
↔ some lecture notes by Thomas Worsch
- Randomised Approximation
↔ full lecture notes by Thomas Worsch
- Streaming Algorithms
- Randomised *Data Structures*
 - Hash Functions
 - application: linear probing hash table
 - application: linear chaining hash table
 - Bloom Filters
 - Cuckoo Hashing
 - The Peeling Algorithm
 - Applications of Peeling

What are you missing?

The lecture by Thomas Worsch also covered

- routing in hypercubes
- an expected $\mathcal{O}(n)$ -time randomised MST algorithm
- online algorithms
- random walks
- Markov chains and Metropolis-Hastings
- pseudorandom number generation

Today: Decision Problems Only

- ~~approximation algorithms~~
- ~~average case analysis~~
- ~~data structures~~
- ~~function problems~~
- **decision problems**
 - for some language L such as $L = \text{PRIMES}$
 - decide for input x the question “is $x \in L$?”
 - can you do it in polynomial time?
 - does randomisation help?

Turing machines

(Non-) deterministic Turing machine

- S : finite state set
- B : finite tape alphabet including blank symbol \square
- $A \subseteq B - \{\square\}$: input alphabet
- one tape, one head
- transition functions
 - *deterministic*: one
 $\delta : S \times B \rightarrow (S \cup \{\text{YES, NO}\}) \times B \times \{-1, 0, 1\}$
 - *non-deterministic* two (or more)
 $\delta_0, \delta_1 : S \times B \rightarrow (S \cup \{\text{YES, NO}\}) \times B \times \{-1, 0, 1\}$
(alternatively: general transition *relation*)
 - in states YES and NO: “ T halts”
- accepted language
 $L(T) = \{w \in A^+ \mid \exists \text{YES-computation for } w\}$

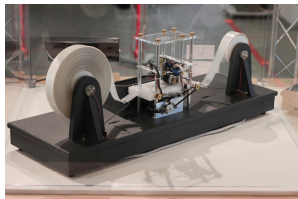


Photo: Rocky Acosta

(Non-) deterministic Turing machine

- S : finite state set
- B : finite tape alphabet including blank symbol \square
- $A \subseteq B - \{\square\}$: input alphabet
- one tape, one head
- transition functions
 - *deterministic*: one
 $\delta : S \times B \rightarrow (S \cup \{\text{YES, NO}\}) \times B \times \{-1, 0, 1\}$
 - *non-deterministic* two (or more)
 $\delta_0, \delta_1 : S \times B \rightarrow (S \cup \{\text{YES, NO}\}) \times B \times \{-1, 0, 1\}$
(alternatively: general transition *relation*)
 - in states YES and NO: “ T halts”
- accepted language
 $L(T) = \{w \in A^+ \mid \exists \text{YES-computation for } w\}$

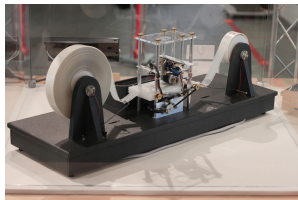


Photo: Rocky Acosta

Probabilistic Turing machine

- definition like non-deterministic TM
- uses δ_0 or δ_1 with probability 1/2 in each step
- output $T(w)$ is random variable
- difference to NTM:
 - *quantified* non-determinism
 - can study e.g. *probability* of acceptance

When is a PTM polynomial time?

Annoying

Running time for input x is random variable $T(x) \in \mathbb{N} \cup \{\infty\}$.

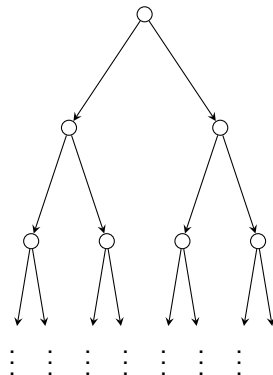
When is a PTM polynomial time?

Annoying

Running time for input x is random variable $T(x) \in \mathbb{N} \cup \{\infty\}$.

Simplification for Today: PTM in normal form

- For all inputs of length n , the PTM *halts* and does so after the *same number of steps* $t(n)$. \leftrightarrow this is without loss of generality under weak conditions
- computation tree of a PTM in normal form is complete binary tree of depth $t(n)$.
- call $t(n)$ the *running time*
- PTM runs in *polynomial time*, if $t(n) \leq p(n)$ for a polynomial $p(n)$.
- acceptance probability is the number of accepting computations, divided by $2^{t(n)}$.



“Classic” Complexity Classes

| class \mathcal{C} | requirement for $L \in \mathcal{C}$ |
|---------------------|-------------------------------------|
| P | polynomial time DTM can decide L |
| NP | polynomial time NTM can decide L |
| PSPACE | polynomial space TM can decide L |

“Classic” Complexity Classes

| class \mathcal{C} | requirement for $L \in \mathcal{C}$ |
|---------------------|-------------------------------------|
| P | polynomial time DTM can decide L |
| NP | polynomial time NTM can decide L |
| PSPACE | polynomial space TM can decide L |

Complement Classes

For class \mathcal{C} let $\text{co-}\mathcal{C} = \{L \mid \bar{L} \in \mathcal{C}\} = \{\bar{L} \mid L \in \mathcal{C}\}$, e.g.

- **$\mathbf{P} = \text{co-P}$**
- **$\mathbf{P} \subseteq \mathbf{NP} \cap \text{co-NP}$**
- relationship between **NP** and **co-NP** unknown
- **$\mathbf{NP} \cup \text{co-NP} \subseteq \mathbf{PSPACE}$**

“Classic” Complexity Classes

| class \mathcal{C} | requirement for $L \in \mathcal{C}$ |
|---------------------|-------------------------------------|
| P | polynomial time DTM can decide L |
| NP | polynomial time NTM can decide L |
| PSPACE | polynomial space TM can decide L |

Complement Classes

For class \mathcal{C} let $\text{co-}\mathcal{C} = \{L \mid \bar{L} \in \mathcal{C}\} = \{\bar{L} \mid L \in \mathcal{C}\}$, e.g.

- **P = co-P**
- **P \subseteq NP \cap co-NP**
- relationship between **NP** and **co-NP** unknown
- **NP \cup co-NP \subseteq PSPACE**

Polynomial time reduction from L_1 to L_2

- in polynomial time computable function $f : A^+ \rightarrow A^+$, such that
- $\forall w \in A^+ : w \in L_1 \iff f(w) \in L_2$.

\hookrightarrow then e.g. $L_2 \in \text{NP}$ implies $L_1 \in \text{NP}$.

“Classic” Complexity Classes

| class \mathcal{C} | requirement for $L \in \mathcal{C}$ |
|---------------------|-------------------------------------|
| P | polynomial time DTM can decide L |
| NP | polynomial time NTM can decide L |
| PSPACE | polynomial space TM can decide L |

Complement Classes

For class \mathcal{C} let $\text{co-}\mathcal{C} = \{L \mid \bar{L} \in \mathcal{C}\} = \{\bar{L} \mid L \in \mathcal{C}\}$, e.g.

- **P = co-P**
- **P \subseteq NP \cap co-NP**
- relationship between **NP** and **co-NP** unknown
- **NP \cup co-NP \subseteq PSPACE**

Polynomial time reduction from L_1 to L_2

- in polynomial time computable function $f : A^+ \rightarrow A^+$, such that
- $\forall w \in A^+ : w \in L_1 \iff f(w) \in L_2$.




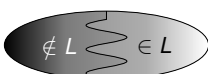
\hookrightarrow then e.g. $L_2 \in \text{NP}$ implies $L_1 \in \text{NP}$.

Hardness

- A language H is **\mathcal{C} -hard**, if every language $L \in \mathcal{C}$ can be reduced to H in polynomial time.
- A language is **\mathcal{C} -complete**, if it is \mathcal{C} -hard and in \mathcal{C} .

Probabilistic Complexity Classes

A language L is in class **P/RP/BPP/PP**, if there exists a probabilistic polynomial time turing machine T such that...

| class | name | requirement | visualisation |
|------------|---------------------------------------------|------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| P | polynomial time | $\forall w \notin L : \Pr[T(w) = \text{YES}] = 0$ $\forall w \in L : \Pr[T(w) = \text{YES}] = 1$ |  no error |
| RP | randomised polynomial time | $\forall w \notin L : \Pr[T(w) = \text{YES}] = 0$ $\forall w \in L : \Pr[T(w) = \text{YES}] \geq 1/2$ |  one-sided error |
| BPP | bounded-error probabilistic polynomial time | $\forall w \notin L : \Pr[T(w) = \text{YES}] < 1/4$ $\forall w \in L : \Pr[T(w) = \text{YES}] > 3/4$ |  two-sided error |
| PP | probabilistic polynomial time | $\forall w \notin L : \Pr[T(w) = \text{YES}] \leq 1/2$ $\forall w \in L : \Pr[T(w) = \text{YES}] > 1/2$ |  two-sided error |

ZPP := **RP** \cap **co-RP**. zero error probabilistic polynomial time
 \leftrightarrow requires *two* Turing machines, one for **RP**, one for **co-RP**.



We say a polynomial time PTM is an **RP-PTM**, **BPP-PTM** or **PP-PTM** if it is of the corresponding form.

Probability Amplification

Theorem

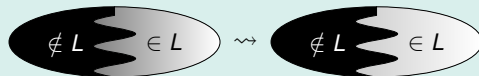
Instead of “ $1/2$ ” we can use “ $1 - 2^{-q(n)}$ ” in the definition of **RP** without affecting the class.



Probability Amplification

Theorem

Instead of “ $1/2$ ” we can use “ $1 - 2^{-q(n)}$ ” in the definition of **RP** without affecting the class.



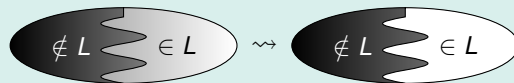
Proof.

Let T be the Turing machine witnessing $L \in \mathbf{RP}$.
By running T independently $q(n)$ times the error probability is $2^{-q(n)}$.
Running time increases by polynomial factor $q(n)$. \square

```
for  $i = 1$  to  $q(n)$  do
  if  $T(w) = \text{YES}$  then
    return YES
return NO
```

Theorem

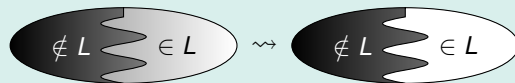
Instead of “ $1/4$ ” and “ $3/4$ ” we can use “ $2^{-q(n)}$ ” and “ $1 - 2^{-q(n)}$ ” in the definition of **BPP** without affecting the class.



Probability Amplification (2)

Theorem

Instead of “ $1/4$ ” and “ $3/4$ ” we can use “ $2^{-q(n)}$ ” and “ $1 - 2^{-q(n)}$ ” in the definition of **BPP** without affecting the class.



Proof.

Recommended (Bonus) Exercise.

↔ solution in lecture notes by Thomas Worsch

ZPP: Zero-Error-Probabilistic Polynomial Time

Theorem: $L \in \mathbf{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \mathbf{ZPP} := \mathbf{RP} \cap \text{co-RP}$ then there exists a PTM that

- decides L with no error
- has *expected* polynomial running time

\hookrightarrow this PTM is not in normal form

$$\begin{aligned}\mathbb{E}[X] &= \sum_{i \geq 0} i \cdot \Pr[X = i] \\ &= \sum_{i \geq 1} \Pr[X \geq i]\end{aligned}$$

ZPP: Zero-Error-Probabilistic Polynomial Time

Theorem: $L \in \text{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \text{ZPP} := \text{RP} \cap \text{co-RP}$ then there exists a PTM that

- decides L with no error
 - has *expected* polynomial running time
- \hookrightarrow this PTM is not in normal form

$$\begin{aligned}\mathbb{E}[X] &= \sum_{i \geq 0} i \cdot \Pr[X = i] \\ &= \sum_{i \geq 1} \Pr[X \geq i]\end{aligned}$$

Proof

Let T be an **RP**-PTM for L with running time $p(n)$.

\hookrightarrow never errs for $x \notin L$

Let \bar{T} be an **RP**-PTM for \bar{L} with running time $p(n)$.

\hookrightarrow never errs for $x \notin \bar{L}$



ZPP: Zero-Error-Probabilistic Polynomial Time

Theorem: $L \in \text{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \text{ZPP} := \text{RP} \cap \text{co-RP}$ then there exists a PTM that

- decides L with no error
 - has *expected* polynomial running time
- \hookrightarrow this PTM is not in normal form

$$\begin{aligned}\mathbb{E}[X] &= \sum_{i \geq 0} i \cdot \Pr[X = i] \\ &= \sum_{i \geq 1} \Pr[X \geq i]\end{aligned}$$

Proof

Let T be an **RP**-PTM for L with running time $p(n)$.

\hookrightarrow never errs for $x \notin L$

Let \bar{T} be an **RP**-PTM for \bar{L} with running time $p(n)$.

\hookrightarrow never errs for $x \notin \bar{L}$



repeat

$r_1 \leftarrow T(w)$

$r_2 \leftarrow \text{not } \bar{T}(w)$

until $r_1 = r_2$

return r_1

ZPP: Zero-Error-Probabilistic Polynomial Time

Theorem: $L \in \mathbf{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \mathbf{ZPP} := \mathbf{RP} \cap \text{co-RP}$ then there exists a PTM that

- decides L with no error
 - has *expected* polynomial running time
- \hookrightarrow this PTM is not in normal form

$$\begin{aligned}\mathbb{E}[X] &= \sum_{i \geq 0} i \cdot \Pr[X = i] \\ &= \sum_{i \geq 1} \Pr[X \geq i]\end{aligned}$$

Proof

Let T be an **RP**-PTM for L with running time $p(n)$.

\hookrightarrow never errs for $x \notin L$

Let \bar{T} be an **RP**-PTM for \bar{L} with running time $p(n)$.

\hookrightarrow never errs for $x \notin \bar{L}$

- T and \bar{T} never *both* answer incorrectly \Rightarrow we always answer correctly.
- Every round gives $r_1 = r_2$ with probability $\geq 1/2$.



repeat

$r_1 \leftarrow T(w)$

$r_2 \leftarrow \text{not } \bar{T}(w)$

until $r_1 = r_2$

return r_1

ZPP: Zero-Error-Probabilistic Polynomial Time

Theorem: $L \in \mathbf{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \mathbf{ZPP} := \mathbf{RP} \cap \text{co-RP}$ then there exists a PTM that

- decides L with no error
- has *expected* polynomial running time
 \hookrightarrow this PTM is not in normal form

$$\begin{aligned}\mathbb{E}[X] &= \sum_{i \geq 0} i \cdot \Pr[X = i] \\ &= \sum_{i \geq 1} \Pr[X \geq i]\end{aligned}$$

Proof

Let T be an \mathbf{RP} -PTM for L with running time $p(n)$.

\hookrightarrow never errs for $x \notin L$

Let \bar{T} be an \mathbf{RP} -PTM for \bar{L} with running time $p(n)$.

\hookrightarrow never errs for $x \notin \bar{L}$

- T and \bar{T} never *both* answer incorrectly \Rightarrow we always answer correctly.
- Every round gives $r_1 = r_2$ with probability $\geq 1/2$.



repeat

$r_1 \leftarrow T(w)$
 $r_2 \leftarrow \text{not } \bar{T}(w)$

until $r_1 = r_2$

return r_1

$$\mathbb{E}[\text{running time}] \leq 2p(|w|) \cdot \mathbb{E}[\#\text{rounds}] \leq 2p(n) \cdot \sum_{i \geq 1} \Pr[\#\text{rounds} \geq i] \leq 2p(n) \cdot \sum_{i \geq 1} 2^{-(i-1)} = 2p(n) \cdot \sum_{i \geq 0} 2^{-i} = 4p(n). \quad \square$$

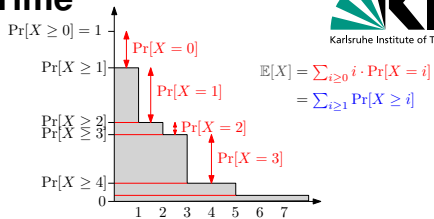
ZPP: Zero-Error-Probabilistic Polynomial Time

Theorem: $L \in \text{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \text{ZPP} := \text{RP} \cap \text{co-RP}$ then there exists a PTM that

- decides L with no error
- has *expected* polynomial running time

\hookrightarrow this PTM is not in normal form



Proof

Let T be an **RP**-PTM for L with running time $p(n)$.
 \hookrightarrow never errs for $x \notin L$

Let \bar{T} be an **RP**-PTM for \bar{L} with running time $p(n)$.
 \hookrightarrow never errs for $x \notin \bar{L}$



- T and \bar{T} never *both* answer incorrectly \Rightarrow we always answer correctly.
- Every round gives $r_1 = r_2$ with probability $\geq 1/2$.

```
repeat
  r1 ← T(w)
  r2 ← not T-bar(w)
until r1 = r2
return r1
```

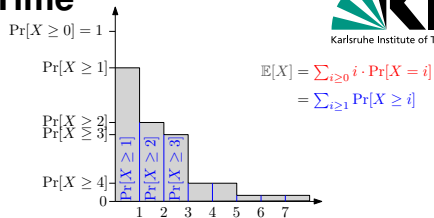
$$\mathbb{E}[\text{running time}] \leq 2p(|w|) \cdot \mathbb{E}[\#\text{rounds}] \leq 2p(n) \cdot \sum_{i \geq 1} \Pr[\#\text{rounds} \geq i] \leq 2p(n) \cdot \sum_{i \geq 1} 2^{-(i-1)} = 2p(n) \cdot \sum_{i \geq 0} 2^{-i} = 4p(n). \quad \square$$

ZPP: Zero-Error-Probabilistic Polynomial Time

Theorem: $L \in \text{ZPP} \Rightarrow$ Las-Vegas Algorithm for L

If $L \in \text{ZPP} := \text{RP} \cap \text{co-RP}$ then there exists a PTM that

- decides L with no error
 - has *expected* polynomial running time
- \hookrightarrow this PTM is not in normal form



Proof

Let T be an **RP**-PTM for L with running time $p(n)$.

\hookrightarrow never errs for $x \notin L$

Let \bar{T} be an **RP**-PTM for \bar{L} with running time $p(n)$.

\hookrightarrow never errs for $x \notin \bar{L}$

- T and \bar{T} never *both* answer incorrectly \Rightarrow we always answer correctly.
- Every round gives $r_1 = r_2$ with probability $\geq 1/2$.



repeat

$r_1 \leftarrow T(w)$

$r_2 \leftarrow \text{not } \bar{T}(w)$

until $r_1 = r_2$

return r_1

$$\mathbb{E}[\text{running time}] \leq 2p(|w|) \cdot \mathbb{E}[\#\text{rounds}] \leq 2p(n) \cdot \sum_{i \geq 1} \Pr[\#\text{rounds} \geq i] \leq 2p(n) \cdot \sum_{i \geq 1} 2^{-(i-1)} = 2p(n) \cdot \sum_{i \geq 0} 2^{-i} = 4p(n). \quad \square$$

Remark

The classes **RP**, **co-RP** and **BPP** are not believed to have complete problems unless, e.g. **BPP = P**.
Underlying issue: “ T is a **BPP**-PTM” is undecidable.

1. Preliminaries

2. Probabilistic Turing Machines

3. Complexity Classes

4. Relationships between Complexity Classes

Preliminaries

○○

Probabilistic Turing Machines

○○

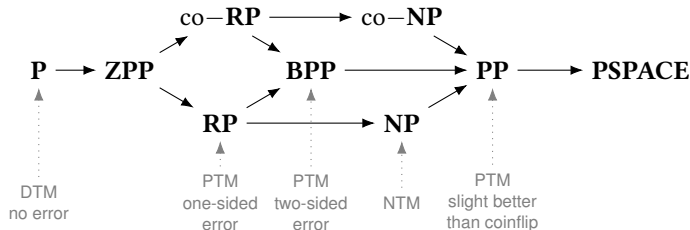
Complexity Classes

○○○○○

Relationships between Complexity Classes

●○○○○○

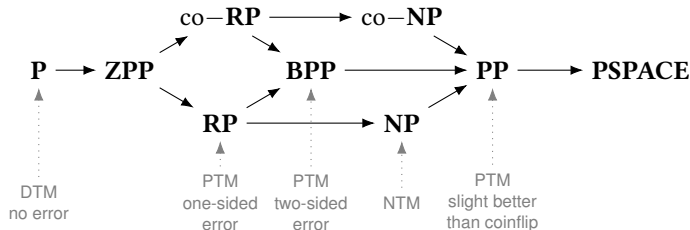
Beziehungen zwischen Komplexitätsklassen



Theorems

- $P \subseteq ZPP$
- $ZPP \subseteq RP$ and $ZPP \subseteq co-RP$
- $RP \subseteq NP$ and $co-RP \subseteq co-NP$
- $RP \subseteq BPP$ and $co-RP \subseteq BPP$
- $BPP \subseteq PP$
- $NP \subseteq PP$ and $co-NP \subseteq PP$
- $PP \subseteq PSPACE$

Beziehungen zwischen Komplexitätsklassen



Theorems

- $P \subseteq ZPP$
- $ZPP \subseteq RP$ and $ZPP \subseteq \text{co-RP}$
- $RP \subseteq NP$ and $\text{co-RP} \subseteq \text{co-NP}$
- $RP \subseteq BPP$ and $\text{co-RP} \subseteq BPP$
- $BPP \subseteq PP$

proved in the following (rest is exercise):

- $NP \subseteq PP$ and $\text{co-NP} \subseteq PP$
- $PP \subseteq PSPACE$

“Typecasting” Turing Machines

DTM as NTM

Given DTM T with transition function δ , consider NTM T' with transition functions $\delta_0 = \delta_1 = \delta$.

\hookrightarrow No change in behaviour: $T(w) = \text{YES} \Leftrightarrow T'(w) = \text{YES}$.

“Typecasting” Turing Machines

DTM as NTM

Given DTM T with transition function δ , consider NTM T' with transition functions $\delta_0 = \delta_1 = \delta$.

\hookrightarrow No change in behaviour: $T(w) = \text{YES} \Leftrightarrow T'(w) = \text{YES}$.

NTM as PTM

Given NTM T , we can reinterpret it as a PTM T' :

$$T(w) = \text{YES} :\Leftrightarrow \exists \text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] > 0$$

$$T(w) = \text{NO} :\Leftrightarrow \nexists \text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] = 0$$

DTM as NTM

Given DTM T with transition function δ , consider NTM T' with transition functions $\delta_0 = \delta_1 = \delta$.
 \hookrightarrow No change in behaviour: $T(w) = \text{YES} \Leftrightarrow T'(w) = \text{YES}$.

NTM as PTM

Given NTM T , we can reinterpret it as a PTM T' :

$$T(w) = \text{YES} :\Leftrightarrow \exists \text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] > 0$$

$$T(w) = \text{NO} :\Leftrightarrow \nexists \text{YES-computation for } T \text{ and } w \Leftrightarrow \Pr[T'(w) = \text{YES}] = 0$$

PTM as DTM

Given PTM T , we can view it as DTM T' with random bitstring $b = b_1 b_2 \dots$ as additional input.
In step i transition function δ_{b_i} is used.

$$\Pr[T(w) = \text{YES}] = \Pr_{b_1, b_2, \dots \sim \text{Ber}(1/2)} [T'(w, b) = \text{YES}].$$

Theorem: $NP \subseteq PP$ (analogously $co-NP \subseteq PP$)

i.e. show that each $L \in NP$ satisfies $L \in PP$

Theorem: $\text{NP} \subseteq \text{PP}$ (analogously $\text{co-NP} \subseteq \text{PP}$)

i.e. show that each $L \in \text{NP}$ satisfies $L \in \text{PP}$

Have: NTM T certifying that $L \in \text{NP}$

$w \in L \Leftrightarrow \exists \text{YES-computation for } T \text{ and } w$

Theorem: $\text{NP} \subseteq \text{PP}$ (analogously $\text{co-NP} \subseteq \text{PP}$)

i.e. show that each $L \in \text{NP}$ satisfies $L \in \text{PP}$

Have: NTM T certifying that $L \in \text{NP}$

$w \in L \Leftrightarrow \exists \text{YES-computation for } T \text{ and } w$

Use the NTM T as a PTM T' :

$\forall w \notin L : \Pr[T'(w) = \text{YES}] = 0$

$\forall w \in L : \Pr[T'(w) = \text{YES}] > 0$



Theorem: $\text{NP} \subseteq \text{PP}$ (analogously $\text{co-NP} \subseteq \text{PP}$)

i.e. show that each $L \in \text{NP}$ satisfies $L \in \text{PP}$

Have: NTM T certifying that $L \in \text{NP}$

$w \in L \Leftrightarrow \exists$ YES-computation for T and w

Use the NTM T as a PTM T' :

$\forall w \notin L : \Pr[T'(w) = \text{YES}] = 0$

$\forall w \in L : \Pr[T'(w) = \text{YES}] > 0$



Want: PTM T'' certifying that $L \in \text{PP}$



$\forall w \notin L : \Pr[T''(w) = \text{YES}] \leq 1/2$

$\forall w \in L : \Pr[T''(w) = \text{YES}] > 1/2$

Theorem: $\text{NP} \subseteq \text{PP}$ (analogously $\text{co-NP} \subseteq \text{PP}$)

i.e. show that each $L \in \text{NP}$ satisfies $L \in \text{PP}$

Have: NTM T certifying that $L \in \text{NP}$

$w \in L \Leftrightarrow \exists$ YES-computation for T and w

Use the NTM T as a PTM T' :

$\forall w \notin L : \Pr[T'(w) = \text{YES}] = 0$

$\forall w \in L : \Pr[T'(w) = \text{YES}] > 0$



Want: PTM T'' certifying that $L \in \text{PP}$



$\forall w \notin L : \Pr[T''(w) = \text{YES}] \leq 1/2$

$\forall w \in L : \Pr[T''(w) = \text{YES}] > 1/2$

T'' achieves this shift with a simple trick

$r \leftarrow T'(w)$ // T' is T as PTM

if $r = \text{YES}$ **then**

return YES

else

 sample $b \sim \mathcal{U}(\{\text{YES}, \text{NO}\})$ // coinflip

return b

Theorem: $PP \subseteq PSPACE$

i.e. show that each $L \in PP$ satisfies $L \in PSPACE$

Proof

- Let T a PP-PTM for L with running time $p(n)$.

Theorem: $PP \subseteq PSPACE$

i.e. show that each $L \in PP$ satisfies $L \in PSPACE$

Proof

- Let T a **PP**-PTM for L with running time $p(n)$.
- Consider DTM T' that simulates T for given w and random choices $b_1 b_2 \dots b_{p(n)}$.
- Consider DTM T'' that for input w runs $T'(w, b_1 b_2 \dots b_{p(n)})$ for all $2^{p(n)}$ possible $b_1 b_2 \dots b_{p(n)}$. Return YES if T' returns YES in majority of cases.

```
n ← |w|
k ← p(n)
a ← 0 // k-bit counter
for  $b_1 \dots b_k \leftarrow 00 \dots 0$  to  $11 \dots 1$  do
  | r ← T'(w,  $b_1 \dots b_k$ )
  | if r = YES then
  | | a ← a + 1
if a >  $2^{k-1}$  then
  | return YES
else
  | return NO
```

Theorem: $PP \subseteq PSPACE$

i.e. show that each $L \in PP$ satisfies $L \in PSPACE$

Proof

- Let T a PP-PTM for L with running time $p(n)$.
- Consider DTM T' that simulates T for given w and random choices $b_1 b_2 \dots b_{p(n)}$.
- Consider DTM T'' that for input w runs $T'(w, b_1 b_2 \dots b_{p(n)})$ for all $2^{p(n)}$ possible $b_1 b_2 \dots b_{p(n)}$. Return YES if T' returns YES in majority of cases.
- space complexity:
 - $p(n)$ bits for counter a
 - $p(n)$ bits for b_1, \dots, b_k
 - $\mathcal{O}(p(n))$ space for simulating T
(can only use $p(n)$ space in its $p(n)$ steps)

```
n ← |w|
k ← p(n)
a ← 0 // k-bit counter
for b1 ... bk ← 00 ... 0 to 11 ... 1 do
  r ← T'(w, b1 ... bk)
  if r = YES then
    a ← a + 1
if a > 2k-1 then
  return YES
else
  return NO
```

Theorem: $PP \subseteq PSPACE$

i.e. show that each $L \in PP$ satisfies $L \in PSPACE$

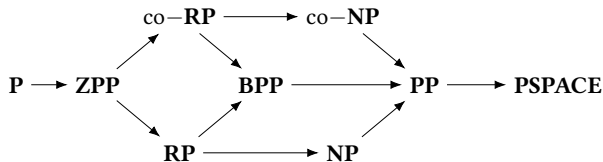
Proof

- Let T a PP-PTM for L with running time $p(n)$.
- Consider DTM T' that simulates T for given w and random choices $b_1 b_2 \dots b_{p(n)}$.
- Consider DTM T'' that for input w runs $T'(w, b_1 b_2 \dots b_{p(n)})$ for all $2^{p(n)}$ possible $b_1 b_2 \dots b_{p(n)}$. Return YES if T' returns YES in majority of cases.
- space complexity:
 - $p(n)$ bits for counter a
 - $p(n)$ bits for b_1, \dots, b_k
 - $\mathcal{O}(p(n))$ space for simulating T
(can only use $p(n)$ space in its $p(n)$ steps)

$\Leftrightarrow T''$ decides L in space $\mathcal{O}(p(n))$ (and time $\Omega(2^{p(n)})$). \square

```
n ← |w|
k ← p(n)
a ← 0 // k-bit counter
for b1 ... bk ← 00 ... 0 to 11 ... 1 do
  r ← T'(w, b1 ... bk)
  if r = YES then
    a ← a + 1
if a > 2k-1 then
  return YES
else
  return NO
```

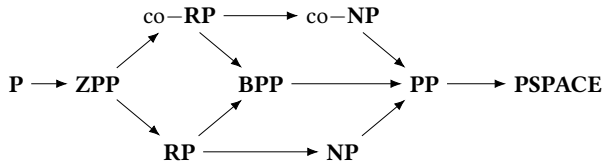
Conclusion



What we learned – not much

- Only “obvious” inclusions known
↳ e.g. one-sided error vs. two-sided error

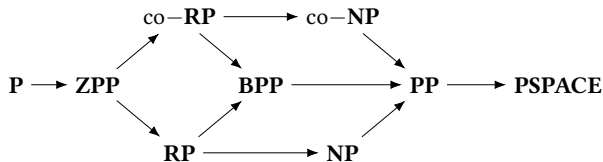
Conclusion



What we learned – not much

- Only “obvious” inclusions known
↪ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.

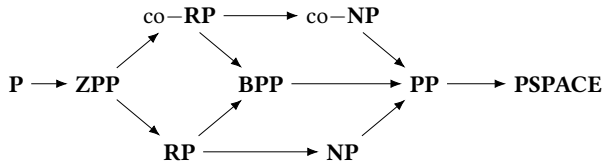
Conclusion



What we learned – not much

- Only “obvious” inclusions known
↳ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:

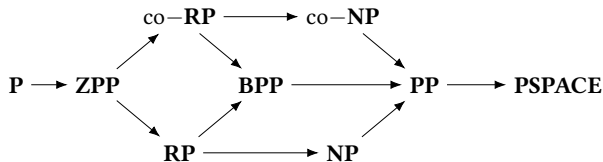
Conclusion



What we learned – not much

- Only “obvious” inclusions known
↪ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
 - obviously: in $co-NP$.

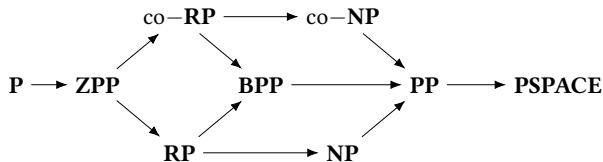
Conclusion



What we learned – not much

- Only “obvious” inclusions known
↪ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
 - obviously: in $co-NP$.
 - 1976: in $co-RP$ (Rabin).

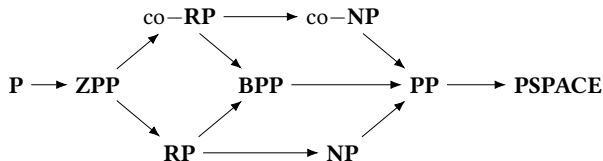
Conclusion



What we learned – not much

- Only “obvious” inclusions known
↪ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
 - obviously: in $co-NP$.
 - 1976: in $co-RP$ (Rabin).
 - 1987: in RP , hence in ZPP (Adleman, Huang).

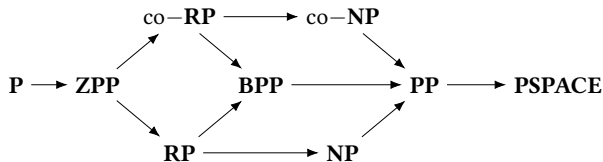
Conclusion



What we learned – not much

- Only “obvious” inclusions known
↔ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
 - obviously: in $co-NP$.
 - 1976: in $co-RP$ (Rabin).
 - 1987: in RP , hence in ZPP (Adleman, Huang).
 - 2002: in P (Agrawal, Kayal, Saxena).

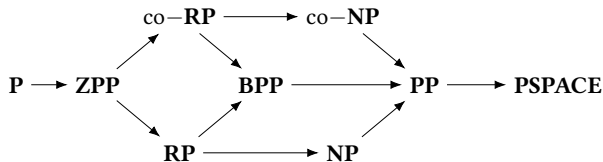
Conclusion



What we learned – not much

- Only “obvious” inclusions known
↔ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
 - obviously: in $co-NP$.
 - 1976: in $co-RP$ (Rabin).
 - 1987: in RP , hence in ZPP (Adleman, Huang).
 - 2002: in P (Agrawal, Kayal, Saxena).

Conclusion



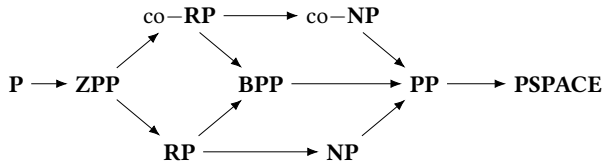
What we learned – not much

- Only “obvious” inclusions known
↪ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
 - obviously: in $co-NP$.
 - 1976: in $co-RP$ (Rabin).
 - 1987: in RP , hence in ZPP (Adleman, Huang).
 - 2002: in P (Agrawal, Kayal, Saxena).

A boring topic?

- People believe $BPP = P$
↪ “each BPP algorithm can be fully derandomised”

Conclusion



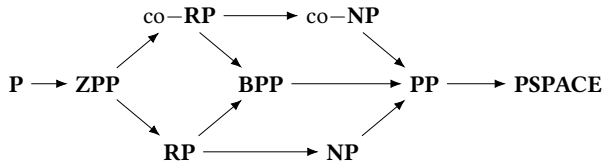
What we learned – not much

- Only “obvious” inclusions known
↪ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
 - obviously: in $co-NP$.
 - 1976: in $co-RP$ (Rabin).
 - 1987: in RP , hence in ZPP (Adleman, Huang).
 - 2002: in P (Agrawal, Kayal, Saxena).

A boring topic?

- People believe $BPP = P$
↪ “each BPP algorithm can be fully derandomised”
- PP is somewhat esoteric
↪ no interesting randomised classes remain?

Conclusion



What we learned – not much

- Only “obvious” inclusions known
↪ e.g. one-sided error vs. two-sided error
- since $P \stackrel{?}{=} PSPACE$ is unsolved, none of the inclusions are known to be strict.
- Remark: History of PRIMES:
 - obviously: in $co-NP$.
 - 1976: in $co-RP$ (Rabin).
 - 1987: in RP , hence in ZPP (Adleman, Huang).
 - 2002: in P (Agrawal, Kayal, Saxena).

A boring topic?

- People believe $BPP = P$
↪ “each BPP algorithm can be fully derandomised”
- PP is somewhat esoteric
↪ no interesting randomised classes remain?
- quantum computing may change the story.
People suspect $NP \not\subseteq BQP \not\subseteq NP$
↪ <https://en.wikipedia.org/wiki/BQP>

- Definiere: Was ist eine PTM? Was ist der Unterschied zu einer NTM?
- Definiere die Komplexitätsklassen **RP**, **co-RP**, **BPP**, **PP**, **ZPP**.
- Inwiefern spielen die Konstanten von $\frac{1}{2}$, $\frac{1}{4}$, $\frac{3}{4}$, die in den Definitionen vorkommen, eine Rolle? Inwiefern sind sie egal?
- Inwiefern steht die Klasse **ZPP** mit dem Konzept eines Las-Vegas Algorithmus in Verbindung? Wie sehen die Umwandlungen in die eine Richtung (Vorlesung) und in die andere Richtung (Übung) aus?
- Welche Inklusionsbeziehungen zwischen diesen Komplexitätsklassen sind bekannt?
- Begründe jede dieser Inklusionsbeziehungen. (In der tatsächlichen Prüfung würde man sich aus Zeitgründen nur eine oder zwei herausgreifen.)
- Gibt es Inklusionsbeziehungen von denen man weiß, dass sie strikt sind? Gibt es Klassen, von denen Experten vermuten, dass sie in Wirklichkeit identisch sind?