Jean Goubault-Larrecq

Randomized complexity classes

Today: **RP**, **coRP**, and **ZPP** (what a zoo!)

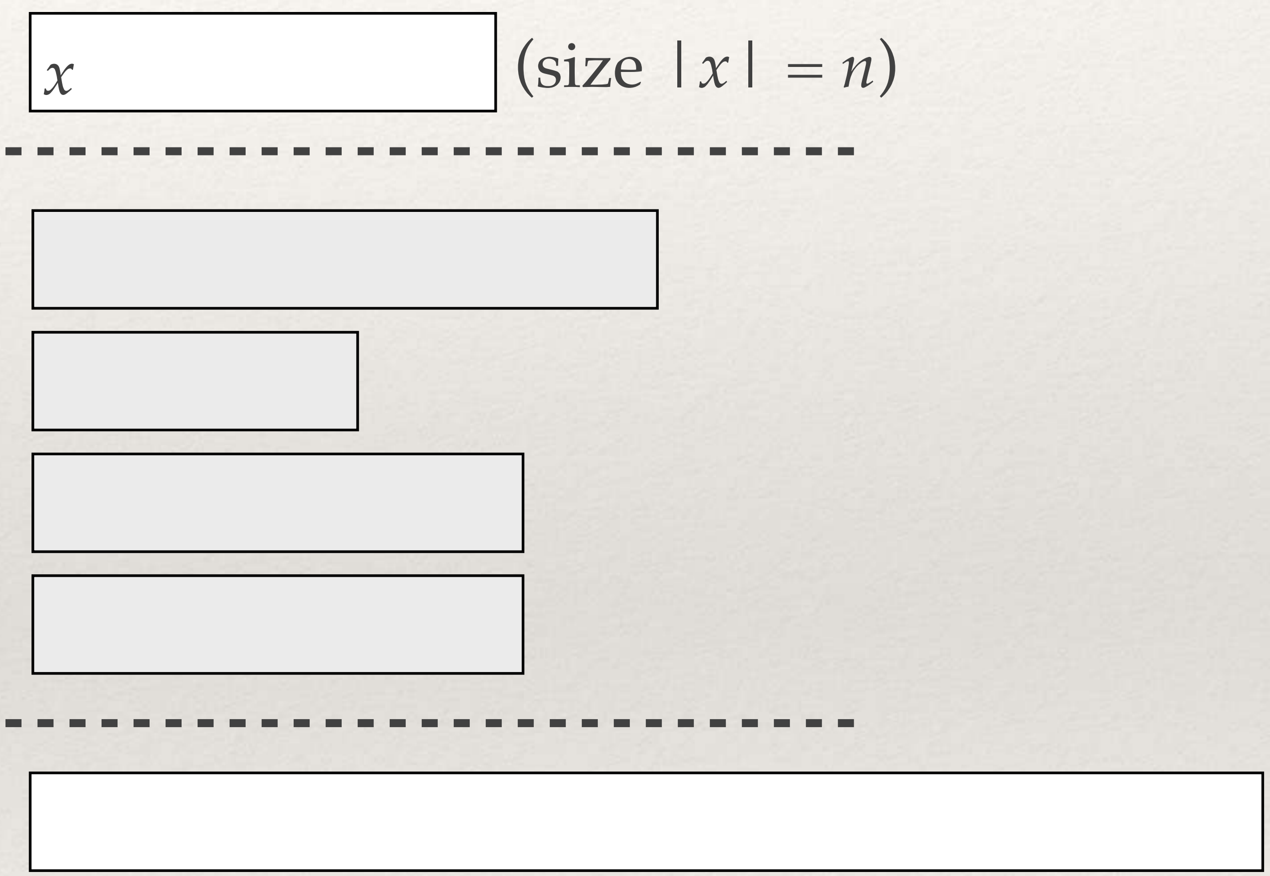
Today

- Randomized Turing machines
- One-sided error: **RP**, **coRP**
- No error: **ZPP**
- Next time: **two**-sided error **BPP**

Randomized Turing machines

Ordinary Turing machines

- One **read-only** input tape



- As many **work tapes** as you need (but only a constant number!)
- (Possibly) one **write-only** output tape

Drawing strings at random

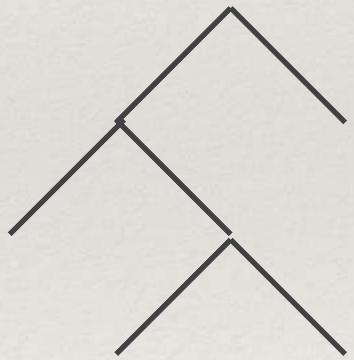
- We will study **probabilistic** complexity classes, where our TMs can now **draw** strings of bits at random

Drawing strings at random

- We will study **probabilistic** complexity classes, where our TMs can now **draw** strings of bits at random
- No need to invent a new TM model

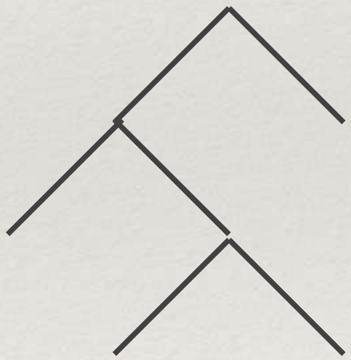
Drawing strings at random

- We will study **probabilistic** complexity classes, where our TMs can now **draw** strings of bits at random
- No need to invent a new TM model
- Choice 1: use a **non-deterministic** TM model
  and draw execution branch at random
  (we won't do that; hard to do it right)



Drawing strings at random

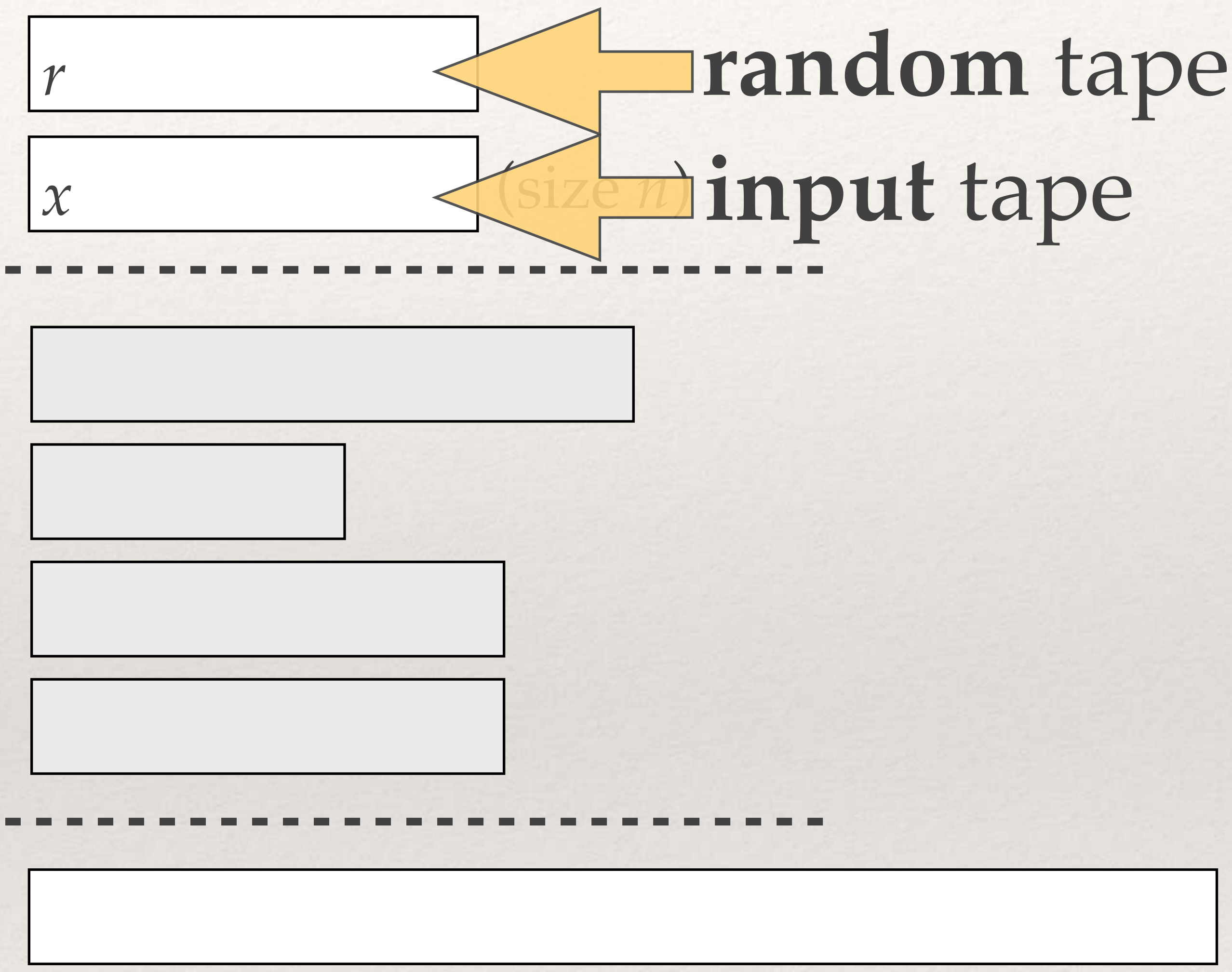
- We will study **probabilistic** complexity classes, where our TMs can now **draw** strings of bits at random
- No need to invent a new TM model
- Choice 1: use a **non-deterministic** TM model
  and draw execution branch at random
  (we won't do that; hard to do it right)



- Choice 2: … next slide

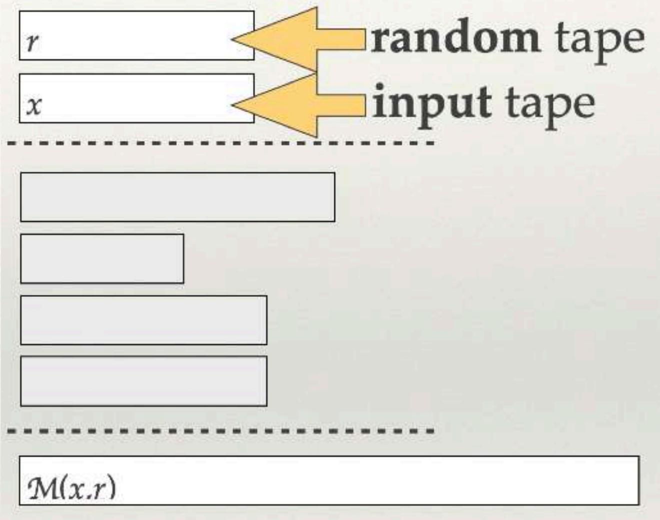
Randomized Turing machines

- Two ~~One~~ **read-only** tapes
- As many **work tapes** as you need (but only a constant number!)
- (Possibly) one **write-only** output tape

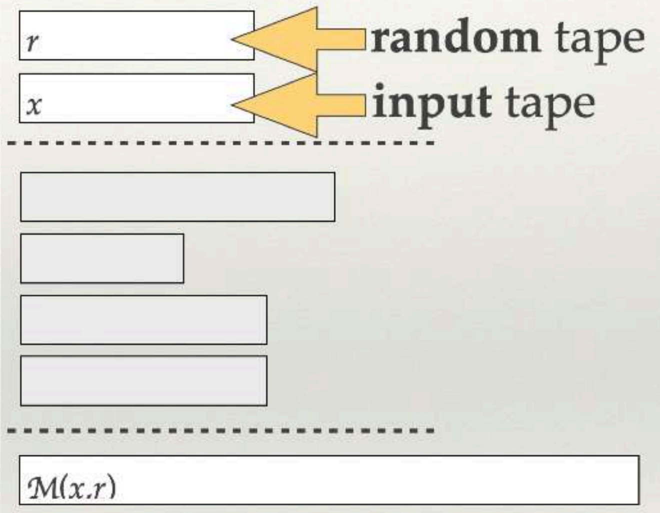


Technical points 1/2

- We draw the random tape r **uniformly at random**



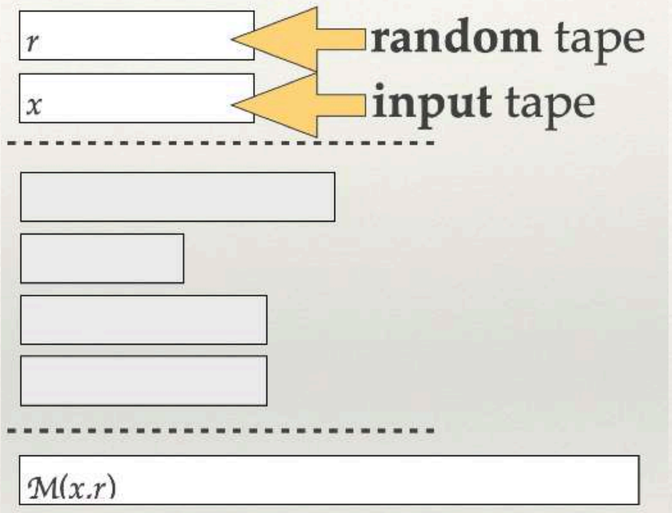
Technical points 1/2



- We draw the random tape r **uniformly at random**
- We will be interested in **probabilities**, e.g.

$$\Pr_r\,[\mathcal{M}(x,r) \text{ accepts}]$$

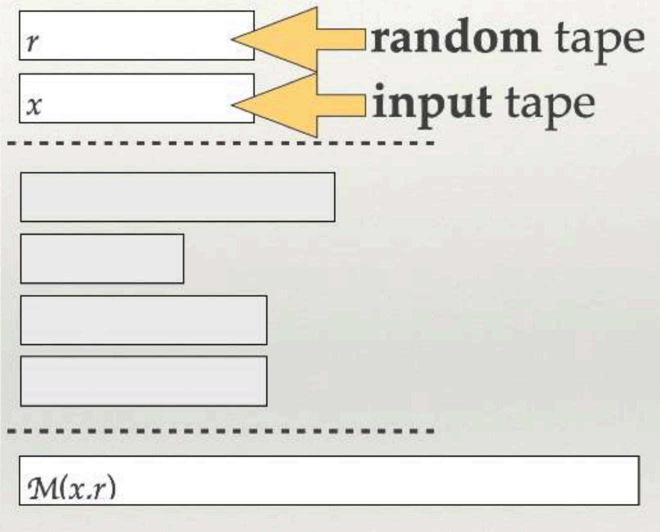
Technical points 1/2



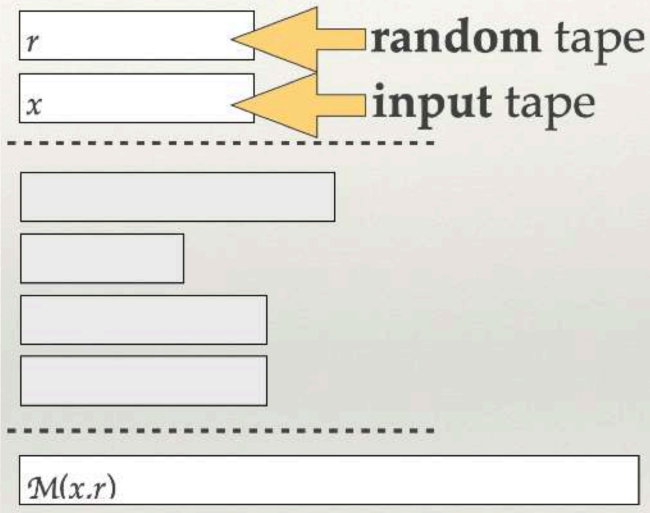
- We draw the random tape r **uniformly at random**
- We will be interested in **probabilities**, e.g.
  $$\Pr_r\,[\mathcal{M}(x,r)\text{ accepts}]$$
- Random tape must not just be read-only: we impose that **no bit on r is ever read twice** (otherwise bits read are not independent)

Technical points 2/2

- $\Rightarrow$ we need r to contain at least $f(n)$ bits, where $f(n)$ is an upper bound on the **time** taken by the TM.

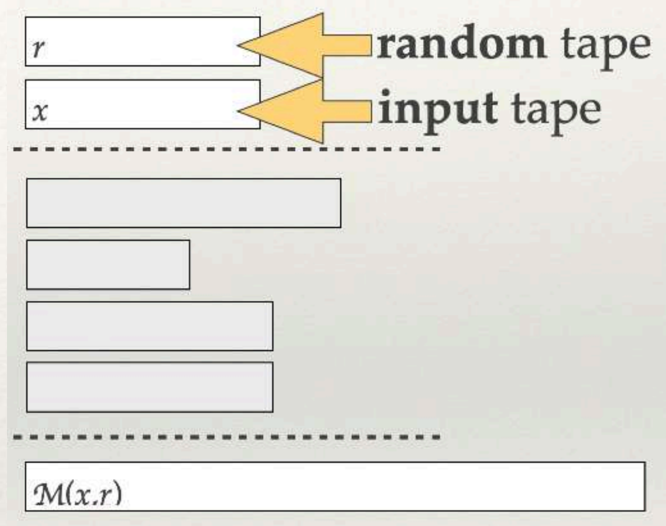


Technical points 2/2



- ⇒ we need r to contain at least $f(n)$ bits, where $f(n)$ is an upper bound on the **time** taken by the TM.
- We will always assume that r is **large enough**

Technical points 2/2



- $\Rightarrow$ we need r to contain at least $f(n)$ bits, where $f(n)$ is an upper bound on the **time** taken by the TM.
- We will always assume that r is **large enough**
- OK for classes defined by **worst-case time**, will cause problems for classes defined with no a priori upper bound on time (e.g., **ZPP**)

Our first probabilistic class: **RP**

(also sometimes known as the class of *Monte Carlo* languages)

http://fr.casino-jackpot.com/wp-content/uploads/2018/04/casino-monaco.jpg

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):

i.e. there is also a **polynomial** $\boldsymbol{p(n)}$ / $\mathcal{M}(x,r)$ terminates in time $\leq p(n)$, where $n=|x|$, in the worst case (and for any value of r)

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):

i.e. there is also a **polynomial $p(n)$** / $\mathcal{M}(x,r)$ terminates in time $\leq p(n)$, where $n=|x|$, in the worst case (and for any value of r)

… hence, implicitly, we require $|r|\geq p(n)$ (let us say $|r| = p(n)$)

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):

- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1/2$

i.e. there is also a **polynomial $p(n)$** / $\mathcal{M}(x,r)$ terminates in time $\leq p(n)$, where $n = |x|$, in the worst case (and for any value of r)

… hence, implicitly, we require $|r| \geq p(n)$ (let us say $|r| = p(n)$)

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1/2$

i.e. there is also a **polynomial** $\boldsymbol{p(n)}$ / $\mathcal{M}(x,r)$ terminates in time $\leq p(n)$, where $n = |x|$, in the worst case (and for any value of r)

… hence, implicitly, we require $|r| \geq p(n)$ (let us say $|r| = p(n)$)

probability taken over all $r \in \{0,1\}^{p(n)}$

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r)$ accepts$] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r [\mathcal{M}(x,r)$ accepts$]=0$).

i.e. there is also a **polynomial $p(n)$** / $\mathcal{M}(x,r)$ terminates in time $\leq p(n)$, where $n=|x|$, in the worst case (and for any value of r)

… hence, implicitly, we require $|r| \geq p(n)$ (let us say $|r| = p(n)$)

probability taken over all $r \in \{0,1\}^{p(n)}$

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

i.e. there is also a **polynomial $p(n)$** / $\mathcal{M}(x,r)$ terminates in time $\leq p(n)$, where $n=|x|$, in the worst case (and for any value of r)

… hence, implicitly, we require $|r| \geq p(n)$ (let us say $|r| = p(n)$)

probability taken over all $r \in \{0,1\}^{p(n)}$

one-sided error: we make **no** error if $x \notin L$

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):

i.e. there is also a **polynomial $p(n)$** / $\mathcal{M}(x,r)$ terminates in time $\leq p(n)$, where $n=|x|$, in the worst case (and for any value of r)

… hence, implicitly, we require $|r| \geq p(n)$ (let us say $|r| = p(n)$)

- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1/2$

probability taken over all $r \in \{0,1\}^{p(n)}$

- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

one-sided error: we make **no** error if $x \notin L$

Perhaps paradoxically, that means that we make **no** error if $\mathcal{M}(x,r)$ **accepts**
(so please do not confuse acceptance with being in the language!)

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r[\mathcal{M}(x,r) \text{ accepts}]=0$).

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):

Note: **RP**-languages are **not** defined by « **RP**-machines » (there is no such notion)

- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

RP: Randomized Polynomial time

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

Note: **RP**-languages are **not** defined by « **RP**-machines » (there is no such notion)

… but if we wanted to define « **RP**-machines », those would be machines $\mathcal{M}$ such that, for every x,
— either $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1/2$
— or $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] = 0$

coRP

- L is in **coRP** iff complement L^c is in **RP**, hence:

coRP

- L is in **coRP** iff complement L^c is in **RP**, hence:
- L is in **coRP** if and only if
  there is a **polynomial-time** TM $\mathcal{M}$
  such that for every input x (of size n):
- if $x \in L$ then $\cancel{\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2}$ $\mathcal{M}(x,r)$ accepts for every r
- if $x \notin L$ then $\cancel{\mathcal{M}(x,r) \text{ accepts for no } r}$ $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \leq 1/2$

A motivating example for (co)RP

- **PRIMALITY**
  INPUT: a natural number p, in binary
  Q: is p prime?

A motivating example for (co)RP

- **PRIMALITY**
  INPUT: a natural number p, in binary
  Q: is p prime?
- For a long time, not known to be in **P**
  (now solved: indeed in **P** [Agrawal,Kayal,Saxena 2004])

A motivating example for (co)RP

- **PRIMALITY**
  INPUT: a natural number p, in binary
  Q: is p prime?
- For a long time, not known to be in **P**
  (now solved: indeed in **P** [Agrawal,Kayal,Saxena 2004])
- In **coNP** (guess a proper divisor)

A motivating example for (**co**)**RP**

- **PRIMALITY**
  INPUT: a natural number p, in binary
  Q: is p prime?
- For a long time, not known to be in **P**
  (now solved: indeed in **P** [Agrawal,Kayal,Saxena 2004])
- In **coNP** (guess a proper divisor)
- In **NP** [Pratt 1975]

A motivating example for (co)RP

- **PRIMALITY**
  INPUT: a natural number *p*, in binary
  Q: is *p* prime?
- For a long time, not known to be in **P**
  (now solved: indeed in **P** [Agrawal,Kayal,Saxena 2004])
- In **coNP** (guess a proper divisor)
- In **NP** [Pratt 1975]
- Can also be solved efficiently with **randomization**…

Fermat's little theorem

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$), $r^{p-1} = 1 \bmod p$.

Fermat's little theorem

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$), $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r at random in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.

Fermat's little theorem

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$),
  $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r at random in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- Note: computing mod p is **efficient**:
  size of all numbers **bounded** by size(p)=O($\log p$).
  — addition mod p in time $O(\log p)$
  — mult. mod p in time $O(\log^2 p)$ (even $O(\log^{1+\varepsilon} p)$)

Fermat's little theorem

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$),
  $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r at random in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- Note: computing mod p is **efficient**:
  size of all numbers **bounded** by size(p)=O(log p).
  — addition mod p in time O(log p)
  — mult. mod p in time O($\log^2 p$) (even O($\log^{1+\varepsilon} p$))
- An experiment… (next slide)

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$), $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r at random in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$), $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r at random in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- Is 87 prime?

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$), $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r at random in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- Is 87 prime?
- Draw r at random… say 25

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$),
  $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r at random in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- Is 87 prime?
- Draw r at random… say 25
- $r^{86} = 16 \bmod 87$

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \leq r < p$),
  $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r at random in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- Is 87 prime?
- Draw r at random… say 25
- $r^{86} = 16 \bmod 87$
- $\Rightarrow$ 87 is **not prime** (definitely)

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \le r < p$), $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- Is 87 prime?

r	r^{86}
2	4
3	9
4	16
5	25
6	36
7	49
8	64
9	81
10	13

r	r^{86}
11	34
12	57
13	82
14	22
15	51
16	82
17	28
18	63
19	13
20	52

r	r^{86}
21	6
22	49
23	7
24	54
25	16
26	67
27	33
28	1
29	58
30	30

r	x^{86}
31	4
32	67
33	45
34	25
35	7
36	78
37	64
38	52
39	42
40	34

r	r^{86}
41	28
42	24
43	22
44	22
45	24
46	28
47	34
48	42
49	52
50	64

r	r^{86}
51	78
52	7
53	25
54	45
55	67
56	4
57	30
58	58
59	1
60	33

r	r^{86}
61	67
62	16
63	54
64	7
65	49
66	6
67	52
68	13
69	63
70	28

r	r^{86}
71	82
72	51
73	22
74	82
75	57
76	34
77	13
78	81
79	64
80	49

r	r^{86}
81	36
82	25
83	16
84	9
85	4

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r $(1 \leq r < p)$,
  $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- Is 87 prime?
- The probability (over r) of error is:
  $$2/84 \approx 0.024$$

r	r^{86}	r	r^{86}	r	r^{86}	r	x^{86}	r	r^{86}	r	r^{86}	r	r^{86}	r	r^{86}	r	r^{86}
		11	34	21	6	31	4	41	28	51	78	61	67	71	82	81	36
2	4	12	57	22	49	32	67	42	24	52	7	62	16	72	51	82	25
3	9	13	82	23	7	33	45	43	22	53	25	63	54	73	22	83	16
4	16	14	22	24	54	34	25	44	22	54	45	64	7	74	82	84	9
5	25	15	51	25	16	35	7	45	24	55	67	65	49	75	57	85	4
6	36	16	82	26	67	36	78	46	28	56	4	66	6	76	34		
7	49	17	28	27	33	37	64	47	34	57	30	67	52	77	13		
8	64	18	63	28	1	38	52	48	42	58	58	68	13	78	81		
9	81	19	13	29	58	39	42	49	52	59	1	69	63	79	64		
10	13	20	52	30	30	40	34	50	64	60	33	70	28	80	49		

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \le r < p$), $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.

L is in **coRP** if and only if
there is a **polynomial-time** TM $\mathcal{M}$
such that for every input x (of size n):

if $x \in L$ then ~~$\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$~~ $\mathcal{M}(x,r)$ accepts for every r

if $x \notin L$ then ~~$\mathcal{M}(x,r)$ accepts for no r~~ $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \leq 1/2$

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \le r < p$), $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- If p is prime, will succeed **for every r**

L is in **coRP** if and only if
there is a **polynomial-time** TM $\mathcal{M}$
such that for every input x (of size n):

if $x \in L$ then ~~$\Pr_r [\mathcal{M}(x,r)$ accepts$] \ge 1/2$~~ $\mathcal{M}(x,r)$ accepts for every r

if $x \notin L$ then ~~$\mathcal{M}(x,r)$ accepts for no r~~ $\Pr_r [\mathcal{M}(x,r)$ accepts$] \le 1/2$

Fermat's little theorem in practice

- **Thm (Fermat).** If p is prime, then for every r ($1 \le r < p$), $r^{p-1} = 1 \bmod p$.
- $\Rightarrow$ draw r in $[2, p-2]$; accept if $r^{p-1} = 1 \bmod p$.
- If p is prime, will succeed **for every r**
- Else, will fail with (hopefully) **high probability** (0.024 in the example, looks good); but…

L is in **coRP** if and only if
there is a **polynomial-time** TM $\mathcal{M}$
such that for every input x (of size n):

if $x \in L$ then $\cancel{\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \ge 1/2}$ $\mathcal{M}(x,r)$ accepts for every r

if $x \notin L$ then $\cancel{\mathcal{M}(x,r) \text{ accepts for no } r}$ $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \le 1/2$

Carmichael numbers

- A **Carmichael number** is a number p:
  — that is **not** prime
  — but passes all Fermat tests ($r^{p-1}=1 \bmod p$ for every r)
- I.e., on which our hopes of low error rate fail miserably

Carmichael numbers

- A **Carmichael number** is a number p:
— that is **not** prime
— but passes all Fermat tests (r^{p-1}=1 mod p for every r)
- I.e., on which our hopes of low error rate fail miserably
- Infinitely many of them [Alford, Granville, Pomerance 1994]:
561, 1105, 1729, 2465, 2821, 6601 , 8911, 10585, 15841, etc.

Carmichael numbers

- A **Carmichael number** is a number p:
  — that is **not** prime
  — but passes all Fermat tests ($r^{p-1}=1 \bmod p$ for every r)
- I.e., on which our hopes of low error rate fail miserably
- Infinitely many of them [Alford, Granville, Pomerance 1994]:
  561, 1105, 1729, 2465, 2821, 6601 , 8911, 10585, 15841, etc.
- Frustrating: if p is not prime and passes at least **one** Fermat test, then it passes at least **half** of them…

The Miller-Rabin test (1/2)

- We use another basic fact: if p is prime, then
  the only square roots of 1 mod p are 1 and -1

The Miller-Rabin test (1/2)

- We use another basic fact: if p is prime, then
  the only square roots of 1 mod p are 1 and -1
- Hence, if p is prime and odd (so $p-1 = 2^k q$, q odd):

(read from right to left : ←)

r^q	…	r^($2^{i-1} q$)	r^($2^i q$)	…	r^($2^{k-1} q$)	r^($2^k q$)	mod p
(don't care …	…. don't care)	-1	1	…	1	1	for some i, or:

The Miller-Rabin test (1/2)

- We use another basic fact: if p is prime, then
  the only square roots of 1 mod p are 1 and –1
- Hence, if p is prime and odd (so p–1 = $2^k q$, q odd):

(read from right to left : ←)

r^q	…	r^($2^{i-1}q$)	r^($2^i q$)	…	r^($2^{k-1}q$)	r^($2^k q$)	mod p
(don't care …	…. don't care)	–1	1	…	1	1	for some i, or:

r^q	…	r^($2^{i-1}q$)	r^($2^i q$)	…	r^($2^{k-1}q$)	r^($2^k q$)	mod p
1	…	1	1	…	1	1	

The Miller-Rabin test (2/2)

- On input p,
  draw r at random:
  — if the test shown here:
  succeeds, then **accept** (p probably prime)
  — otherwise **reject** (p definitely not prime)

Hence, if p is prime and odd (so $p–1 = 2^k\, q$, q odd):

(read from right to left : ←)

r^q	…	r^($2^{i–1}\,q$)	r^($2^i\,q$)	…	r^($2^{k–1}\,q$)	r^($2^k\,q$)	mod p
(don't care …	…. don't care)	–1	1	…	1	1	for some i, or:

r^q	…	r^($2^{i–1}\,q$)	r^($2^i\,q$)	…	r^($2^{k–1}\,q$)	r^($2^k\,q$)	mod p
1	…	1	1	…	1	1	

The Miller-Rabin test (2/2)

Hence, if p is prime and odd (so $p-1 = 2^k q$, q odd):

(read from right to left : ←)

r^q	…	r^($2^{i-1}q$)	r^($2^i q$)	…	r^($2^{k-1}q$)	r^($2^k q$)	mod p
(don't care …	…. don't care)	–1	1	…	1	1	for some i, or:

r^q	…	r^($2^{i-1}q$)	r^($2^i q$)	…	r^($2^{k-1}q$)	r^($2^k q$)	mod p
1	…	1	1	…	1	1	

- On input p,
  draw r at random:
  — if the test shown here:
  succeeds, then **accept** (p probably prime)
  — otherwise **reject** (p definitely not prime)
- Probability of error ≤ 1/4. Excellent! Hence:

The Miller-Rabin test (2/2)

- On input p,
  draw r at random:
  — if the test shown here:

  Hence, if p is prime and odd (so $p-1 = 2^k q$, q odd):

  (read from right to left : ←)

r^q	…	r^($2^{i-1} q$)	r^($2^i q$)	…	r^($2^{k-1} q$)	r^($2^k q$)	mod p
(don't care …	…. don't care)	−1	1	…	1	1	for some i, or:

r^q	…	r^($2^{i-1} q$)	r^($2^i q$)	…	r^($2^{k-1} q$)	r^($2^k q$)	mod p
1	…	1	1	…	1	1	

  succeeds, then **accept** (p probably prime)
  — otherwise **reject** (p definitely not prime)
- Probability of error ≤ 1/4. Excellent! Hence:
- **Theorem. PRIMALITY** is in **coRP**.

The Miller-Rabin test (2/2)

Hence, if p is prime and odd (so $p-1 = 2^k q$, q odd):

(read from right to left : ←)

r^q	…	r^$(2^{i-1} q)$	r^$(2^i q)$	…	r^$(2^{k-1} q)$	r^$(2^k q)$	mod p
(don't care …	…. don't care)	-1	1	…	1	1	for some i, or:

r^q	…	r^$(2^{i-1} q)$	r^$(2^i q)$	…	r^$(2^{k-1} q)$	r^$(2^k q)$	mod p
1	…	1	1	…	1	1	

- On input p,
  draw r at random:
  — if the test shown here:
  succeeds, then **accept** (p probably prime)
  — otherwise **reject** (p definitely not prime)
- Probability of error $\leq 1/4$. Excellent! Hence:
- **Theorem. PRIMALITY** is in **coRP**.
- (Superseded by [AKS04]…
  but Miller-Rabin works in log space, not [AKS04]!)

To know more

Notes on Primality Testing
And Public Key Cryptography
Part 1: Randomized Algorithms
Miller–Rabin and Solovay–Strassen Tests

Jean Gallier and Jocelyn Quaintance
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, USA
e-mail: jean@cis.upenn.edu

February 27, 2019

https://www.cis.upenn.edu/~jean/RSA-primality-testing.pdf

Error reduction

- What is so special about error $1/2$?

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error $= 1 - 1/2$
($= 1/2$ here)

Error reduction

- What is so special about error 1/2?
- Nothing!

error = 1 – 1/2
(= 1/2 here)

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

Error reduction

- What is so special about error 1/2?
- Nothing!

error = 1 – 1/2
(= 1/2 here)

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

Error reduction

- What is so special about error 1/2?
- Nothing!
- **Theorem.** $\forall\ \varepsilon \in\ ]0, 1[$, **RP** = **RP**($\varepsilon$).

error = 1 – 1/2 (= 1/2 here)

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

Error reduction

- What is so special about error 1/2?
- Nothing!
- **Theorem.** $\forall\ \varepsilon \in\ ]0, 1[$, **RP** = **RP**($\varepsilon$).
- Note: **RP**=**RP**(1/2) (def.)

error = 1 – 1/2 (= 1/2 here)

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
  - if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
  - if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
  - if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1–\varepsilon$
  - if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

Error reduction: the easy direction

- Clearly, if $\eta \leq \varepsilon$ then
  $\mathbf{RP}(\eta) \subseteq \mathbf{RP}(\varepsilon)$

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r[\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

Error reduction: the easy direction

- Clearly, if $\eta \leq \varepsilon$ then
  $\mathbf{RP}(\eta) \subseteq \mathbf{RP}(\varepsilon)$

- Proof: take any $L \in \mathbf{RP}(\eta)$
  … I'll let you finish the argument

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

Error reduction: the easy direction

- Clearly, if $\eta \le \varepsilon$ then

  $\mathbf{RP}(\eta) \subseteq \mathbf{RP}(\varepsilon)$

- Proof: take any $L \in \mathbf{RP}(\eta)$

  … I'll let you finish the argument

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r)$ accepts$] \ge 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r [\mathcal{M}(x,r)$ accepts$]=0)$.

error = ε

- Note: $\mathbf{RP}(0)=\mathbf{P}$ (believed $\neq$ $\mathbf{RP}$)

  $\mathbf{RP}(1)=\{$*all languages*$\}$ (why?)

The hard direction: repeating experiments

- Let $L \in \mathbf{RP}(\varepsilon)$, $0<\eta<\varepsilon<1$
- On input x, let us do the following (at most) K times:

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

The hard direction: repeating experiments

- Let $L \in \mathbf{RP}(\varepsilon)$, $0<\eta<\varepsilon<1$
- On input x, let us do the following (at most) K times:
- Draw r at random, simulate $\mathcal{M}(x, r)$ and:

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

The hard direction: repeating experiments

- Let $L \in \mathbf{RP}(\varepsilon)$, $0<\eta<\varepsilon<1$
- On input x, let us do the following (at most) K times:
- Draw r at random, simulate $\mathcal{M}(x, r)$ and:
  - If $\mathcal{M}(x, r)$ accepts, then exit the loop and **accept**;

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r[\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

Remember: if $\mathcal{M}(x, r)$ accepts, then x **must** be in L.

The hard direction: repeating experiments

- Let $L \in \mathbf{RP}(\varepsilon)$, $0<\eta<\varepsilon<1$
- On input x, let us do the following (at most) K times:
- Draw r at random, simulate $\mathcal{M}(x, r)$ and:
  - If $\mathcal{M}(x, r)$ accepts, then exit the loop and **accept**;
  - Otherwise, proceed and loop.

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r[\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

Remember: if $\mathcal{M}(x, r)$ accepts, then x **must** be in L.

The hard direction: repeating experiments

- Let $L \in \mathbf{RP}(\varepsilon)$, $0<\eta<\varepsilon<1$
- On input x, let us do the following (at most) K times:
- Draw r at random, simulate $\mathcal{M}(x, r)$ and:
  - If $\mathcal{M}(x, r)$ accepts, then exit the loop and **accept**;
  - Otherwise, proceed and loop.
- At the end of the loop, **reject**.

> - A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
> - if $x \in L$ then $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
> - if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r[\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

Remember: if $\mathcal{M}(x, r)$ accepts, then x **must** be in L.

Repeating experiments (pretty) formally

- We have defined a **new** randomized TM

  $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:

- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Remember: if $\mathcal{M}(x, r[i])$ accepts, then x **must** be in L.

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

Acceptance: 1. if $x \in L$

- If $x \in L$ (recall L in **RP**(ε)), then letting $r=r[1]\#\ldots\#r[K]$, $Pr_r(\mathcal{M}'(x, r)$ rejects)

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Acceptance: 1. if $x \in L$

- If $x \in L$ (recall L in **RP**(ε)), then letting r=r[1]#…#r[K], $\Pr_r(\mathcal{M}'(x, r)$ rejects)
  - = $\Pr_r(\forall i$=1..K, $\mathcal{M}(x, r[i])$ rejects)

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r[\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Acceptance: 1. if $x \in L$

- If $x \in L$ (recall L in **RP**(ε)), then letting $r{=}r[1]\#\ldots\#r[K]$, $\Pr_r(\mathcal{M}'(x, r)$ rejects)
  - $= \Pr_r(\forall i{=}1..K, \mathcal{M}(x, r[i])$ rejects)
  - $= \Pi_{i=1..K} \Pr_{r[i]}(\mathcal{M}(x, r[i])$ rejects) **(independence)**

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r)$ accepts$] \geq 1{-}\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r[\mathcal{M}(x,r)$ accepts$]{=}0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` $i{=}1$ `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Acceptance: 1. if $x \in L$

- If $x \in L$ (recall L in **RP**(ε)), then letting r=$r[1]\#\ldots\#r[K]$, $\Pr_r(\mathcal{M}'(x, r)$ rejects)
  - $= \Pr_r(\forall i$=1..K, $\mathcal{M}(x, r[i])$ rejects)
  - $= \Pi_{i=1..K} \Pr_{r[i]}(\mathcal{M}(x, r[i])$ rejects) **(independence)**
  - $\leq \varepsilon^K$

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r)$ accepts] $\geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r)$ accepts]=0).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Acceptance: 1. if $x \in L$

- If $x \in L$ (recall L in **RP**(ε)), then letting $r=r[1]\#\ldots\#r[K]$, $\Pr_r(\mathcal{M}'(x, r)$ rejects)
  - $= \Pr_r(\forall i=1..K, \mathcal{M}(x, r[i])$ rejects)
  - $= \Pi_{i=1..K} \Pr_{r[i]}(\mathcal{M}(x, r[i])$ rejects) **(independence)**
  - $\leq \varepsilon^K$
- $\Rightarrow$ If $x \in L$ then $\Pr_r(\mathcal{M}'(x, r)$ accepts) $\geq 1-\varepsilon^K$

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Acceptance: 2. if $x \notin L$; Complexity

- If $x \in L$ (recall L in **RP**(ε)) then $\Pr_r(\mathcal{M}'(x, r)\text{ accepts}) \geq 1-\varepsilon^K$

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r)\text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r[\mathcal{M}(x,r)\text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Acceptance: 2. if $x \notin L$; Complexity

- If $x \in L$ (recall L in **RP**(ε)) then $Pr_r(\mathcal{M}'(x, r)$ accepts$) \geq 1-\varepsilon^K$
- If $x \notin L$, then $\mathcal{M}'(x, r)$ accepts for no r

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $Pr_r [\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $Pr_r [\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Acceptance: 2. if $x \notin L$; Complexity

- If $x \in L$ (recall L in **RP**(ε)) then $\Pr_r(\mathcal{M}'(x, r) \text{ accepts}) \geq 1-\varepsilon^K$
- If $x \notin L$, then $\mathcal{M}'(x, r)$ accepts for no r
- If $\mathcal{M}$ runs in time $p(n)$, then $\mathcal{M}'$ runs in time $O(Kp(n))$

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Acceptance: 2. if $x \notin L$; Complexity

- If $x \in L$ (recall L in **RP**(ε)) then $\Pr_r(\mathcal{M}'(x, r) \text{ accepts}) \geq 1-\varepsilon^K$
- If $x \notin L$, then $\mathcal{M}'(x, r)$ accepts for no r
- If $\mathcal{M}$ runs in time $p(n)$, then $\mathcal{M}'$ runs in time $O(Kp(n))$
- Hence L is in **RP**(ε^K)

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

The hard direction: the end

- We have shown that every language L in **RP**(ε)

  is in **RP**(ε^K)

  (for any $\varepsilon \in [0,1]$, $K \geq 1$)

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r [\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

The hard direction: the end

- We have shown that every language L in **RP**(ε)
  is in **RP**(ε^K)
  (for any $\varepsilon \in [0,1]$, $K \geq 1$)
- If $0<\eta<\varepsilon<1$,
  choose K **large enough**
  so that $\varepsilon^K \leq \eta$
  (explicitly, $K \geq \eta / \log \varepsilon$)

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

The hard direction: the end

- We have shown that every language L in **RP**(ε) is in **RP**(ε^K) (for any $\varepsilon \in [0,1]$, $K \geq 1$)
- If $0<\eta<\varepsilon<1$, choose K **large enough** so that $\varepsilon^K \leq \eta$ (explicitly, $K \geq \eta / \log \varepsilon$)
- Then L is in **RP**(η). □

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

Can we do even better?

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

Can we do even better?

- Hence we define the same class with error $\varepsilon = 0.000000000001$

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r[\mathcal{M}(x,r)$ accepts$]=0$).

error $= \varepsilon$

Can we do even better?

- Hence we define the same class with error $\varepsilon = 0.000000000001$
- … or with error $\varepsilon = 0.99999999$!

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r[\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

Can we do even better?

- Hence we define the same class with error $\varepsilon = 0.000000000001$
- … or with error $\varepsilon = 0.99999999$!
- Can we make ε go to 0 as $n \to \infty$?

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

The hard direction **revisited**

- If $x \in L$ (recall L in **RP**(ε)) then $\Pr_r(\mathcal{M}'(x, r) \text{ accepts}) \geq 1-\varepsilon^K$
- If $x \notin L$, then $\mathcal{M}'(x, r)$ accepts for no r
- If $\mathcal{M}$ runs in time $p(n)$, then $\mathcal{M}'$ runs in time $O(Kp(n))$
- Hence L is in **RP**(ε^K).

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

The hard direction **revisited**

Let us take K = a **polynomial** $q(n)$

- If $x \in L$ (recall L in **RP**(ε)) then $\Pr_r(\mathcal{M}'(x, r) \text{ accepts}) \geq 1-\varepsilon^K$
- If $x \notin L$, then $\mathcal{M}'(x, r)$ accepts for no r
- If $\mathcal{M}$ runs in time $p(n)$, then $\mathcal{M}'$ runs in time $O(Kp(n))$
- Hence L is in **RP**(ε^K).

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
  - if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
  - if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

The hard direction **revisited**

Let us take K = a **polynomial** $q(n)$

- If $x \in L$ (recall L in **RP**(ε)) then $\Pr_r(\mathcal{M}'(x, r) \text{ accepts}) \geq 1-\varepsilon^K$
- If $x \notin L$, then $\mathcal{M}'(x, r)$ accepts for no r
- If $\mathcal{M}$ runs in time $p(n)$, then $\mathcal{M}'$ runs in time $O(Kp(n))$
- Hence L is in **RP**(ε^K).

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

=$O(q(n)p(n))$, still **polynomial time**

The hard direction **revisited**

Let us take K = a **polynomial** $q(n)$

- If $x \in L$ (recall L in $\mathbf{RP}(\varepsilon)$) then $\Pr_r(\mathcal{M}'(x, r) \text{ accepts}) \geq 1-\varepsilon^K$
- If $x \notin L$, then $\mathcal{M}'(x, r)$ accepts for no r
- If $\mathcal{M}$ runs in time $p(n)$, then $\mathcal{M}'$ runs in time $O(Kp(n))$
- Hence L is in $\mathbf{RP}(\varepsilon^K)$.

error $\varepsilon^K = \varepsilon^{q(n)}$
(exponentially small)

$=O(q(n)p(n))$, still **polynomial time**

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Define $\mathcal{M}'(x, r[1]\#\ldots\#r[K])$ by:
- `for` i=1 `to` K:
  - If $\mathcal{M}(x, r[i])$ accepts, then exit the loop and **accept**;
- **reject**.

The hard direction **revisited**

- Let ε=1/2. We have proved:
- **Theorem. RP=RP**$(1/2^{q(n)})$
  for every polynomial $q(n)$.
- I.e., error can be made
  exponentially small.
- (Note: **RP**(ε) called $\cup_{p(n)}$ **RTIME**$(p(n),p(n),0,\varepsilon)$
  in the notes: ignore the complication)

- A language L is in **RP**(ε) and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1-\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

error = ε

The hard direction **revisited**

- A language L is in $\mathbf{RP}(\varepsilon)$ and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\, [\mathcal{M}(x,r) \text{ accepts}] \geq 1\!-\!\varepsilon$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
  (i.e., $\Pr_r\, [\mathcal{M}(x,r) \text{ accepts}]=0$).

error = ε

- Let ε=1/2. We have proved:
- **Theorem.** $\mathbf{RP}=\mathbf{RP}(1/2^{q(n)})$
  for every polynomial $q(n)$.
- I.e., error can be made exponentially small.
- (Note: $\mathbf{RP}(\varepsilon)$ called $\cup_{p(n)}$ $\mathbf{RTIME}(p(n),p(n),0,\varepsilon)$ in the notes: ignore the complication)

- Exercise: show that, conversely:
- **Theorem.** $\mathbf{RP}=\mathbf{RP}(1-1/q(n))$
  for every polynomial $q(n)$.
- I.e., error can be assumed « polynomially large » as well

Relation to ordinary classes

- **Theorem. P $\subseteq$ RP $\subseteq$ NP.**
- *Proof.* First,

  $$\mathbf{P} = \mathbf{RP}(0) \subseteq \mathbf{RP}(1/2) = \mathbf{RP}$$

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)\text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r\,[\mathcal{M}(x,r)\text{ accepts}]=0$).

Relation to ordinary classes

- **Theorem. P $\subseteq$ RP $\subseteq$ NP.**
- *Proof.* First,

  $$\mathbf{P} = \mathbf{RP}(0) \subseteq \mathbf{RP}(1/2) = \mathbf{RP}$$

- Second, let $L \in \mathbf{RP}$.

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r

  (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

Relation to ordinary classes

- **Theorem.** $\mathbf{P} \subseteq \mathbf{RP} \subseteq \mathbf{NP}$.
- *Proof.* First,

  $\mathbf{P} = \mathbf{RP}(0) \subseteq \mathbf{RP}(1/2) = \mathbf{RP}$

- Second, let $L \in \mathbf{RP}$.
  - If $x \in L \Rightarrow$ for some r, $\mathcal{M}(x, r)$ accepts (in fact, for at least half of them!)

> - A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
> - if $x \in L$ then $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
> - if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r[\mathcal{M}(x,r) \text{ accepts}]=0$).

Relation to ordinary classes

- **Theorem. P $\subseteq$ RP $\subseteq$ NP.**
- *Proof.* First,

  $\mathbf{P} = \mathbf{RP}(0) \subseteq \mathbf{RP}(1/2) = \mathbf{RP}$
- Second, let $L \in \mathbf{RP}$.
  - If $x \in L \Rightarrow$ for some r, (in fact, for at least half of them!)

    $\mathcal{M}(x, r)$ accepts
  - If $x \notin L \Rightarrow$ for no r.

> - A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
> - if $x \in L$ then $\Pr_r\,[\mathcal{M}(x,r)$ accepts$] \geq 1/2$
> - if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r
>
>   (i.e., $\Pr_r\,[\mathcal{M}(x,r)$ accepts$]=0$).

Relation to ordinary classes

- **Theorem. P $\subseteq$ RP $\subseteq$ NP.**
- *Proof.* First,

  $$\mathbf{P}=\mathbf{RP}(0) \subseteq \mathbf{RP}(1/2) = \mathbf{RP}$$

- Second, let $L \in \mathbf{RP}$.
  - If $x \in L \Rightarrow$ for some r, $\mathcal{M}(x, r)$ accepts (in fact, for at least half of them!)
  - If $x \notin L \Rightarrow$ for no r.
- Hence $L = \{x \mid \exists r, \mathcal{M}(x, r) \text{ accepts}\}$ is in **NP**. $\square$

- A language L is in **RP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r [\mathcal{M}(x,r) \text{ accepts}] \geq 1/2$
- if $x \notin L$ then $\mathcal{M}(x,r)$ accepts for no r (i.e., $\Pr_r [\mathcal{M}(x,r) \text{ accepts}]=0$).

Our second probabilistic class: **ZPP**

(also known as the class of
Las Vegas languages)

https://www.agoda.com/fr-fr/paris-las-vegas_8/hotel/las-vegas-nv-us.html?cid=1844104

ZPP

- **ZPP** = Zero Probability of error Polynomial-time

ZPP

- **ZPP** = Zero Probability of error Polynomial-time
- Usually defined as the class of languages L
  which we can decide in **average** polynomial-time
  (not worst-case!)
  with probability **zero** of making a mistake.

ZPP

- **ZPP** = Zero Probability of error Polynomial-time
- Usually defined as the class of languages L
  which we can decide in **average** polynomial-time
  (not worst-case!)
  with probability **zero** of making a mistake.
- Alternate definition:

  $$\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$$

ZPP

- **ZPP** = Zero Probability of error Polynomial-time
- Usually defined as the class of languages L
  which we can decide in **average** polynomial-time
  (not worst-case!)
  with probability **zero** of making a mistake.
- Alternate definition:

  $$\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$$

- Not clear that those two definitions are equivalent, right?

ZPP

- Let us start simple:

ZPP

- Let us start simple:
- **Definition.** $\mathbf{ZPP} = \mathbf{RP} \cap \mathbf{coRP}$

ZPP

- Let us start simple:
- **Definition. ZPP = RP ∩ coRP**
- I.e., L is in **ZPP** iff there are
  <u>two</u> poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:
  - if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
    $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
  - if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$
    $\mathcal{M}_2(x,r)$ rejects for every r [no error]

ZPP

- Let us start simple:
- **Definition. ZPP = RP ∩ coRP**

an **RP** machine for L

- I.e., L is in **ZPP** iff there are

  <u>two</u> poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

  - if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]

    $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
  - if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$

    $\mathcal{M}_2(x,r)$ rejects for every r [no error]

ZPP

- Let us start simple:
- **Definition. ZPP** = **RP** ∩ **coRP**
- I.e., L is in **ZPP** iff there are

  two poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

  - if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]

    $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$

  - if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$

    $\mathcal{M}_2(x,r)$ rejects for every r [no error]

a **coRP** machine for L

an **RP** machine for L

ZPP, alternate form

- Let us define **ZPP′** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.
- I claim that **ZPP** = **ZPP′**.

ZPP, alternate form

- Let us define **ZPP′** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.
- I claim that **ZPP** = **ZPP′**.
- The definition of **ZPP′** has a few technical problems…
  (see next slides)

ZPP, alternate form

- Let us define **ZPP′** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.
- I claim that **ZPP** = **ZPP′**.
- The definition of **ZPP′** has a few technical problems… (see next slides)
- we will need something called **Markov's inequality** too

ZPP, alternate form

- Let us define **ZPP′** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.
- I claim that **ZPP** = **ZPP′**.
- The definition of **ZPP′** has a few technical problems… (see next slides)
- we will need something called **Markov's inequality** too
- … but before that, we explain why (intuitively) **ZPP** $\subseteq$ **ZPP′**.

Deciding L in **ZPP = RP ∩ coRP** with no error

- Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ such as here→
- Now run the following on input x:

  `forever`:

  `if` $\mathcal{M}_1(x,\ldots)$ rejects: stop and **reject**

  `if` $\mathcal{M}_2(x,\ldots)$ accepts: stop and **accept**

I.e., L is in **ZPP** iff there are
__two__ poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

- if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
  $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
- if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$
  $\mathcal{M}_2(x,r)$ rejects for every r [no error]

Deciding L in **ZPP = RP ∩ coRP** with no error

I.e., L is in **ZPP** iff there are
two poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

- if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
  $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
- if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$
  $\mathcal{M}_2(x,r)$ rejects for every r [no error]

- Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ such as here→
- Now run the following on input x:

`forever`:
  `if` $\mathcal{M}_1(x,\ldots)$ rejects: stop and **reject**
  `if` $\mathcal{M}_2(x,\ldots)$ accepts: stop and **accept**

then x cannot be in L (sure)

Deciding L in **ZPP = RP ∩ coRP** with no error

- Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ such as here→
- Now run the following on input x:

`forever`:

  `if` $\mathcal{M}_1(x,\ldots)$ rejects: stop and **reject**

  `if` $\mathcal{M}_2(x,\ldots)$ accepts: stop and **accept**

I.e., L is in **ZPP** iff there are
<u>**two**</u> poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

- if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
  $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
- if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$
  $\mathcal{M}_2(x,r)$ rejects for every r [no error]

then x cannot be in L (sure)

then x must be in L (sure)

Deciding L in ZPP = RP ∩ coRP with no error

I.e., L is in **ZPP** iff there are
<u>**two**</u> poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

- if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
  $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
- if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$
  $\mathcal{M}_2(x,r)$ rejects for every r [no error]

- Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ such as here→
- Now run the following on input x:

`forever:`
  `if` $\mathcal{M}_1(x,\ldots)$ rejects: stop and **reject**
  `if` $\mathcal{M}_2(x,\ldots)$ accepts: stop and **accept**

then x cannot be in L (sure)

then x must be in L (sure)

Hence this machine **never makes any mistake**

Deciding L in **ZPP** = **RP** ∩ **coRP** with no error

I.e., L is in **ZPP** iff there are
two poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

- if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
  $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
- if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$
  $\mathcal{M}_2(x,r)$ rejects for every r [no error]

- Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ such as here→
- Now run the following on input x:

`forever:`
    `if` $\mathcal{M}_1(x,\ldots)$ rejects: stop and **reject**
    `if` $\mathcal{M}_2(x,\ldots)$ accepts: stop and **accept**

then x cannot be in L (sure)

then x must be in L (sure)

Hence this machine **never makes any mistake**

It may be that $\mathcal{M}_1(x,\ldots)$ accepted and $\mathcal{M}_2(x,\ldots)$ rejected,
— in which case we loop
— and that happens with probability $\leq 1/2$…
why?
(if you tell me that this is even $\leq 1/4$, you are wrong)

Deciding L in **ZPP** = **RP** ∩ **coRP** with no error

- Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ such as here→

I.e., L is in **ZPP** iff there are
two poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

- if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
  $\mathcal{M}_2(x,r)$ accepts with prob.≥1/2
- if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.≤1/2
  $\mathcal{M}_2(x,r)$ rejects for every r [no error]

- Now run the following on input x:

`forever`:
    `if` $\mathcal{M}_1(x,\ldots)$ rejects: stop and **reject**
    `if` $\mathcal{M}_2(x,\ldots)$ accepts: stop and **accept**

then x cannot be in L (sure)

then x must be in L (sure)

Hence this machine **never makes any mistake**

It may be that $\mathcal{M}_1(x,\ldots)$ accepted and $\mathcal{M}_2(x,\ldots)$ rejected,
— in which case we loop
— and that happens with probability ≤1/2…
why?
(if you tell me that this is even ≤1/4, you are wrong)

We will see that this implies that the machine terminates in **≤ 2 turns** of the loop **on average**

A technical problem

- All this requires us to draw **arbitrarily long** bitstrings

A technical problem

- All this requires us to draw **arbitrarily long** bitstrings
- In fact, even **infinite** bit strings (for those computations that do not terminate)

A technical problem

- All this requires us to draw **arbitrarily long** bitstrings
- In fact, even **infinite** bit strings (for those computations that do not terminate)
- Requires **measure theory**:
  there is a unique measure μ on $\{0,1\}^\omega$
  with σ-algebra generated by cylinders $w.\{0,1\}^\omega$
  such that $\mu(w.\{0,1\}^\omega) = 1/2^{|w|}$ (Carathéodory)

A technical problem

- All this requires us to draw **arbitrarily long** bitstrings
- In fact, even **infinite** bit strings (for those computations that do not terminate)
- Requires **measure theory**:
  there is a unique measure μ on $\{0,1\}^\omega$
  with σ-algebra generated by cylinders $w.\{0,1\}^\omega$
  such that $\mu(w.\{0,1\}^\omega) = 1/2^{|w|}$ (Carathéodory)
- We will happily ignore this.

Rejection sampling

- A classic probabilistic procedure (**rejection sampling**):

  `forever`:

  compute something (with some random data r), x;

  `if` $P(x)$ holds: stop and return x

- **Trick.** If:

  — the random bits are independent across turns of the loop

  — and $P(x)$ holds with **prob.** $\geq \alpha$ at each turn

  then rejection sampling terminates in

  $\mathbf{1/\alpha}$ **turns** of the loop **on average**.

Rejection sampling

- A classic probabilistic procedure (**rejection sampling**):

```
forever:
    compute something (with some random data r), x;
  if P(x) holds: stop and return x
```

- **Trick.** If:
  — the random bits are independent across turns of the loop
  — and $P(x)$ holds with **prob. $\geq \alpha$** at each turn
  then rejection sampling terminates in
  $1/\alpha$ turns of the loop **on average**.

Rejection sampling

- An classic probabilistic procedure (**rejection sampling**):

  `forever`:

  compute something (with some random data *r*), *x*;

  `if` $P(x)$ holds: stop and return x

  prob. $\geq \alpha$

- *Proof.* Let X be the random variable « # turns through the loop »

Rejection sampling

- An classic probabilistic procedure (**rejection sampling**):

  `forever`:

  compute something (with some random data r), x;

  `if` $P(x)$ holds: stop and return x prob. $\geq \alpha$

- *Proof*. Let X be the random variable « # turns through the loop »

- $\Pr(X \geq n) = \Pr(P$ failed at turns $1, \ldots, n-1)$

  $\leq (1-\alpha)^{n-1}$ (by **independence**)

Rejection sampling

- An classic probabilistic procedure (**rejection sampling**):

`forever`:

  compute something (with some random data r), x;

  `if` $P(x)$ holds: stop and return x

prob. $\geq \alpha$

- *Proof.* Let X be the random variable « # turns through the loop »

- $\Pr(X \geq n) = \Pr(P$ failed at turns $1, \ldots, n-1)$

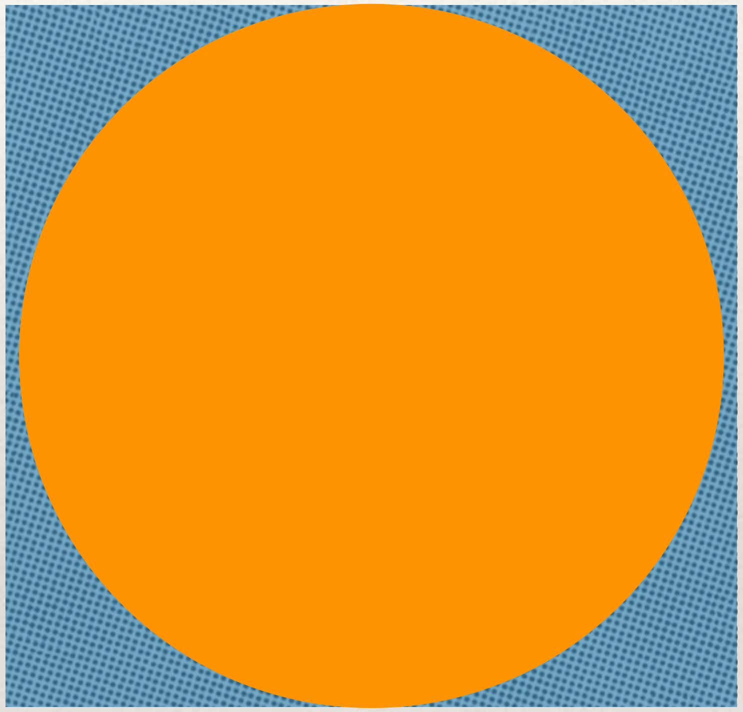
  $\leq (1-\alpha)^{n-1}$ (by **independence**)

- $E(X) = \sum_{n\geq 1} n.\Pr(X=n) = \sum_{n\geq 1} \Pr(X\geq n) \leq \sum_{n\geq 1} (1-\alpha)^{n-1} = 1/\alpha.$ $\square$

Expectation (average)

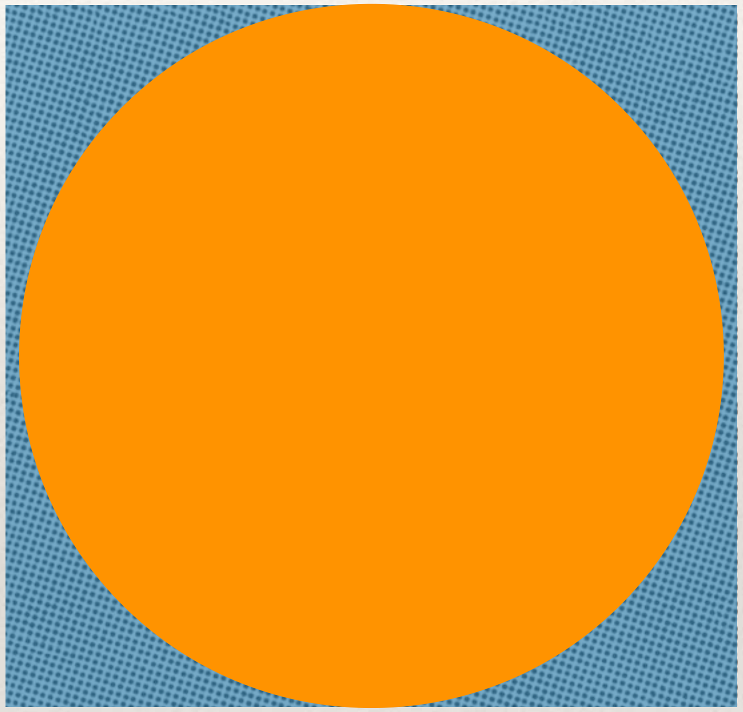
Rejection sampling: a typical application

- Draw a point inside the disc:
- Repeatedly draw a point inside the inscribing square
  - If it is in the disc, return it.



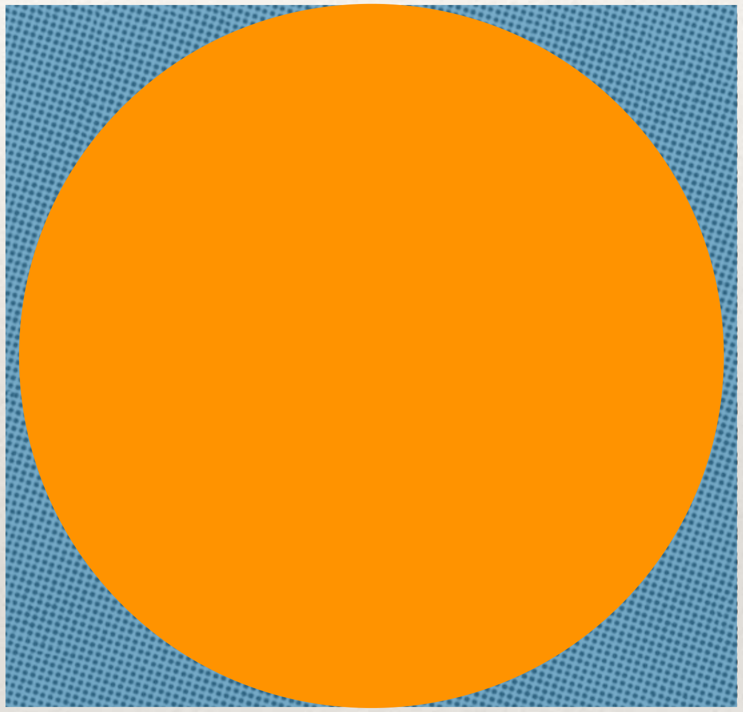
Rejection sampling: a typical application

- Draw a point inside the disc:
- Repeatedly draw a point inside the inscribing square
  - If it is in the disc, return it.
- Terminates in $\leq 4/\pi$ $\sim$ 1.27324 turns



Rejection sampling: a typical application

- Draw a point inside the disc:
- Repeatedly draw a point inside the inscribing square
  - If it is in the disc, return it.
- Terminates in $\leq 4/\pi$ ~ 1.27324 turns



- (Used as first step in the **Box-Muller** procedure drawing two independent numbers with a normal distribution)

Deciding L in **ZPP** = **RP** ∩ **coRP** with no error

- Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ such as here:
- Now run the following on input x:

`forever:`
    `if` $\mathcal{M}_1(x,\ldots)$ rejects: stop and **reject**
    `if` $\mathcal{M}_2(x,\ldots)$ accepts: stop and **accept**

I.e., L is in **ZPP** iff there are
two poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

- if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
  $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
- if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$
  $\mathcal{M}_2(x,r)$ rejects for every r [no error]

then x cannot be in L (sure)

then x must be in L (sure)

Hence this machine **never makes any mistake**

It may be that $\mathcal{M}_1(x,\ldots)$ accepted and $\mathcal{M}_2(x,\ldots)$ rejected,
— in which case we loop
— and that happens with probability $\leq 1/2\ldots$
(two cases: x in L, x not in L)

Deciding L in **ZPP** = **RP** ∩ **coRP** with no error

- Assume $\mathcal{M}_1$ and $\mathcal{M}_2$ such as here:
- Now run the following on input x:

I.e., L is in **ZPP** iff there are
two poly-time rand. TMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

- if $x \in L$ then $\mathcal{M}_1(x,r)$ accepts for every r [no error]
  $\mathcal{M}_2(x,r)$ accepts with prob.$\geq 1/2$
- if $x \notin L$ then $\mathcal{M}_1(x,r)$ accepts with prob.$\leq 1/2$
  $\mathcal{M}_2(x,r)$ rejects for every r [no error]

`forever:`
  `if` $\mathcal{M}_1(x,\ldots)$ rejects: stop and **reject**
  `if` $\mathcal{M}_2(x,\ldots)$ accepts: stop and **accept**

then x cannot be in L (sure)

then x must be in L (sure)

Hence this machine **never makes any mistake**

It may be that $\mathcal{M}_1(x,\ldots)$ accepted and $\mathcal{M}_2(x,\ldots)$ rejected,
— in which case we loop
— and that happens with probability $\leq 1/2$…
(two cases: x in L, x not in L)

This is **rejection sampling**:
stops in ≤ 2 turns on average
hence in **polytime on average.**

Markov's inequality

> Let us define **ZPP′** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.
>
> I claim that **ZPP** = **ZPP′**.

- Hence:
  ZPP (= **RP** $\cap$ **coRP**) $\subseteq$ **ZPP′**
- In order to show the reverse inclusion, we use:
- **Theorem (Markov's inequality).**
  Let X be a **non-negative real-valued** random variable with **finite** expectation $E(X)$. For every $a>0$:
  $\Pr(X \geq a.E(X)) \leq 1/a$.

Markov's inequality

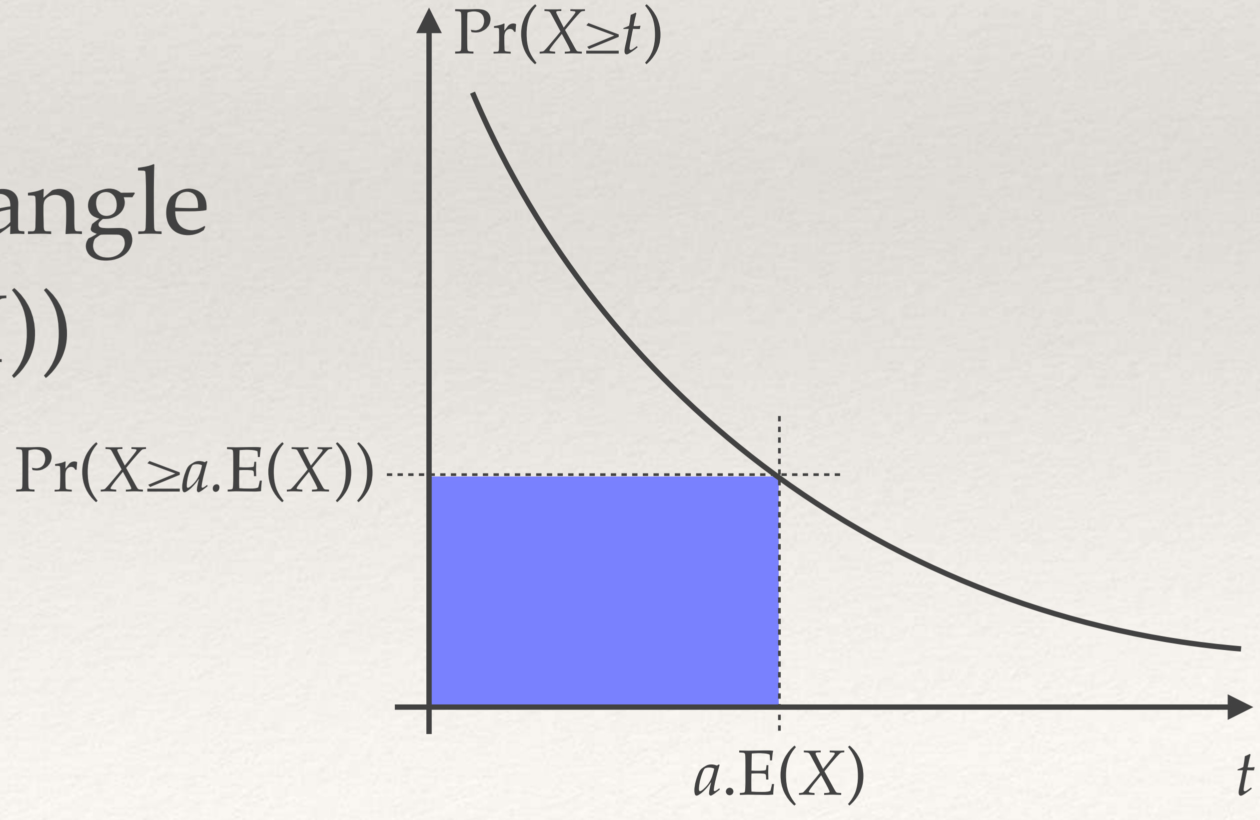
Let us define **ZPP′** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.

I claim that **ZPP** = **ZPP′**.

- Hence:
  ZPP (= **RP** $\cap$ **coRP**) $\subseteq$ **ZPP′**
- In order to show the reverse inclusion, we use:
- **Theorem (Markov's inequality).**
  Let X be a **non-negative real-valued** random variable with **finite** expectation $E(X)$. For every $a>0$:
  $\Pr(X \geq a.E(X)) \leq 1/a$.

Markov's inequality

- **Theorem (Markov's inequality).**
  Let X be a **non-negative real-valued** random variable with **finite** expectation $E(X)$. For every $a>0$:
  $\Pr(X \geq a.E(X)) \leq 1/a$.
- *Proof.* $E(X) = \int_t \Pr(X \geq t)\, dt$
  $\geq$ area of the blue rectangle
  $= a\,.\,E(X)\,.\,\Pr(X \geq a.E(X))$
  Then divide out by $a\,.\,E(X)$. □



The reverse inclusion **ZPP' ⊆ ZPP**

Let us define **ZPP′** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.

I claim that **ZPP** = **ZPP′**. Recall **ZPP** = **RP** ∩ **coRP**

- Let L in **ZPP'**, decided by $\mathcal{M}$ running in **average** poly. time $p(n)$ with **no** error.
- Define $\mathcal{M}_1$ as follows: on input x (and random tape r of size $a.\ p(n)$)

simulate $\mathcal{M}$ on x for at most $a.\ p(n)$ steps (**timeout**).
If timeout reached, then **accept** (that may be an error).

The reverse inclusion **ZPP’ ⊆ ZPP**

- Markov on r.v. X = time taken by $\mathcal{M}$ on x; also let a=2.
- $E(X) \leq p(n)$ **finite** OK

Let us define **ZPP′** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.

I claim that **ZPP** = **ZPP′**. Recall **ZPP** = **RP** ∩ **coRP**

Let L in **ZPP′**, decided by $\mathcal{M}$ running in **average** poly. time $p(n)$ with **no** error.

Define $\mathcal{M}_1$ as follows: on input x (and random tape r of size $a.\ p(n)$)

simulate $\mathcal{M}$ on x for at most $a.\ p(n)$ steps (**timeout**).

If timeout reached, then **accept** (that may be an error).

The reverse inclusion **ZPP’** $\subseteq$ **ZPP**

Let us define **ZPP’** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.

I claim that **ZPP** = **ZPP’**. Recall **ZPP** = **RP** $\cap$ **coRP**

Let L in **ZPP’**, decided by $\mathcal{M}$ running in **average** poly. time $p(n)$ with **no** error.

Define $\mathcal{M}_1$ as follows: on input x (and random tape r of size $a.\, p(n)$)

simulate $\mathcal{M}$ on x for at most $a.\, p(n)$ steps (**timeout**).

If timeout reached, then **accept** (that may be an error).

- Markov on r.v. X = time taken by $\mathcal{M}$ on x; also let a=2.
- $E(X) \leq p(n)$ **finite** OK
- If $x \notin L \Rightarrow$ error = $\Pr_r(\mathcal{M}_1(x,r)$ accepts)

$$
\begin{aligned}
&= \Pr(X \geq a.\, p(n)) && (\mathcal{M} \text{ makes no mistake})\\
&\leq \Pr(X \geq a.\, E(X)) && (E(X) \leq p(n))\\
&\leq 1/a = 1/2 && (\text{Markov})
\end{aligned}
$$

The reverse inclusion **ZPP’ $\subseteq$ ZPP**

Let us define **ZPP’** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.

I claim that **ZPP** = **ZPP’**. Recall **ZPP** = **RP** $\cap$ **coRP**

Let L in **ZPP’**, decided by $\mathcal{M}$ running in **average** poly. time $p(n)$ with **no** error.

Define $\mathcal{M}_1$ as follows: on input x (and random tape r of size $a.\ p(n)$)

simulate $\mathcal{M}$ on x for at most $a.\ p(n)$ steps (**timeout**).

If timeout reached, then **accept** (that may be an error).

- Markov on r.v. X = time taken by $\mathcal{M}$ on x; also let a=2.
- $E(X) \leq p(n)$ **finite** OK
- If $x \notin L \Rightarrow$ error = $\Pr_r(\mathcal{M}_1(x,r)$ accepts$)$

$$\begin{aligned} &= \Pr(X \geq a.\ p(n)) && (\mathcal{M} \text{ makes no mistake}) \\ &\leq \Pr(X \geq a.\ E(X)) && (E(X) \leq p(n)) \\ &\leq 1/a = 1/2 && (\text{Markov}) \end{aligned}$$

- If $x \in L \Rightarrow \mathcal{M}_1(x,r)$ must accept.

The reverse inclusion **ZPP’** $\subseteq$ **ZPP**

> Let us define **ZPP’** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.
>
> I claim that **ZPP** = **ZPP’**. Recall **ZPP** = **RP** $\cap$ **coRP**
>
> Let L in **ZPP’**, decided by $\mathcal{M}$ running in **average** poly. time $p(n)$ with **no** error.
>
> Define $\mathcal{M}_1$ as follows: on input x (and random tape r of size $a.\,p(n)$)
>
> **simulate** $\mathcal{M}$ on x for at most $a.\,p(n)$ steps (**timeout**).
>
> If timeout reached, then **accept** (that may be an error).

- Markov on r.v. X = time taken by $\mathcal{M}$ on x; also let a=2.
- $E(X) \leq p(n)$ **finite** OK
- If $x \notin L \Rightarrow$ error = $\Pr_r(\mathcal{M}_1(x,r)$ accepts)

$$
\begin{aligned}
&= \Pr(X \geq a.\,p(n)) && (\mathcal{M} \text{ makes no mistake})\\
&\leq \Pr(X \geq a.\,E(X)) && (E(X) \leq p(n))\\
&\leq 1/a = 1/2 && (\text{Markov})
\end{aligned}
$$

- If $x \in L \Rightarrow \mathcal{M}_1(x,r)$ must accept.
- Hence L is in **coRP**.

The reverse inclusion **ZPP' ⊆ ZPP**

Symmetrically:

- Markov on r.v. X = time taken by $\mathcal{M}$ on x; also let a=2.
- $E(X) \leq p(n)$ **finite** OK
- If $x \in L \Rightarrow$ error = $\Pr_r(\mathcal{M}_2(x,r)$ ~~accepts~~ rejects)

$$= \Pr(X \geq a.\ p(n)) \quad (\mathcal{M} \text{ makes no mistake})$$
$$\leq \Pr(X \geq a.\ E(X)) \quad (E(X) \leq p(n))$$
$$\leq 1/a = 1/2 \quad (\text{Markov})$$

- If $x \notin L \Rightarrow \mathcal{M}_2(x,r)$ must ~~accept~~ reject.
- Hence L is in ~~**coRP**~~ **RP.**

Let us define **ZPP'** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.

I claim that **ZPP** = **ZPP'**. Recall **ZPP** = **RP** ∩ **coRP**

Let L in **ZPP'**, decided by $\mathcal{M}$ running in **average** poly. time $p(n)$ with **no** error.

Define $\mathcal{M}_2$ as follows: on input x (and random tape r of size $a.\ p(n)$)

simulate $\mathcal{M}$ on x for at most $a.\ p(n)$ steps (**timeout**).

If timeout reached, then ~~accept~~ **reject** (that may be an error).

The reverse inclusion **ZPP' ⊆ ZPP**

Symmetrically:

Let us define **ZPP'** (for now) as the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.

I claim that **ZPP** = **ZPP'**. Recall **ZPP** = **RP** ∩ **coRP**

Let L in **ZPP'**, decided by $\mathcal{M}$ running in **average** poly. time $p(n)$ with **no** error.

Define $\mathcal{M}_2$ as follows: on input x (and random tape r of size $a.\ p(n)$)

simulate $\mathcal{M}$ on x for at most $a.\ p(n)$ steps (**timeout**).

If timeout reached, then ~~accept~~ **reject** (that may be an error).

- Markov on r.v. X = time taken by $\mathcal{M}$ on x; also let a=2.
- $E(X) \leq p(n)$ **finite** OK
- If $x \in L \Rightarrow$ error $= \Pr_r(\mathcal{M}_2(x,r)$ ~~accepts~~ rejects)

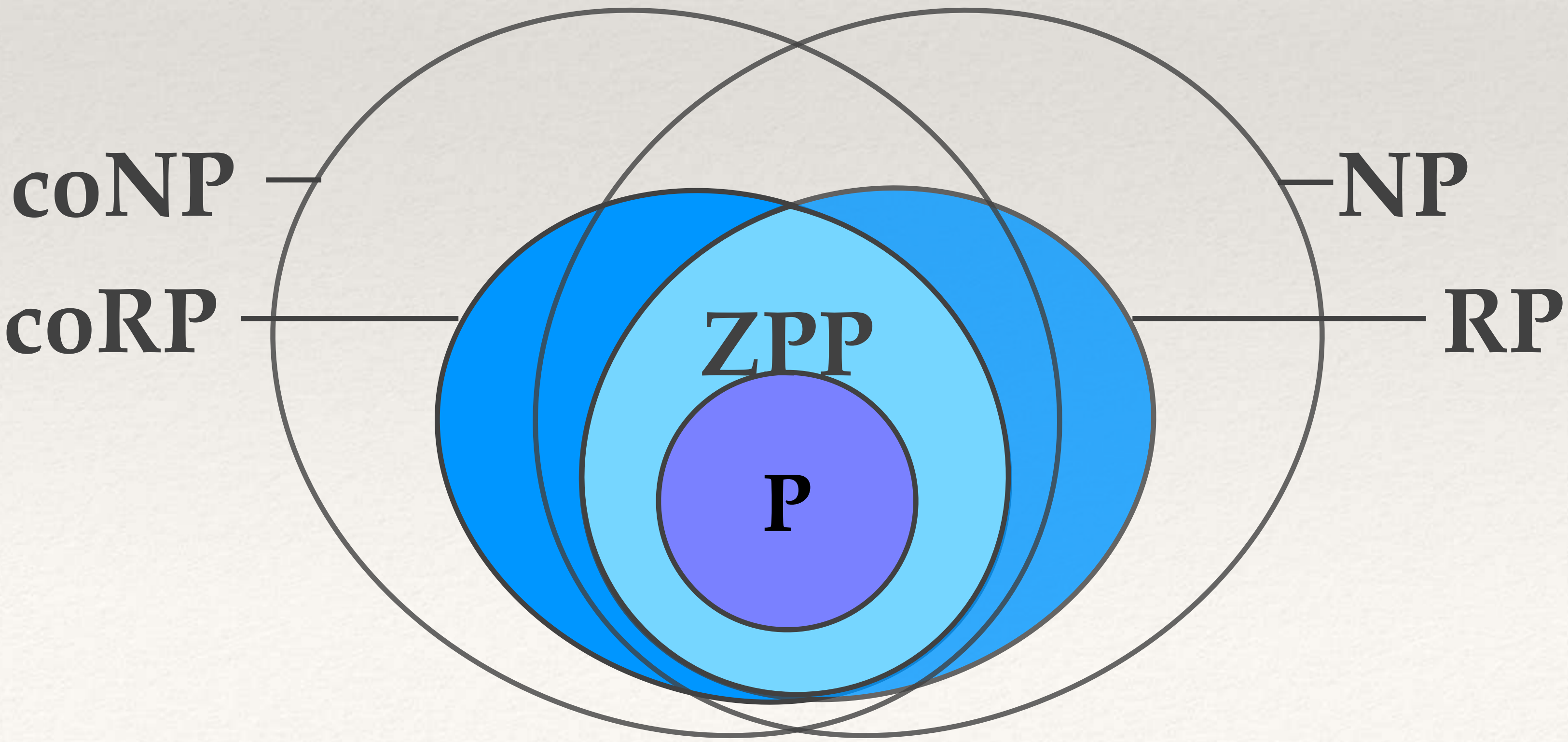
$$= \Pr(X \geq a.\ p(n)) \quad (\mathcal{M} \text{ makes no mistake})$$
$$\leq \Pr(X \geq a.\ E(X)) \quad (E(X) \leq p(n))$$
$$\leq 1/a = 1/2 \quad (\text{Markov})$$

- If $x \notin L \Rightarrow \mathcal{M}_2(x,r)$ must ~~accept~~ reject.
- Hence L is in ~~coRP~~ **RP.**

Hence L is both in **RP** and in **coRP**, namely in **ZPP**. □

Summary on **ZPP**

- **Definition. ZPP** = **RP** ∩ **coRP**
- **Theorem. ZPP** is the class of languages L which we can decide in **average** polynomial-time with probability **zero** of making a mistake.



Next time…

BPP: Bounded Prob. of Error Polynomial time

- A language L is in **BPP** if and only if there is a **polynomial-time** TM $\mathcal{M}$ such that for every input x (of size n):
- if $x \in L$ then $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \geq 2/3$
- if $x \notin L$ then $\Pr_r[\mathcal{M}(x,r) \text{ accepts}] \leq 1/3$.

two-sided error:
$\Pr_r[\mathcal{M}(x,r) \text{ errs}] \leq 1/3$