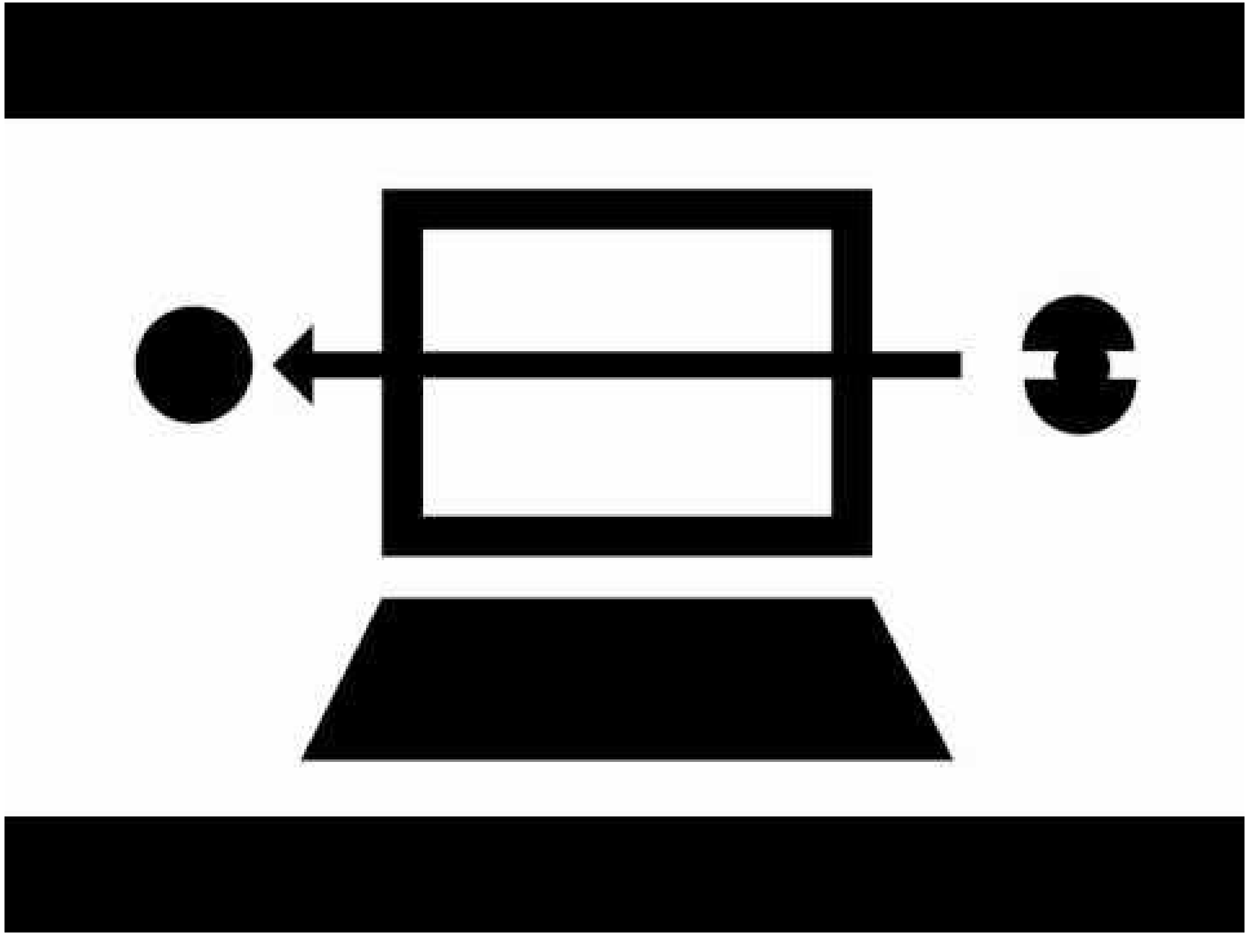
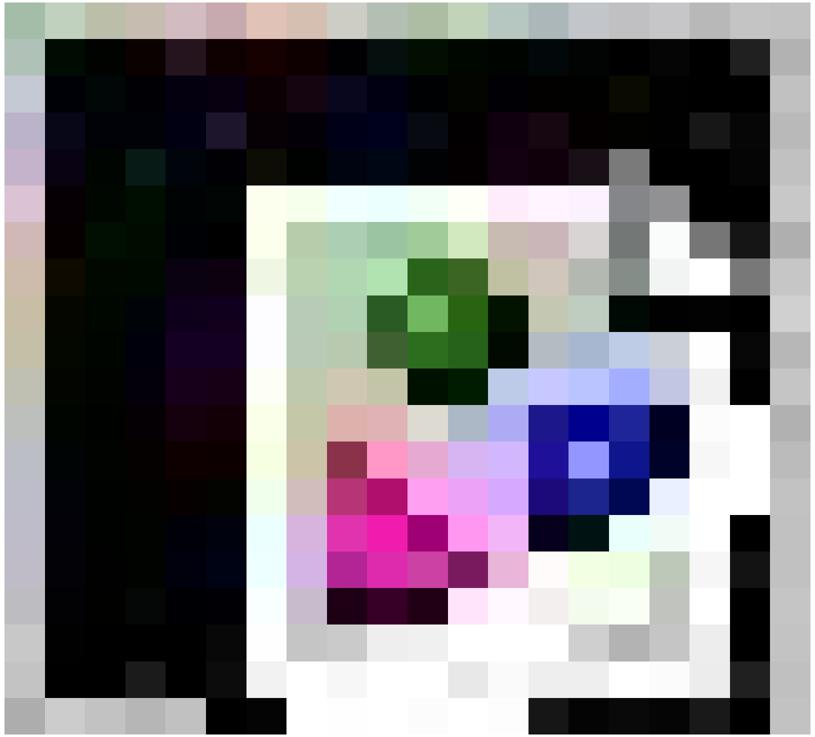
Probabilistic Computation

Harold Frank and Alexa Becker



Computational complexity theory



Branch of the theory of computation in theoretical computer science and mathematics Classifying computational problems into specific sets according to their inherent difficulty, and relating those classes to each other.

Complexity Class

A **complexity class** is a set of problems that can be solved with a quantum Turing machine in polynomial time, **exactly**.

The most commonly used problems are **decision problems.** However, complexity classes can be defined based on function problems, counting problems, optimization problems, promise problems, etc.

Why does this matter?

The ability to make probabilistic decisions often helps algorithms solve problems more efficiently.

Complexity Class

A non-deterministic Turing machine is a deterministic Turing machine with an added feature of non-determinism, which allows a Turing machine to have multiple possible future actions from a given state.

One way to view non-determinism is that the Turing machine branches into many possible computational paths at each step, and if it solves the problem in any of these branches, it is said to have solved the problem.

**1

Probabilistic Computers take a simulation problem (forward) into an inference program (reverse).

Computer: input to output

Simulation problems: background assumptions, produces output like a trajectory, going from cause to effects

Inference problems: same background assumptions, inputs trajectory, outputs a configuration of the world that explains it, effects back to causes. However, there is fundamental uncertainty (probability) for what the cause(s) is. .

***2

Output depends not only on x (input) but also on a random variable(s) and a probability distribution(s).

Different versions of Probability Machine:

The machine rejects input x if it has probability = 0, and it takes x for probability > 0.

‘ ‘ ‘ ‘ ‘ ‘ probability < ½, and it takes x for probability >= ½.

‘ ‘ ‘ ‘ ‘ ‘ probability <= ⅓, and it takes x for probability > ⅔.

You can set bounds on the input of x

**3

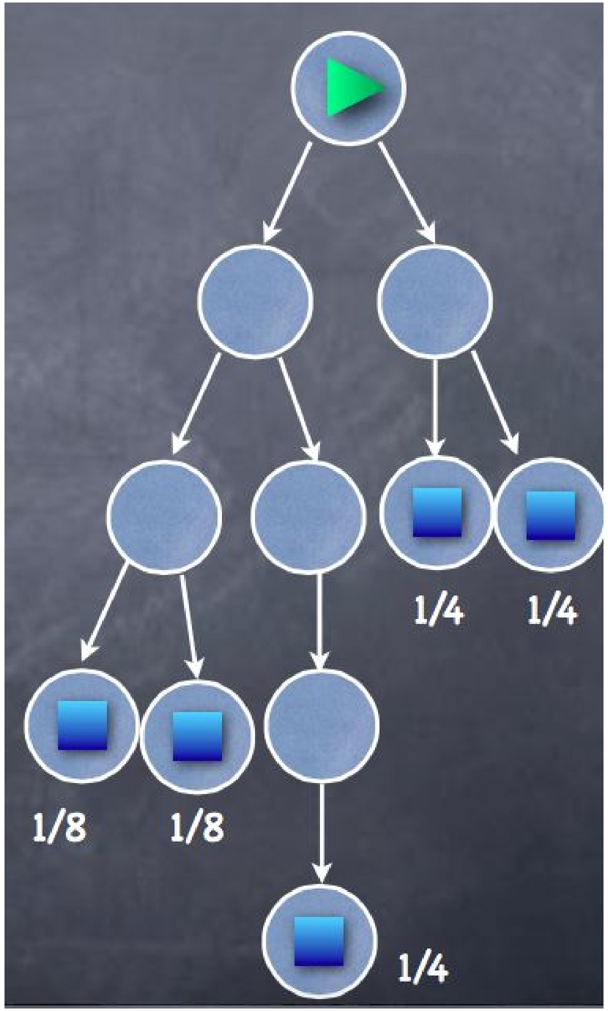
The output will give us specific answers (inferences) based on our conditions on x.

If the restrictions for x are say Probability >= 2/3 and Probability != 0, then if an x is input with Probability in between, then the machine will do a random guess.

If not, the machine will give us an answer with much higher likelihood of being right.

*** Transition Machines

(simply) Like normal probabilistic computing, but there are probabilities to determine if the next step moves on.



***Random Tape

Acts like an activation switch.

If enough random “coin flips” are recorded, the machine will move down the circuit.

A circuit may link to another circuit, or it may have a probability of outputting result.

***Languages Decided by Probabilistic Computation

The machine language that is decided depends on the input x.

PTIME BTIME RTIME languages

Time dependent

***List of some Languages

Accept if pr[yes] > 0; reject if pr[yes] = 0 NTM

Accept if pr[yes] = 1; reject if pr[yes] < 1 co-NTM

Accept if pr[yes] > 1/2; reject if pr[yes] < 1/2 PTM

Accept if pr[yes] " 1/2; reject if pr[yes] < 1/2 co-PTM

Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 BPTM

Accept if pr[yes] > 2/3; reject if pr[yes] < 1/3 co-BPTM

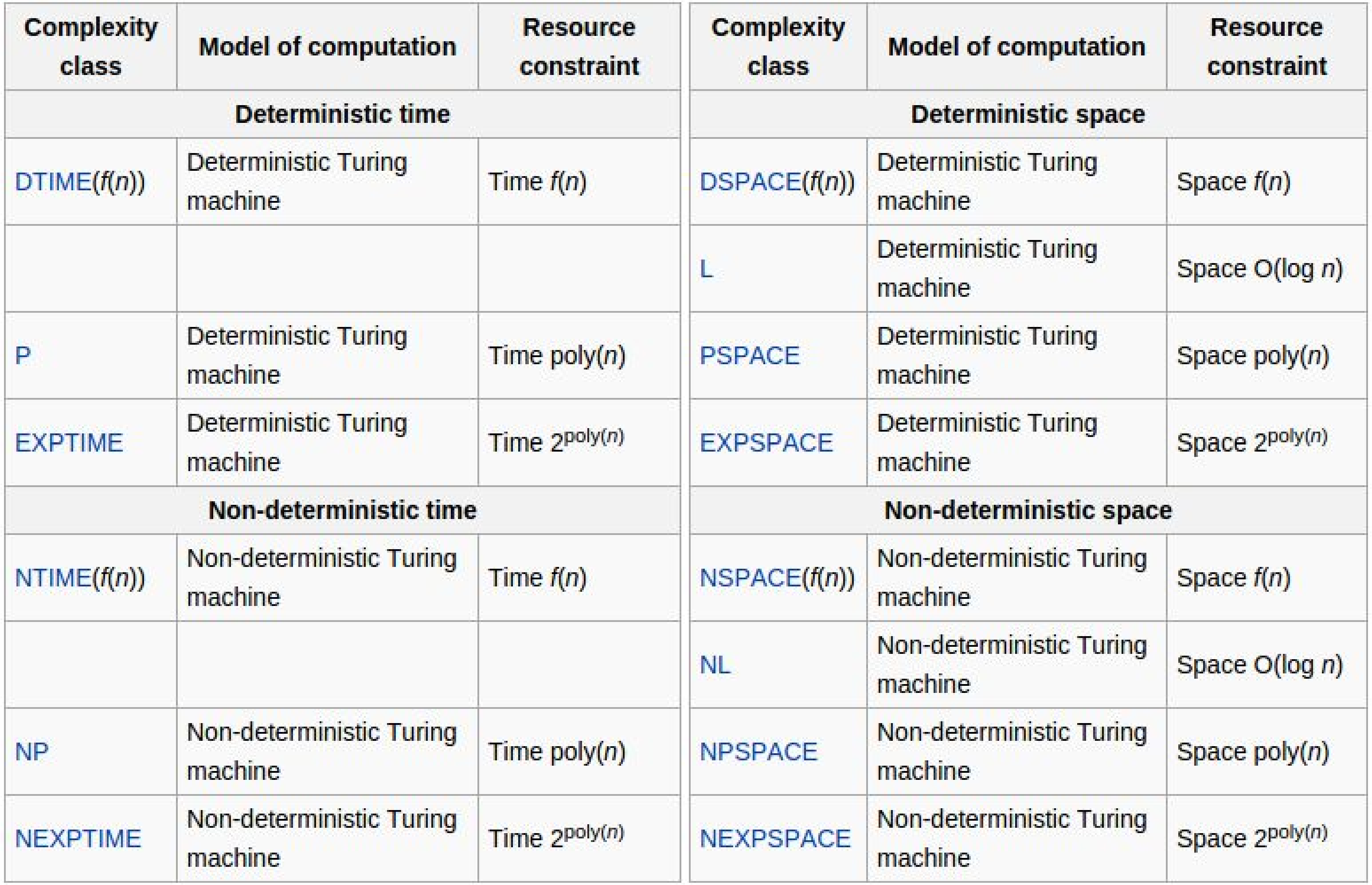
Accept if pr[yes] > 2/3; reject if pr[yes] = 0 RTM

Accept if pr[yes] = 1; reject if pr[yes] < 1/3 co-RTM

✻✻✻

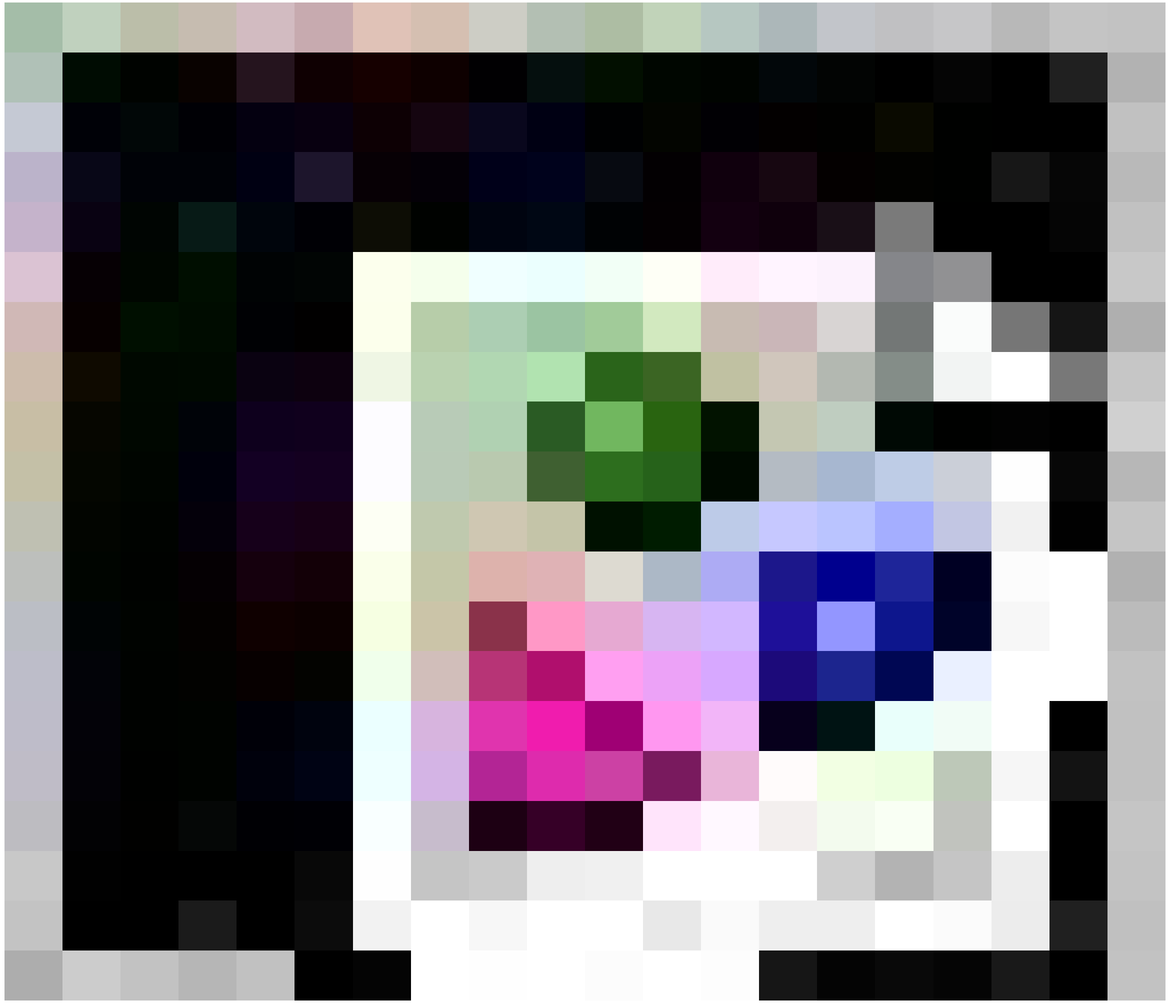
Languages will output an answer with a probability of being right or wrong. Say always ⅔ right and always ⅓ wrong.

It may output an answer that is “I don’t know” with a probability of say ½



Complexity class	Model of computation	Resource constraint
Deterministic time		
DTIME($f(n)$)	Deterministic Turing machine	Time $f(n)$
P	Deterministic Turing machine	Time poly(n)
EXPTIME	Deterministic Turing machine	Time $2^{\text{poly}(n)}$
Non-deterministic time		
NTIME($f(n)$)	Non-deterministic Turing machine	Time $f(n)$
NP	Non-deterministic Turing machine	Time poly(n)
NEXPTIME	Non-deterministic Turing machine	Time $2^{\text{poly}(n)}$

Complexity class	Model of computation	Resource constraint
Deterministic space		
DSPACE($f(n)$)	Deterministic Turing machine	Space $f(n)$
L	Deterministic Turing machine	Space O(log n)
PSPACE	Deterministic Turing machine	Space poly(n)
EXPSPACE	Deterministic Turing machine	Space $2^{\text{poly}(n)}$
Non-deterministic space		
NSPACE($f(n)$)	Non-deterministic Turing machine	Space $f(n)$
NL	Non-deterministic Turing machine	Space O(log n)
NPSPACE	Non-deterministic Turing machine	Space poly(n)
NEXPSPACE	Non-deterministic Turing machine	Space $2^{\text{poly}(n)}$



BPP

BPP: bounded-error probabilistic polynomial time

A problem is in BPP if there is a polynomial time probabilistic algorithm such that:

in **BPP** if and only if there exists a probabilistic Turing machine *M*, such that

- *M* runs for polynomial time on all inputs
- For all *x* in *L*, *M* outputs 1 with probability greater than or equal to 2/3
- For all *x* not in *L*, *M* outputs 1 with probability less than or equal to 1/3

BPR

BPR: randomized probabilistic polynomial time

- For all *x NOT* in *L*, *M* outputs 1 with probability 1
- no false positives

ZPP

ZPP: zero error probabilistic polynomial time

the class of problems for which a probabilistic Turing machine exists with these properties:

- It always runs in polynomial time.
- It returns an answer YES, NO or DO NOT KNOW.
- The answer is always either DO NOT KNOW or the correct answer.
- It returns DO NOT KNOW with probability at most 1/2 (and the correct answer otherwise).

BQP

BQP: bounded-error quantum polynomial time

if and only if there exists a $\{Q_n : n \in \mathbb{N}\}$ polynomial-time uniform family of quantum circuits , such that

- For all $n \in \mathbb{N}$ Q_n takes *n* qubits as input and outputs 1 bit
- For all *x* in *L*, $\Pr(Q_{|x|}(x) = 1) \geq \frac{2}{3}$
- For all *x* not in *L*, $\Pr(Q_{|x|}(x) = 0) \geq \frac{2}{3}$

Can be solved in worst case polynomial time by a quantum computer with probability > ⅔ (probability of error is bounded, thus the "B.")

ZQP

ZQP: zero error quantum polynomial time

- For all *x* in *L*, *M* outputs "I don't know" with a probability < ½
- Can be solved with zero error probability in expected polynomial time with a quantum computer

EQP

EQP: equational prover

- solvable by a quantum algorithm in polynomial time with zero error

The End