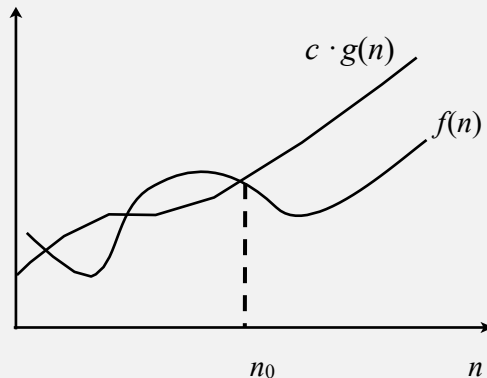
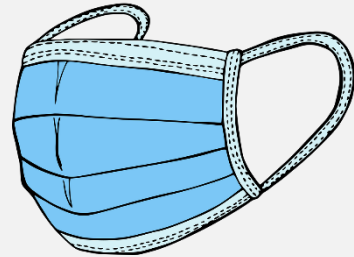
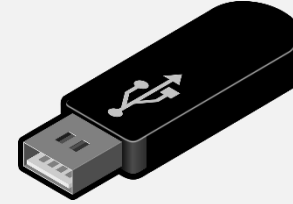
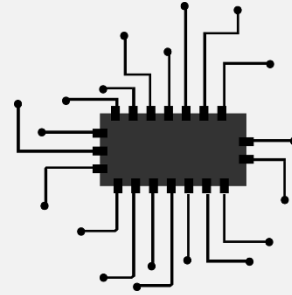
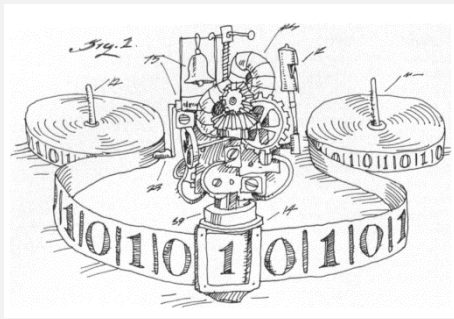


COMP 3602

Intro to Theory of Computation

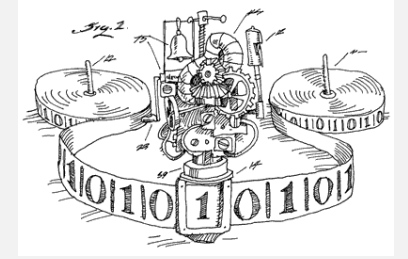
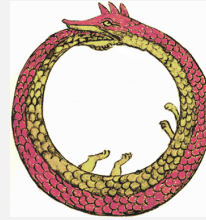
Lecture 15

Announcements



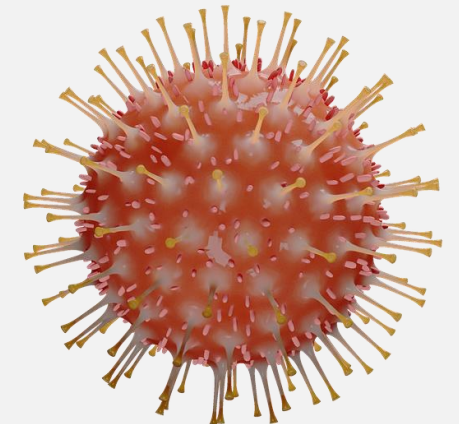
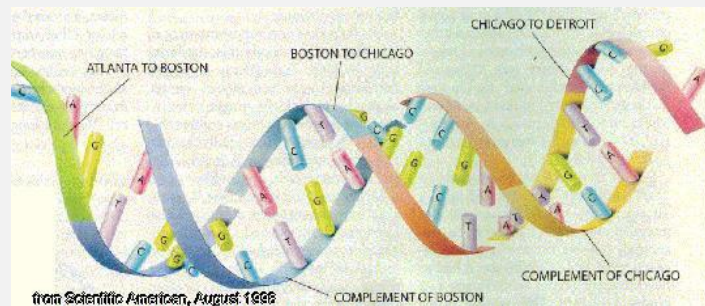
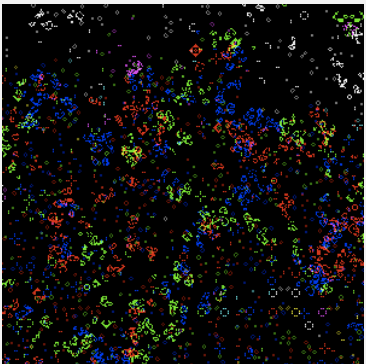
- Presentation proposals due today
 - I will ask for slides closer to the talks.
- Computability drill due Mar 12
 - One more drill (on complexity) after that
- Drop date was extended till March 10th
- Two more tests left:
 - Oral exam on NP-completeness
 - Written exam on all the material we covered in the course.

Questions/comments?



Efficient computation

What can be computed in a reasonable amount
of time/space/etc?



Defining “efficient”: polynomial time

Polynomial time (polytime). The worst-case running time of the algorithm on an input of length n is in $O(n^d)$ for some constant $d \geq 0$.

Desirable scaling property. When the input size doubles, the algorithm should slow down by at most some multiplicative constant factor C .

Alternative definition. An algorithm is **polynomial time (polytime)** if the above scaling property holds.

There exist constants $c > 0$ and $d > 0$ such that for every input of size n , the running time of the algorithm is bounded above by $c n^d$ primitive computational steps.

← corresponds
to $C = 2^d$



von Neumann
(1953)



Nash
(1955)



Gödel
(1956)



Cobham
(1964)



Edmonds
(1965)



Rabin
(1966)

Polynomial in input length $\sim$ efficient

We say that an algorithm is **efficient** if it has a polynomial worst-case asymptotic running time as a function of its input length.

Some poly-time algorithms have galactic constants and/or huge exponents.

Theory. Definition is (relatively) insensitive to a (deterministic) model of computation.

- The difference between single-tape and multi-tape TM running time is at most quadratic.
- Non-deterministic, randomized and quantum – may be exponential!

n^{120}
↙

Practice. It really works!

- The poly-time algorithms that people develop usually have both small constants and small exponents.
- Breaking through the exponential barrier of brute force typically exposes some crucial structure of the problem.

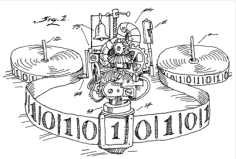
Mikkel Thorup
Department of Computer Science, University of
Copenhagen Universitetsparken 1, DK-2100 Copenhagen
East, Denmark mthorup@diku.dk

Map graphs in polynomial time Abstract

Chen, Grigni, and Papadimitriou (WADS'97 and STOC'98) have introduced a modified notion of planarity, where two faces are considered adjacent if they share at least one point. The corresponding abstract graphs are called map graphs. Chen et.al. raised the question of whether map graphs can be recognized in polynomial time. They showed that the decision problem is in NP and presented a polynomial time algorithm for the special case where we allow at most 4 faces to intersect in any point — if only 3 are allowed to intersect in a point, we get the usual planar graphs.

Chen, et.al. conjectured that map graphs can be recognized in polynomial time, and in this paper, their conjecture is settled affirmatively.

Which time bound would you prefer: $20 n^{120}$ or $n^{1+0.02 \log n}$?

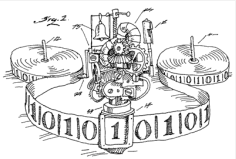


Running time of a Turing machine

- Given a deterministic TM M , let the function $T_M: \mathbb{N} \rightarrow \mathbb{N}$ be: $\forall n \in \mathbb{N}$,
$$T_M(n) = \max_{x, |x|=n} \{ \text{number of steps } M \text{ takes to halt on } x \}$$
 - If M goes into an infinite loop on some x of length n , $T_M(n) = \infty$
 - This also applies to TMs computing a function.

Definition: A function g is *polytime (efficiently) computable* iff there exists a TM M computing g and constants c, d such that $\forall n \ T_M(n) \leq cn^d$

Definition: A language L is *polytime (efficiently) decidable* iff there exists a TM M deciding L and constants c, d such that $\forall n \ T_M(n) \leq cn^d$



Time complexity classes

– $T_M(n) = \max_{x, |x|=n} \{ \text{number of steps } M \text{ takes to halt on } x \}$

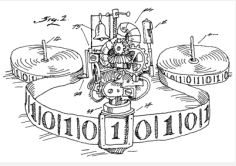
- More generally, let $f: \mathbb{N} \rightarrow \mathbb{N}$ be a function. Define a class of languages decidable in time f

$$\text{Time}(f(n)) = \{L \mid \exists M \ \mathcal{L}(M) = L \text{ and } T_M \in O(f)\}$$

- This gives us another definition of a language L being polytime decidable:
 $\exists d \ L \in \text{Time}(n^d)$.

- *Definition:* A class P of polytime decidable languages is

$$P = \bigcup_{d \in \mathbb{N}} \text{Time}(n^d)$$



Time complexity classes

- $T_M(n) = \max_{x, |x|=n} \{ \text{number of steps } M \text{ takes to halt on } x \}$
- $\text{Time}(f(n)) = \{L \mid \exists M \ \mathcal{L}(M) = L \text{ and } T_M \in O(f) \}$

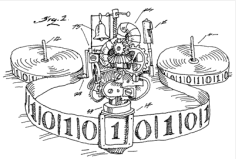
- *Definition:* A class P of polytime decidable languages is

$$P = \bigcup_{d \in \mathbb{N}} \text{Time}(n^d)$$

- *Definition:* A class EXP of exponential-time decidable languages is

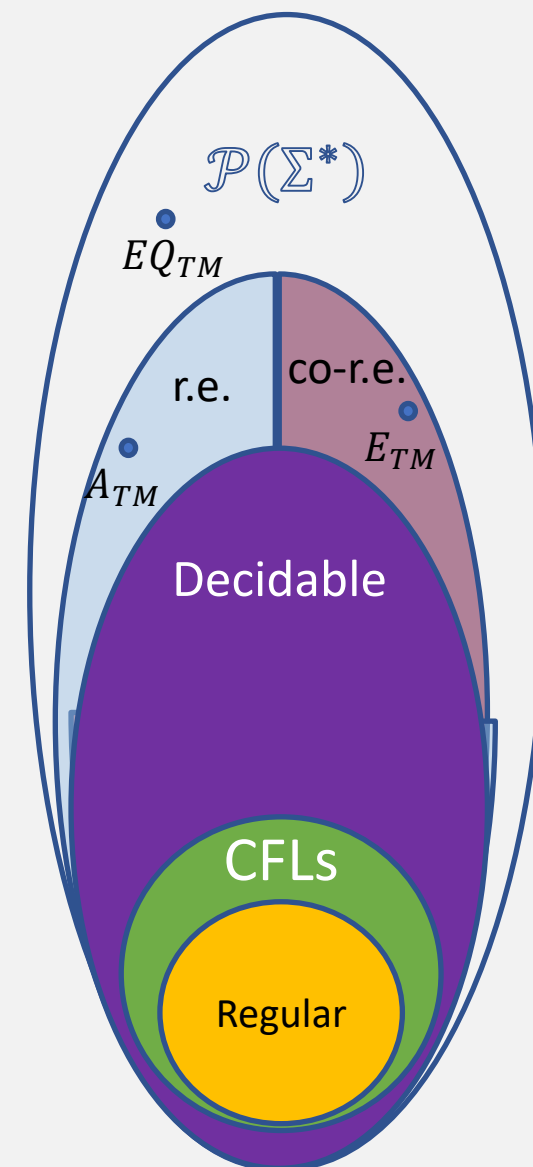
$$P = \bigcup_{d \in \mathbb{N}} \text{Time}(2^{n^d})$$

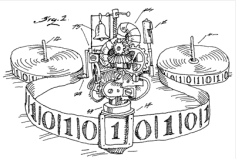
... and so on.



Scaling down to complexity

- We scaled down from “decidable” and “computable” to “efficiently decidable” and “efficiently computable”
 - Polynomial time $\equiv$ efficient
- Let’s scale down everything else!
 - All we do is to add “polynomial” to definitions of our computability classes.





Scaling down to complexity

“efficient”, “small”, “easy” $\equiv$ polynomial

- Decidable
 - Turing-recognizable
- For some decidable R
- $$L = \{x \mid \exists y R(x, y)\}$$

– P

– NP

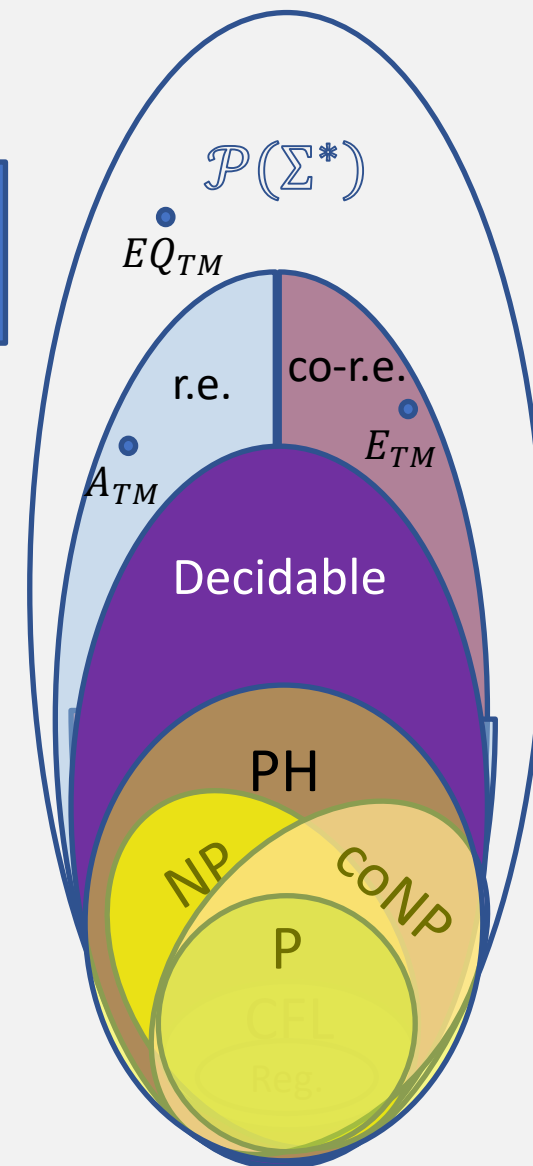
For some **polytime** decidable R ,
some constants c, d

$$L = \{x \mid \exists y \mid y \mid \leq cn^d \wedge R(x, y)\}$$

- Co-r.e
- Arithmetic hierarchy

– CoNP

– Polytime hierarchy (PH)

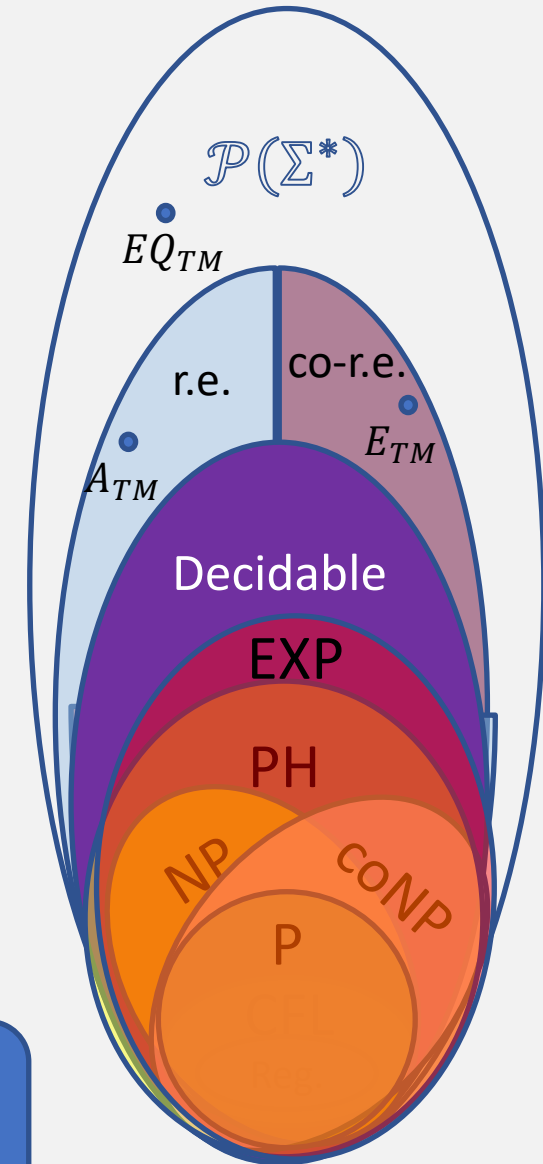


Complexity class NP

Complexity class NP: Problems for which *checking* solutions is in polynomial time: $L = \{x \mid \exists y \ |y| \leq cn^d \wedge R(x, y)\}$

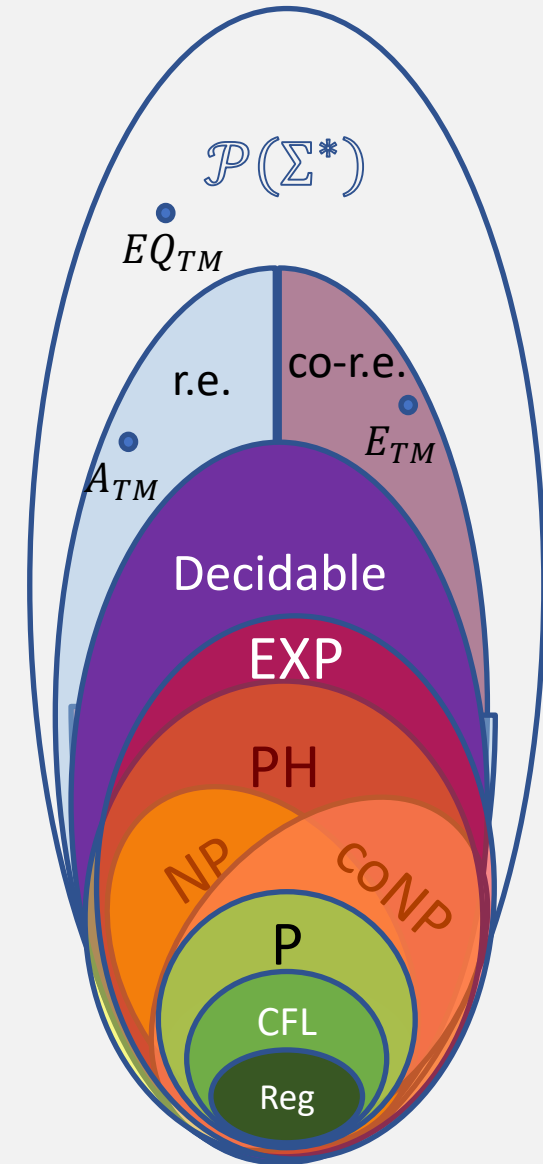
- Theorem: $P \subseteq NP \subseteq EXP$
 - $P \subseteq NP$: if an algorithm found solution, it exists. To check, compute it.
 - $NP \subseteq EXP$: There are $2 \cdot 2^{cn^d} - 1$ possible solutions y of size $\leq cn^d$ and checking each one takes polynomial time (say $O(n^k)$). Checking all takes time $O(n^k \cdot 2^{cn^d})$
- Note: NP contains a lot of easy problems! Exponential time is only an upper bound (O rather than Θ)

Are there any problems in NP for which the brute-force algorithm above is the fastest? That's a million dollar question!



Time complexity classes

- $T_M(n) = \max_{x, |x|=n} \{ \text{number of steps } M \text{ takes to halt on } x \}$
- $DTime(f(n)) = \{ L \mid \exists M \ \mathcal{L}(M) = L \text{ and } T_M \in O(f) \}$
 - Where M is a deterministic TM.
- $P = \bigcup_{d \in \mathbb{N}} DTime(n^d)$
- $EXP = \bigcup_{d \in \mathbb{N}} DTime(2^{n^d})$
- $L \in NP$ iff for some **polytime** decidable R , **some constants** c, d , $L = \{ x \mid \exists y \ |y| \leq cn^d \wedge R(x, y) \}$
- $L \in coNP$ iff $\bar{L} \in NP$
- k -th level of PH: generalizes NP and coNP to k quantifier alternations, all variables bounded by cn^d .
 - k is constant.
 - Σ_k^p is when the k -quantifier condition starts with $\exists$, Π_k^p with $\forall$.





Another definition of NP

- $T_M(n) = \max_{x, |x|=n} \{ \text{number of steps } M \text{ takes to halt on } x \}$. M is deterministic.
- $DTime(f(n)) = \{L \mid \exists M \ \mathcal{L}(M) = L \text{ and } T_M \in O(f)\}$
- $P = \bigcup_{d \in \mathbb{N}} DTime(n^d)$

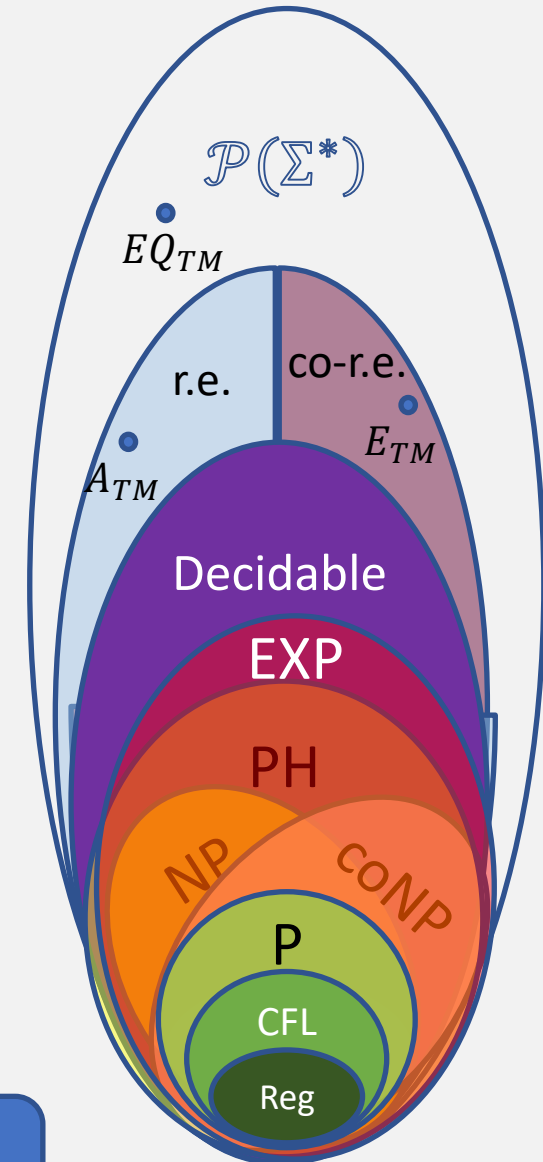
– For a **non-deterministic** TM M , define

$$NT_M(n) = \max_{x, |x|=n} \{ \text{height of configuration tree of } M \text{ on } x \}$$

- That is, the longest numbers of steps it takes to halt over all branches of its non-deterministic computation, whether accepting or not.

– $NTime(f(n)) = \{L \mid \exists M \ \mathcal{L}(M) = L \text{ and } NT_M \in O(f)\}$

- **Definition:** $NP = \bigcup_{d \in \mathbb{N}} NTime(n^d)$

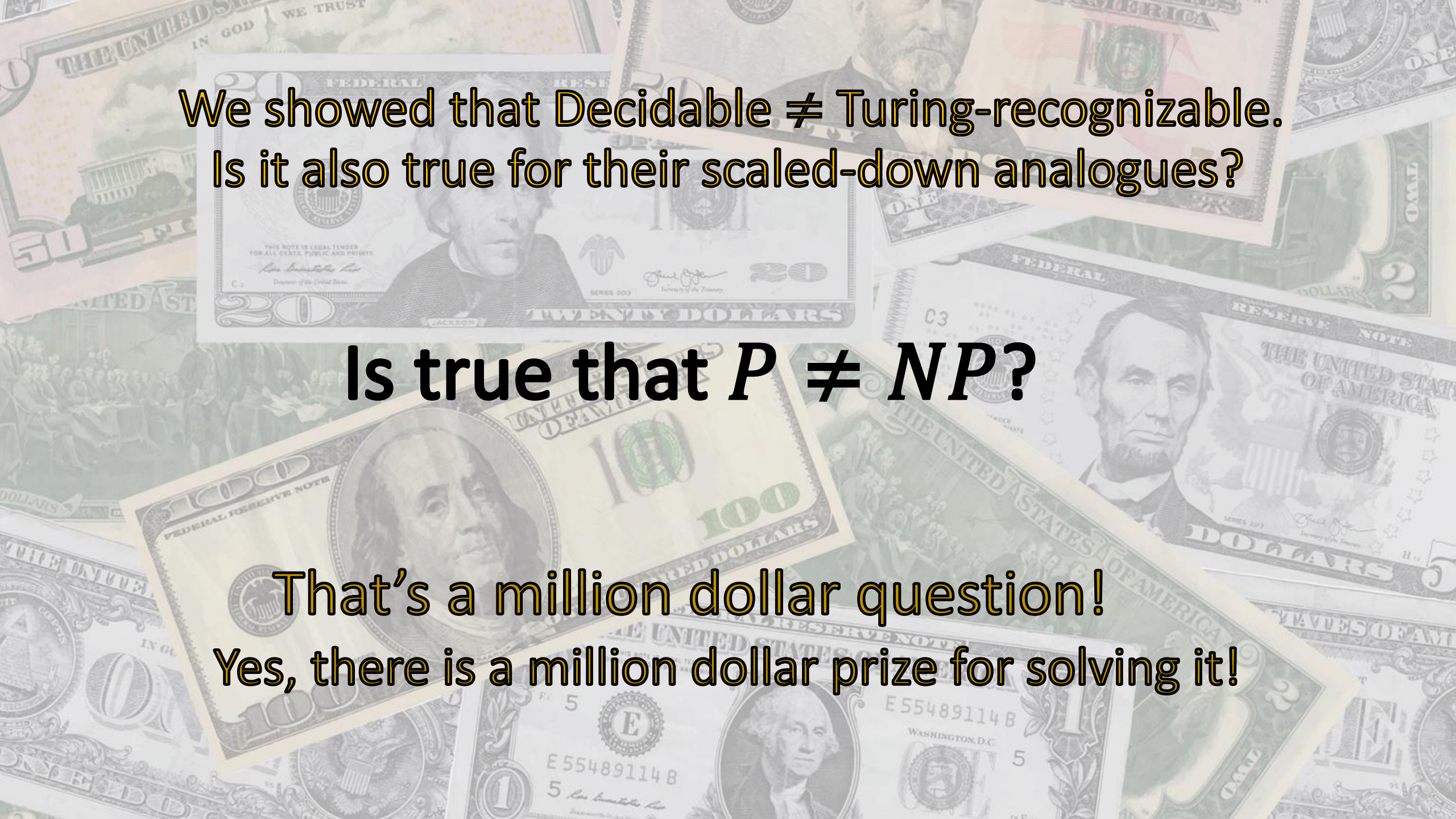


This is where the name **NP** comes from
("NP" stands for "Non-deterministic Polynomial-time")



Another definition of NP

- $NT_M(n) = \max_{x, |x|=n} \{\text{height of configuration tree of } M \text{ on } x\}$
- $NTime(f(n)) = \{L \mid \exists M \ \mathcal{L}(M) = L \text{ and } NT_M \in O(f)\}$
- Def 1: $L \in NP$ iff for some **polytime** decidable R , **some constants** c, d , $L = \{x \mid \exists y \ |y| \leq cn^d \wedge R(x, y)\}$
- Def 2: $NP = \bigcup_{d \in \mathbb{N}} NTime(n^d)$
- **Theorem:** Def 1 and Def 2 indeed define the same class of languages.
 - $\Leftarrow$ Guess which transitions to choose as a certificate y
 - $\Rightarrow$ Create two transitions for every time R would read a bit of y .

The background of the slide is a collage of various US dollar bills, including \$1, \$2, \$5, \$20, \$50, and \$100 bills, scattered and overlapping. The bills are faded and serve as a decorative backdrop for the text.

We showed that Decidable $\neq$ Turing-recognizable.
Is it also true for their scaled-down analogues?

Is true that $P \neq NP$?

That's a million dollar question!
Yes, there is a million dollar prize for solving it!

[ABOUT](#)[PROGRAMS](#)[MILLENNIUM PROBLEMS](#)[PEOPLE](#)[PUBLICATIONS](#)[EVENTS](#)[EUCLID](#)

The Millennium Prize Problems

In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) established seven *Prize Problems*. The Prizes were conceived to record some of the most difficult problems with which mathematicians were grappling at the turn of the second millennium; to elevate in the consciousness of the general public the fact that in mathematics, the frontier is still open and abounds in important unsolved problems; to emphasize the importance of working towards a solution of the deepest, most difficult problems; and to recognize achievement in mathematics of historical magnitude.

The prizes were announced at a meeting in Paris, held on May 24, 2000 at the Collège de France. Three lectures were presented: Timothy Gowers spoke on *The Importance of Mathematics*; Michael Atiyah and John Tate spoke on the problems themselves.

The seven Millennium Prize Problems were chosen by the founding Scientific Advisory Board of CMI, which conferred with leading experts worldwide. The focus of the board was on important classic questions that have resisted solution for many years.

Following the decision of the Scientific Advisory Board, the Board of Directors of CMI designated a \$7 million prize fund for the solutions to these problems, with \$1 million allocated to the solution of each problem.

See also:

[2000 Millennium Event](#)[Rules for the Millennium Prizes](#)

Millennium Problems

Yang–Mills and Mass Gap

Experiment and computer simulations suggest the existence of a "mass gap" in the solution to the quantum versions of the Yang-Mills equations. But no proof of this property is known.

Riemann Hypothesis

The prime number theorem determines the average distribution of the primes. The Riemann hypothesis tells us about the deviation from the average. Formulated in Riemann's 1859 paper, it asserts that all the 'non-obvious' zeros of the zeta function are complex numbers with real part $1/2$.

P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit, how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily find a solution.

Navier–Stokes Equation

This is the equation which governs the flow of fluids such as water and air. However, there is no proof for the most basic questions one can ask: do solutions exist, and are they unique? Why ask for a proof? Because a proof gives not only certitude, but also understanding.

Hodge Conjecture