The Number of Witnesses to Compositeness

Theorem 71 (Solovay and Strassen (1977)) *If N is an odd composite, then $(M|N) \neq M^{(N-1)/2} \bmod N$ for at least half of $M \in \Phi(N)$.*

- By Lemma 70 (p. 501) there is at least one $a \in \Phi(N)$ such that $(a|N) \neq a^{(N-1)/2} \bmod N$.
- Let $B = \{b_1, b_2, \ldots, b_k\} \subseteq \Phi(N)$ be the set of *all* distinct residues such that $(b_i|N) = b_i^{(N-1)/2} \bmod N$.
- Let $aB = \{ab_i \bmod N : i = 1, 2, \ldots, k\}$.

The Proof (concluded)

- $|aB| = k$.
  - $ab_i = ab_j \bmod N$ implies $N|a(b_i - b_j)$, which is impossible because $\gcd(a, N) = 1$ and $N > |b_i - b_j|$.
- $aB \cap B = \emptyset$ because

$$(ab_i)^{(N-1)/2} = a^{(N-1)/2} b_i^{(N-1)/2} \neq (a|N)(b_i|N) = (ab_i|N).$$

- Combining the above two results, we know

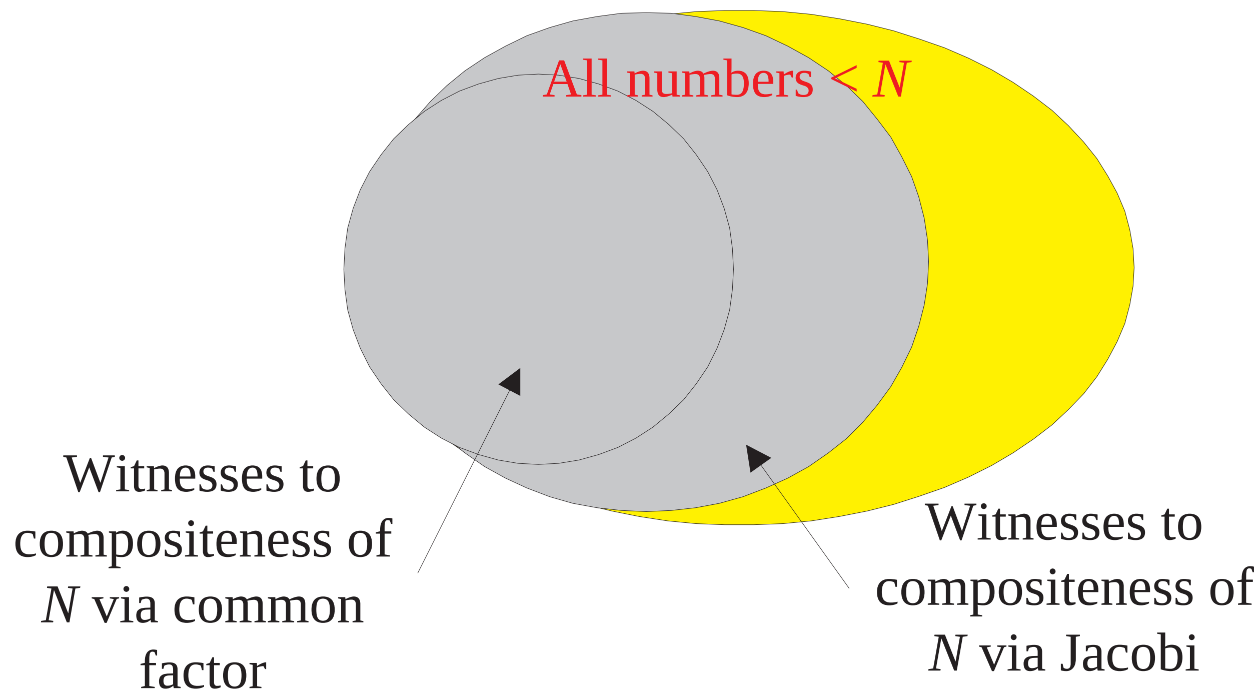
$$\frac{|\,B\,|}{\phi(N)} \leq \frac{|\,B\,|}{|\,B \cup aB\,|} = 0.5.$$

if N is even but $N \neq 2$ **then**
    return "N is composite";
else if $N = 2$ **then**
    return "N is a prime";
end if
Pick $M \in \{2, 3, \ldots, N-1\}$ randomly;
if $\gcd(M, N) > 1$ **then**
    return "N is a composite";
else
    if $(M|N) \neq M^{(N-1)/2} \bmod N$ **then**
        return "N is composite";
    else
        return "N is a prime";
    end if
end if

Analysis

- The algorithm certainly runs in polynomial time.
- There are no false positives (for COMPOSITENESS).
  - When the algorithm says the number is composite, it is always correct.
- The probability of a false negative is at most one half.
  - If the input is composite, then the probability that the algorithm says the number is a prime is ≤ 0.5.
- So it is a Monte Carlo algorithm for COMPOSITENESS.

The Improved Density Attack for COMPOSITENESS



Randomized Complexity Classes; RP

- Let N be a polynomial-time precise NTM that runs in time $p(n)$ and has 2 nondeterministic choices at each step.
- N is a **polynomial Monte Carlo Turing machine** for a language L if the following conditions hold:
  - If $x \in L$, then at least half of the $2^{p(n)}$ computation paths of N on x halt with "yes" where $n = |\, x \,|$.
  - If $x \notin L$, then all computation paths halt with "no."
- The class of all languages with polynomial Monte Carlo TMs is denoted **RP** (**randomized polynomial time**).[a]

[a] Adleman and Manders (1977).

Comments on RP

- Nondeterministic steps can be seen as fair coin flips.
- There are no false positive answers.
- The probability of false negatives, $1 - \epsilon$, is at most 0.5.
- But any constant between 0 and 1 can replace 0.5.
  - By repeating the algorithm $k = \lceil -\frac{1}{\log_2 1-\epsilon} \rceil$ times, the probability of false negatives becomes $(1 - \epsilon)^k \leq 0.5$.
- In fact, ϵ can be arbitrarily close to 0 as long as it is of the order $1/p(n)$ for some polynomial $p(n)$.
  - $-\frac{1}{\log_2 1-\epsilon} = O(\frac{1}{\epsilon}) = O(p(n))$.

Where RP Fits

- P $\subseteq$ RP $\subseteq$ NP.
  - A deterministic TM is like a Monte Carlo TM except that all the coin flips are ignored.
  - A Monte Carlo TM is an NTM with extra demands on the number of accepting paths.
- COMPOSITENESS $\in$ RP; PRIMES $\in$ coRP; PRIMES $\in$ RP.[a]
  - In fact, PRIMES $\in$ P.[b]
- RP $\cup$ coRP is another "plausible" notion of efficient computation.

---

[a] Adleman and Huang (1987).
[b] Agrawal, Kayal, and Saxena (2002).

ZPP[a] (Zero Probabilistic Polynomial)

- The class **ZPP** is defined as RP $\cap$ coRP.
- A language in ZPP has *two* Monte Carlo algorithms, one with no false positives and the other with no false negatives.
- If we repeatedly run both Monte Carlo algorithms, *eventually* one definite answer will come (unlike RP).
  - A *positive* answer from the one without false positives.
  - A *negative* answer from the one without false negatives.

[a] Gill (1977).

The ZPP Algorithm (**Las Vegas**)

{Suppose $L \in \text{ZPP}$.}
{N_1 has no false positives, and N_2 has no false negatives.}
while `true` **do**
    if $N_1(x) =$ "yes" **then**
        return "yes";
    end if
    if $N_2(x) =$ "no" **then**
        return "no";
    end if
end while

ZPP (concluded)

- The *expected* running time for the correct answer to emerge is polynomial.
  - The probability that a run of the 2 algorithms does not generate a definite answer is 0.5 (why?).
  - Let $p(n)$ be the running time of each run of the while-loop.
  - The expected running time for a definite answer is

$$\sum_{i=1}^{\infty} 0.5^i i p(n) = 2p(n).$$

- Essentially, ZPP is the class of problems that can be solved without errors in expected polynomial time.

Et Tu, RP?

```
{Suppose L ∈ RP.}
{N decides L without false positives.}
while true do
   if N(x) = "yes" then
      return "yes";
   end if
   {But what to do here?}
end while
```

- You eventually get a "yes" if $x \in L$.
- But how to get a "no" when $x \notin L$?
- You have to sacrifice either correctness or bounded running time.

Large Deviations

- Suppose you have a *biased* coin.
- One side has probability $0.5 + \epsilon$ to appear and the other $0.5 - \epsilon$, for some $0 < \epsilon < 0.5$.
- But you do not know which is which.
- How to decide which side is the more likely—with high confidence?
- Answer: Flip the coin many times and pick the side that appeared the most times.
- Question: Can you quantify the confidence?

The Chernoff Bound[a]

Theorem 72 (Chernoff (1952)) *Suppose* $x_1, x_2, \ldots, x_n$ *are independent random variables taking the values 1 and 0 with probabilities* p *and* $1 - p$*, respectively. Let* $X = \sum_{i=1}^n x_i$*. Then for all* $0 \leq \theta \leq 1$*,*

$$\text{prob}[\, X \geq (1 + \theta)\, pn \,] \leq e^{-\theta^2 pn/3}.$$

- The probability that the deviate of a **binomial random variable** from its expected value $E[\, X \,] = E[\, \sum_{i=1}^n x_i \,] = pn$ decreases exponentially with the deviation.
- The Chernoff bound is asymptotically optimal.

[a]Herman Chernoff (1923–).

The Proof

- Let t be any positive real number.
- Then

$$\text{prob}[\, X \geq (1+\theta)\, pn \,] = \text{prob}[\, e^{tX} \geq e^{t(1+\theta)\, pn} \,].$$

- Markov's inequality (p. 462) generalized to real-valued random variables says that

$$\text{prob}\left[\, e^{tX} \geq kE[\, e^{tX} \,] \right] \leq 1/k.$$

- With $k = e^{t(1+\theta)\, pn} / E[\, e^{tX} \,]$, we have

$$\text{prob}[\, X \geq (1+\theta)\, pn \,] \leq e^{-t(1+\theta)\, pn} E[\, e^{tX} \,].$$

The Proof (continued)

- Because $X = \sum_{i=1}^{n} x_i$ and x_i's are independent,

$$E[\, e^{tX} \,] = (E[\, e^{tx_1} \,])^n = [\, 1 + p(e^t - 1) \,]^n.$$

- Substituting, we obtain

$$\begin{aligned} \text{prob}[\, X \geq (1+\theta)\, pn \,] &\leq e^{-t(1+\theta)\, pn} [\, 1 + p(e^t - 1) \,]^n \\ &\leq e^{-t(1+\theta)\, pn} e^{pn(e^t - 1)} \end{aligned}$$

as $(1+a)^n \leq e^{an}$ for all $a > 0$.

The Proof (concluded)

- With the choice of $t = \ln(1+\theta)$, the above becomes

$$\text{prob}[\,X \geq (1+\theta)\,pn\,] \leq e^{pn[\,\theta-(1+\theta)\ln(1+\theta)\,]}.$$

- The exponent expands to $-\frac{\theta^2}{2} + \frac{\theta^3}{6} - \frac{\theta^4}{12} + \cdots$ for $0 \leq \theta \leq 1$, which is less than

$$-\frac{\theta^2}{2} + \frac{\theta^3}{6} \leq \theta^2 \left(-\frac{1}{2} + \frac{\theta}{6} \right) \leq \theta^2 \left(-\frac{1}{2} + \frac{1}{6} \right) = -\frac{\theta^2}{3}.$$

Power of the Majority Rule

From $\text{prob}[\, X \leq (1-\theta)\, pn\,] \leq e^{-\frac{\theta^2}{2} pn}$ (prove it):

Corollary 73 *If* $p = (1/2) + \epsilon$ *for some* $0 \leq \epsilon \leq 1/2$, *then*

$$\text{prob}\left[\sum_{i=1}^{n} x_i \leq n/2\right] \leq e^{-\epsilon^2 n/2}.$$

- The textbook's corollary to Lemma 11.9 seems incorrect.
- Our original problem (p. 520) hence demands $\approx 1.4k/\epsilon^2$ independent coin flips to guarantee making an error with probability at most 2^{-k} with the majority rule.

BPP[a] (Bounded Probabilistic Polynomial)

- The class **BPP** contains all languages for which there is a precise polynomial-time NTM N such that:
  - If $x \in L$, then at least $3/4$ of the computation paths of N on x lead to "yes."
  - If $x \notin L$, then at least $3/4$ of the computation paths of N on x lead to "no."
- N accepts or rejects by a *clear* majority.

---

[a] Gill (1977).

Magic 3/4?

- The number 3/4 bounds the probability of a right answer away from 1/2.
- Any constant *strictly* between 1/2 and 1 can be used without affecting the class BPP.
- In fact, 0.5 plus any inverse polynomial between 1/2 and 1,
$$0.5 + \frac{1}{p(n)},$$
can be used.

The Majority Vote Algorithm

Suppose L is decided by N by majority $(1/2) + \epsilon$.

for $i = 1, 2, \ldots, 2k + 1$ **do**
    Run N on input x;
end for
if "yes" is the majority answer **then**
    "yes";
else
    "no";
end if

Analysis

- The running time remains polynomial, being $2k + 1$ times N's running time.
- By Corollary 73 (p. 525), the probability of a false answer is at most $e^{-\epsilon^2 k}$.
- By taking $k = \lceil 2/\epsilon^2 \rceil$, the error probability is at most $1/4$.
- As with the RP case, ϵ can be any inverse polynomial, because k remains polynomial in n.

Probability Amplification for BPP

- Let m be the number of random bits used by a BPP algorithm.
  - By definition, m is polynomial in n.
- With $k = \Theta(\log m)$ in the majority vote algorithm, we can lower the error probability to, say,

$$\leq (3m)^{-1}.$$

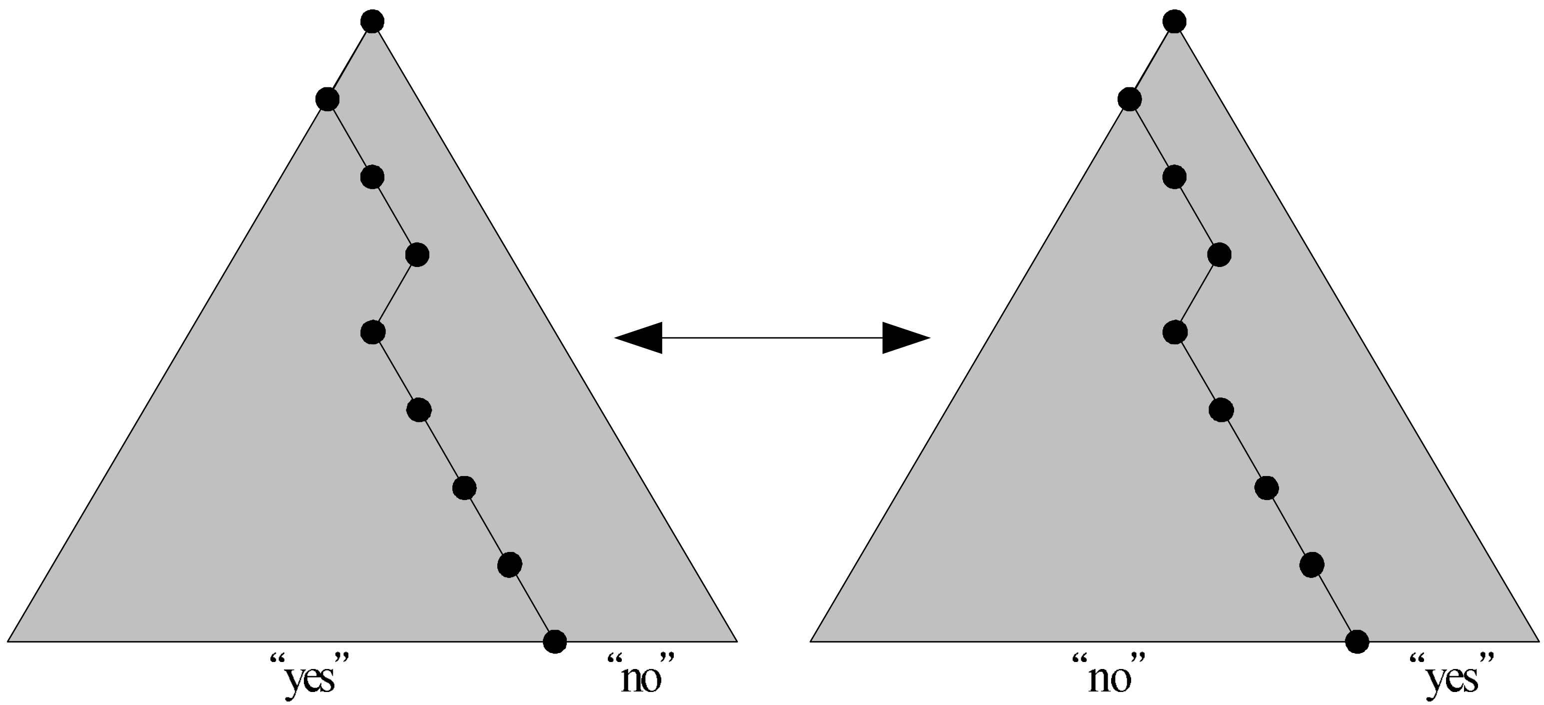
Aspects of BPP

- BPP is the most comprehensive yet plausible notion of efficient computation.
  - If a problem is in BPP, we take it to mean that the problem can be solved efficiently.
  - In this aspect, BPP has effectively replaced P.
- $(\text{RP} \cup \text{coRP}) \subseteq (\text{NP} \cup \text{coNP})$.
- $(\text{RP} \cup \text{coRP}) \subseteq \text{BPP}$.
- Whether $\text{BPP} \subseteq (\text{NP} \cup \text{coNP})$ is unknown.
- But it is unlikely that $\text{NP} \subseteq \text{BPP}$ (p. 546).

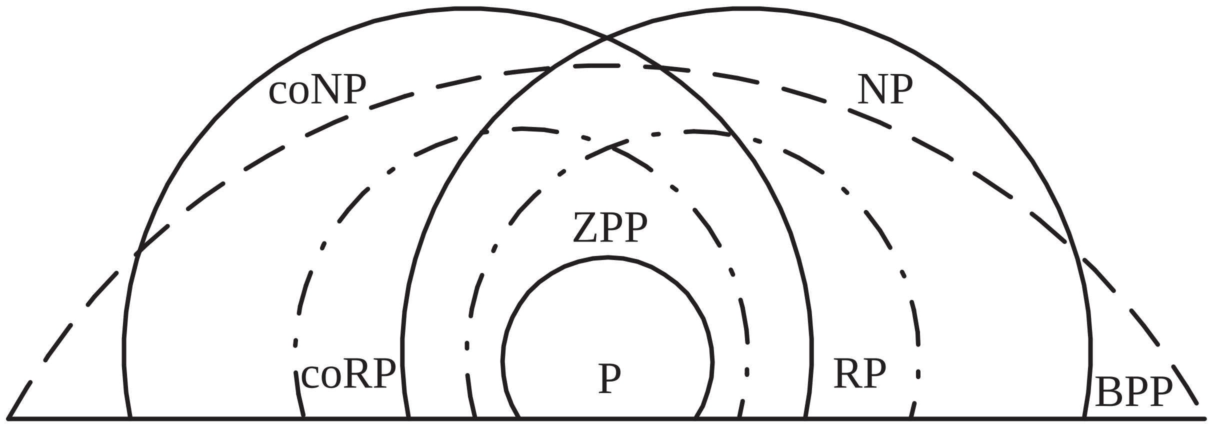
coBPP

- The definition of BPP is symmetric: acceptance by clear majority and rejection by clear majority.
- An algorithm for $L \in$ BPP becomes one for $\bar{L}$ by reversing the answer.
- So $\bar{L} \in$ BPP and BPP $\subseteq$ coBPP.
- Similarly coBPP $\subseteq$ BPP.
- Hence BPP $=$ coBPP.
- This approach does not work for RP.
- It did not work for NP either.

BPP and coBPP



"The Good, the Bad, and the Ugly"



Circuit Complexity

- Circuit complexity is based on boolean circuits instead of Turing machines.
- A boolean circuit with n inputs computes a boolean function of n variables.
- By identify `true` with 1 and `false` with 0, a boolean circuit with n inputs accepts certain strings in $\{0, 1\}^n$.
- To relate circuits with arbitrary languages, we need one circuit for each possible input length n.

Formal Definitions

- The **size** of a circuit is the number of *gates* in it.
- A **family of circuits** is an infinite sequence $\mathcal{C} = (C_0, C_1, \ldots)$ of boolean circuits, where C_n has n boolean inputs.
- $L \subseteq \{0,1\}^*$ has **polynomial circuits** if there is a family of circuits $\mathcal{C}$ such that:
  - The size of C_n is at most $p(n)$ for some fixed polynomial p.
  - For input $x \in \{0,1\}^*$, $C_{|\,x\,|}$ outputs 1 if and only if $x \in L$.
    * C_n accepts $L \cap \{0,1\}^n$.

Exponential Circuits Contain All Languages

- Theorem 14 (p. 164) implies that there are languages that cannot be solved by circuits of size $2^n/(2n)$.
- But exponential circuits can solve all problems.

Proposition 74 *All decision problems (decidable or otherwise) can be solved by a circuit of size* 2^{n+2}*.*

- We will show that for any language $L \subseteq \{0,1\}^*$, $L \cap \{0,1\}^n$ can be decided by a circuit of size 2^{n+2}.

The Proof (concluded)

- Define boolean function $f : \{0,1\}^n \to \{0,1\}$, where

$$f(x_1 x_2 \cdots x_n) = \begin{cases} 1 & x_1 x_2 \cdots x_n \in L, \\ 0 & x_1 x_2 \cdots x_n \not\in L. \end{cases}$$

- $f(x_1 x_2 \cdots x_n) = (x_1 \wedge f(1 x_2 \cdots x_n)) \vee (\neg x_1 \wedge f(0 x_2 \cdots x_n))$.
- The circuit size $s(n)$ for $f(x_1 x_2 \cdots x_n)$ hence satisfies

$$s(n) = 4 + 2s(n-1)$$

  with $s(1) = 1$.
- Solve it to obtain $s(n) = 5 \times 2^{n-1} - 4 \leq 2^{n+2}$.

The Circuit Complexity of P

Proposition 75 *All languages in P have polynomial circuits.*

- Let $L \in \mathrm{P}$ be decided by a TM in time $p(n)$.
- By Corollary 31 (p. 255), there is a circuit with $O(p(n)^2)$ gates that accepts $L \cap \{0,1\}^n$.
- The size of the circuit depends only on L and the length of the input.
- The size of the circuit is polynomial in n.

Polynomial Circuits vs. P

- Is the converse of Proposition 75 true?
  - Do polynomial circuits accept only languages in P?
- They can accept *undecidable* languages!

Languages That Polynomial Circuits Accept (concluded)

- Let $L \subseteq \{0,1\}^*$ be an undecidable language.
- Let $U = \{1^n : \text{the binary expansion of } n \text{ is in } L\}$.[a]
  - For example, $11111_1 \in U$ if $101_2 \in L$.
- U is also undecidable.
- $U \cap \{1\}^n$ can be accepted by the trivial circuit C_n that outputs 1 if $1^n \in U$ and outputs 0 if $1^n \notin U$.
  - We may not know which is the case for *general* n.
- The family of circuits $(C_0, C_1, \ldots)$ is polynomial in size.

[a] Assume n's leading bit is always 1 without loss of generality.

A Patch

- Despite the simplicity of a circuit, the previous discussions imply the following:
  - Circuits are *not* a realistic model of computation.
  - Polynomial circuits are *not* a plausible notion of efficient computation.
- What is missing?
- The *effective and efficient constructibility* of

$$C_0, C_1, \ldots.$$

Uniformity

- A family $(C_0, C_1, \ldots)$ of circuits is **uniform** if there is a $\log n$-space bounded TM which on input 1^n outputs C_n.
  - Note that n is the length of the input to C_n.
  - Circuits now cannot accept undecidable languages (why?).
  - The circuit family on p. 541 is not constructible by a *single* Turing machine (algorithm).
- A language has **uniformly polynomial circuits** if there is a *uniform* family of polynomial circuits that decide it.

Uniformly Polynomial Circuits and P

Theorem 76 *$L \in P$ if and only if L has uniformly polynomial circuits.*

- One direction was proved in Proposition 75 (p. 539).
- Now suppose L has uniformly polynomial circuits.
- Decide $x \in L$ in polynomial time as follows:
  - Calculate $n = |\, x \,|$.
  - Generate C_n in $\log n$ space, hence polynomial time.
  - Evaluate the circuit with input x in polynomial time.
- Therefore $L \in$ P.

Relation to P vs. NP

- Theorem 76 implies that P $\neq$ NP if and only if NP-complete problems have no *uniformly* polynomial circuits.
- A stronger conjecture: NP-complete problems have no polynomial circuits, *uniformly or not*.
- The above is currently the preferred approach to proving the P $\neq$ NP conjecture—without success so far.