Pseudorandomness and Combinatorial Constructions

Luca Trevisan

U.C. Berkeley

Pseudorandomness and Combinatorial Constructions

(1)The unreasonable effectiveness of randomness
(probabilistic method, randomized algorithms)

(2)"Hardness versus randomness"
Theorems: if **<complexity conjecture>** then we can get most of (1) without randomness

(3) Connections
Proofs of (2) related to unconditional explicit constructions as in (1)

The Unreasonable Effectiveness of Randomness

In Combinatorics: **The Probabilistic Method**

In Computer Science: **Randomized Algorithms**

The Probabilistic Method

Idea:

prove that an object with a certain property must exist, because a random object has said property with positive probability

Erdos (1940s): Ramsey Graphs

- **Ramsey Theorem.**
  In every graph with n vertices there is either a clique of size $\geq \frac{1}{2} \log n$ or an independent set of size $\geq \frac{1}{2} \log n$.

Erdos (1940s): Ramsey Graphs

- **Ramsey Theorem.**
  In every graph with n vertices there is either a clique of size $\geq \frac{1}{2} \log n$ or an independent set of size $\geq \frac{1}{2} \log n$.

- Erdos: pick an n-vertex graph at random. With positive probability, size of all cliques and all independent sets $\leq 2\log n$

Erdos (1940s): Ramsey Graphs

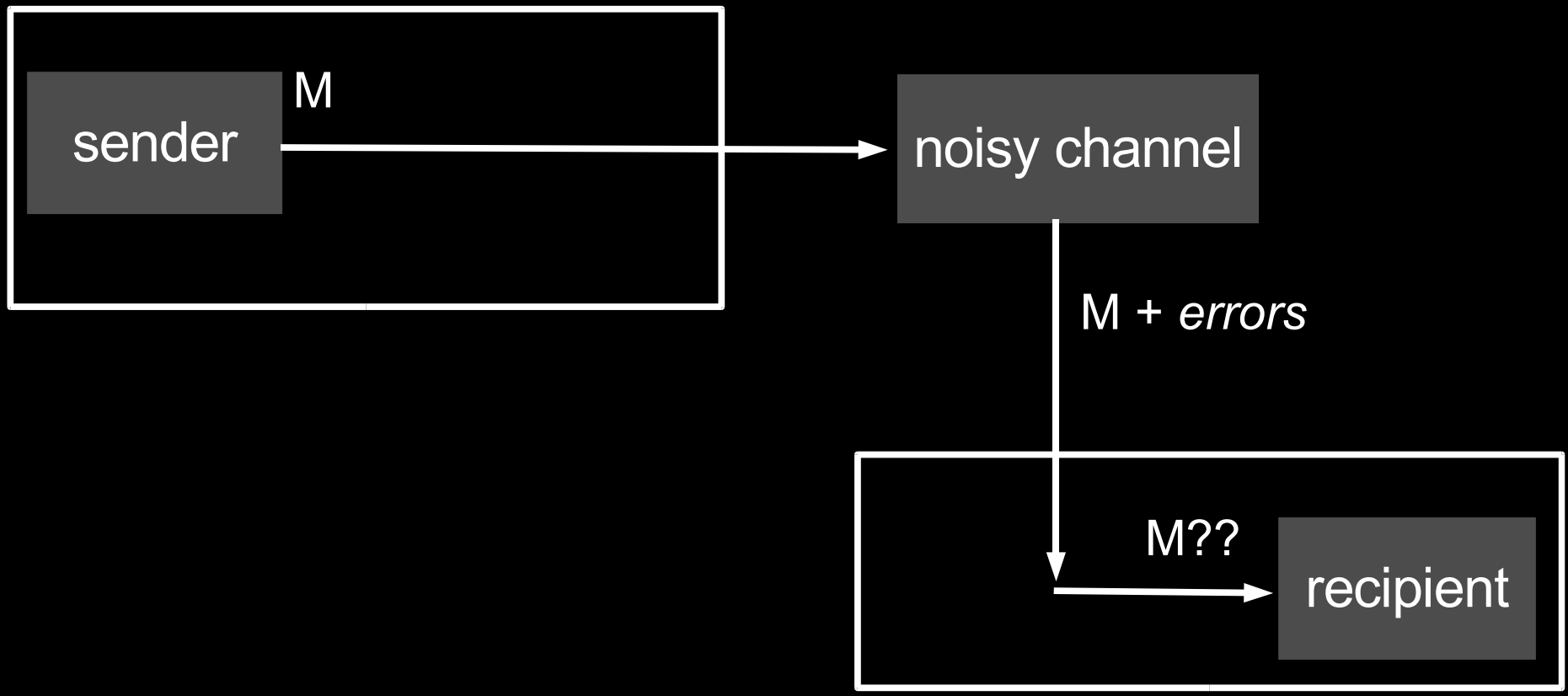
- Erdos: pick an *n*-vertex graph at random. With positive probability, size of all cliques and all independent sets $\leq$ 2log n

Explicit constructions:

- Frankl-Wilson (1981): $\exp((\log n)^{1/2})$
- Barak-Rao-Shaltiel-Wigderson ('06): $\exp((\log n)^{o(1)})$

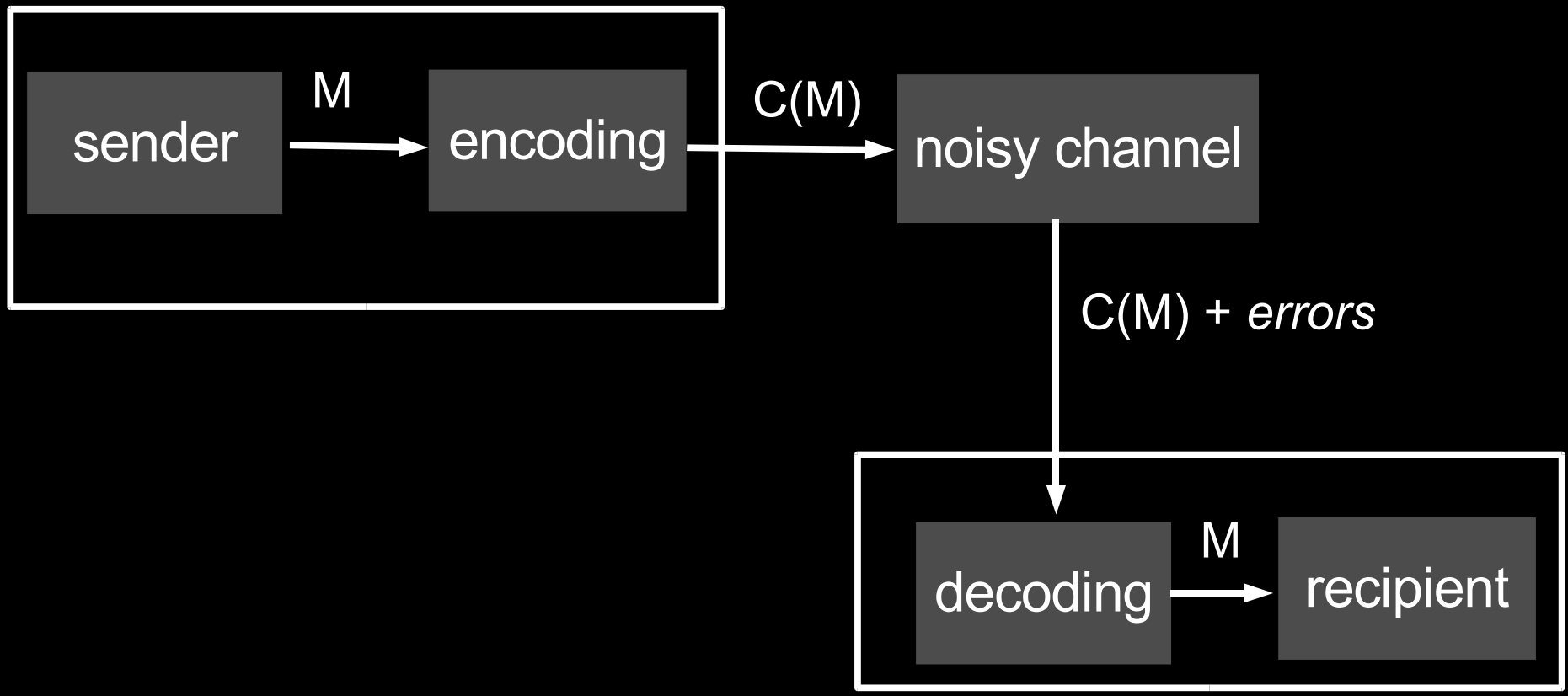
Shannon (1940s): Codes

Problem: reliable transmission over a noisy channel



Shannon (1940s): Codes

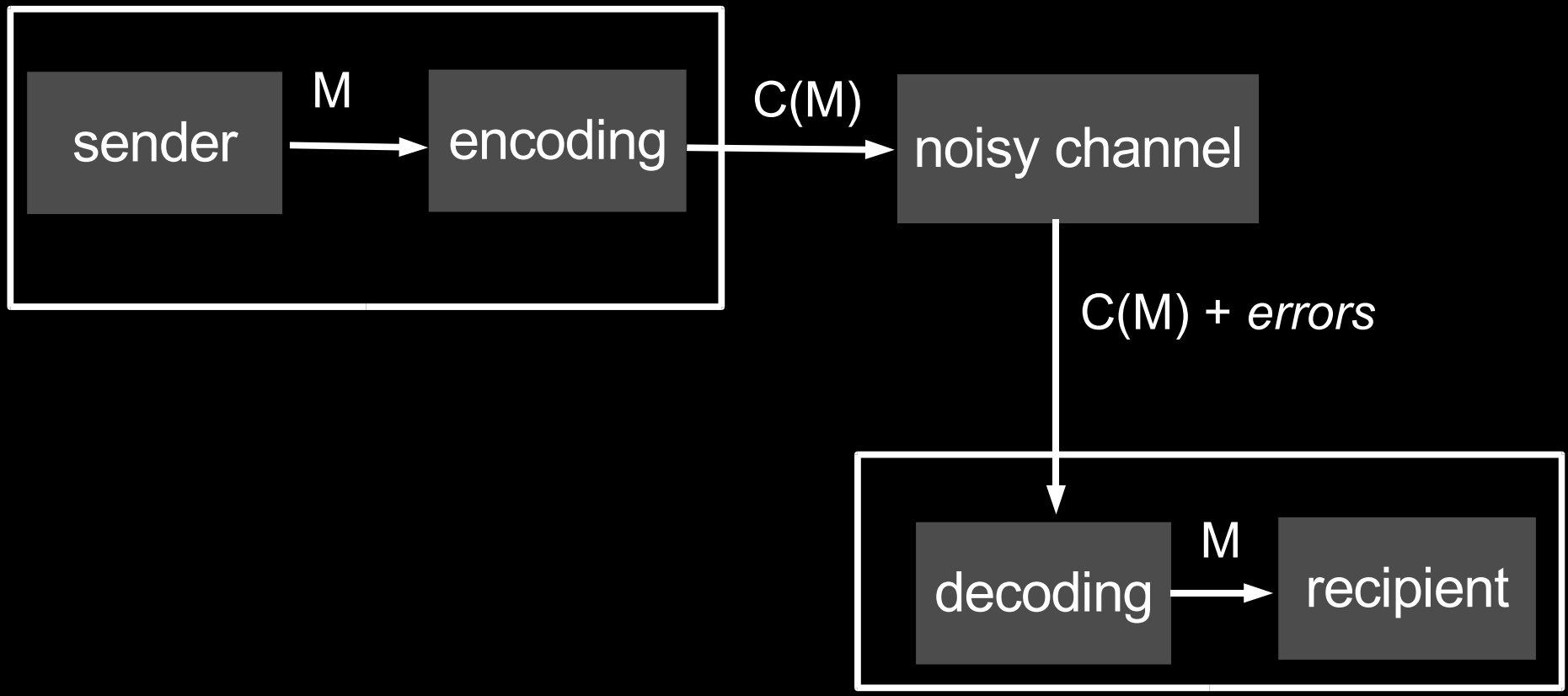
Problem: reliable transmission over a noisy channel



Shannon (1940s): Codes

Problem: reliable transmission over a noisy channel

Shannon: encode the message using a random function



Shannon (1940s): Codes

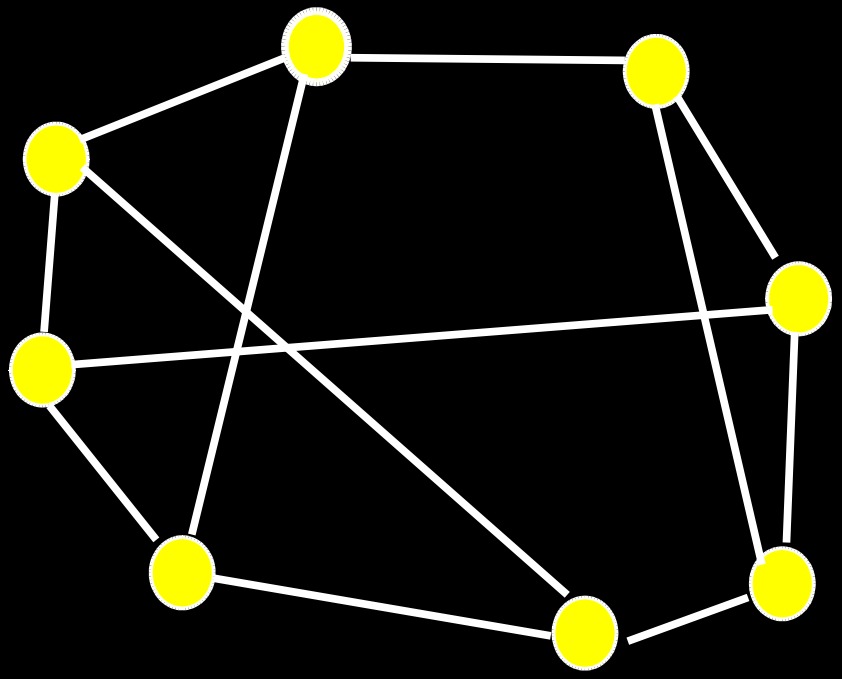
Problem: reliable transmission over a noisy channel

Shannon (1940s): encode the message using a random function

1990s: Non-constructive performance of Shannon's code matched by efficient encoding and decoding algorithms

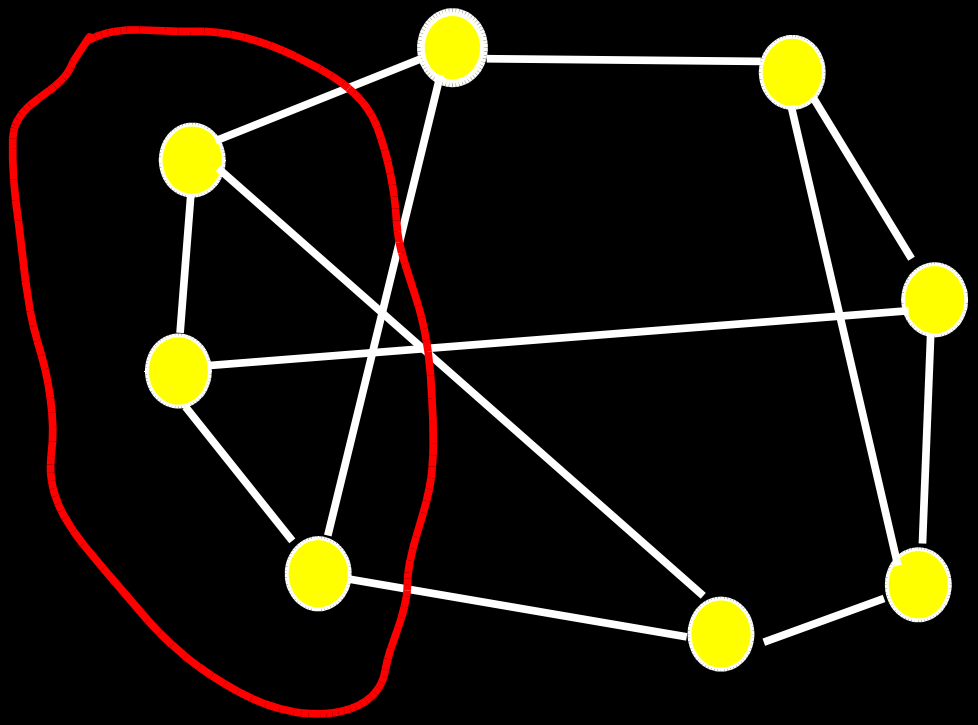
Pippenger Valiant (1970s): Expander Graphs

- Graphs such that every set of vertices has several outgoing edges



Pippenger Valiant (1970s): Expander Graphs

- Graphs such that every set of vertices has several outgoing edges



Pippenger Valiant (1970s): Expander Graphs

- Graphs such that every set of vertices has several outgoing edges

Explicit constructions:

- Lubotzky-Phillips-Sarnak (1986): simple construction, very difficult analysis
- Reingold-Vadhan-Wigderson (2000): more complex construction, simpler analysis

Probabilistic Method

Simple non-constructive existence proofs of:

- Ramsey graphs
- Error-control codes
- Expander graphs

Comparable constructive results took decades or are still open questions.

Probabilistic Algorithms

- Primality
  randomized alg [Miller Rabin Solovay Strassen 1970s]
  deterministic alg [Agarwal Kayal Saxena 2002]

- Connectivity in graphs using 'constant' memory
  random walk algorithm
  [Aleliunas Karp Lipton Lovász Rackoff '79]
  deterministic algorithm [Reingold 2005]

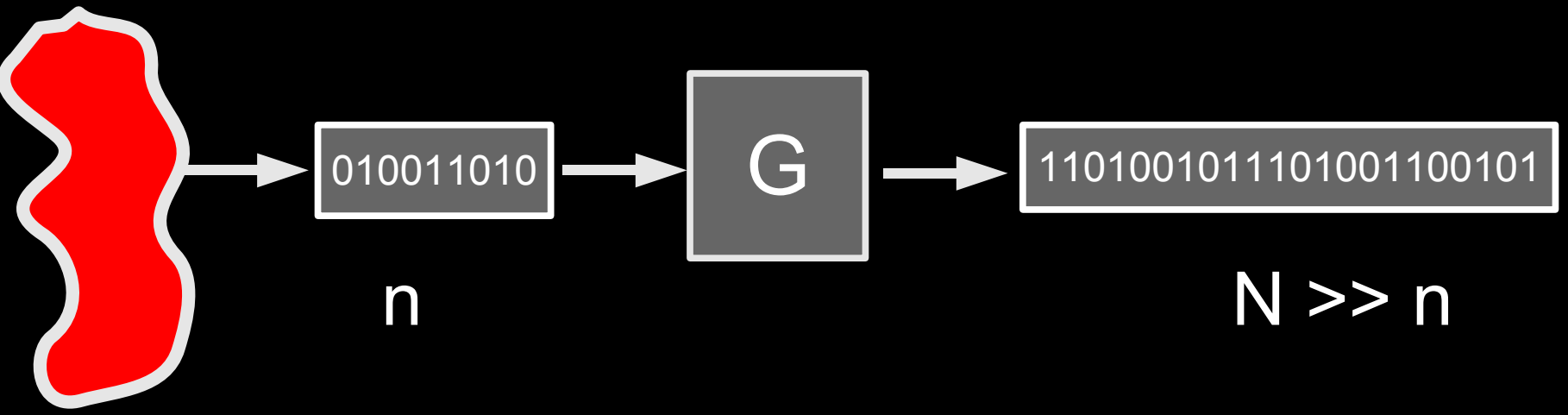
- Counting problems: e.g. permanent
  randomized [Jerrum Sinclair 80s; JS Vigoda 2001]
  deterministic algorithms ??

Hardness versus Randomness

Theorems of the form

- If [complexity-theoretic conjecture], then
  - every problem efficiently solvable with a randomized algorithm also "efficiently" solvable with a deterministic algorithm
  - a general class of existence proofs using probabilistic method automatically yields explicit constructions

Pseudorandom Generator



- Deterministic procedure
- Input: short sequence of random bits
- Output: long sequence of “random looking” bits

Desirable Properties

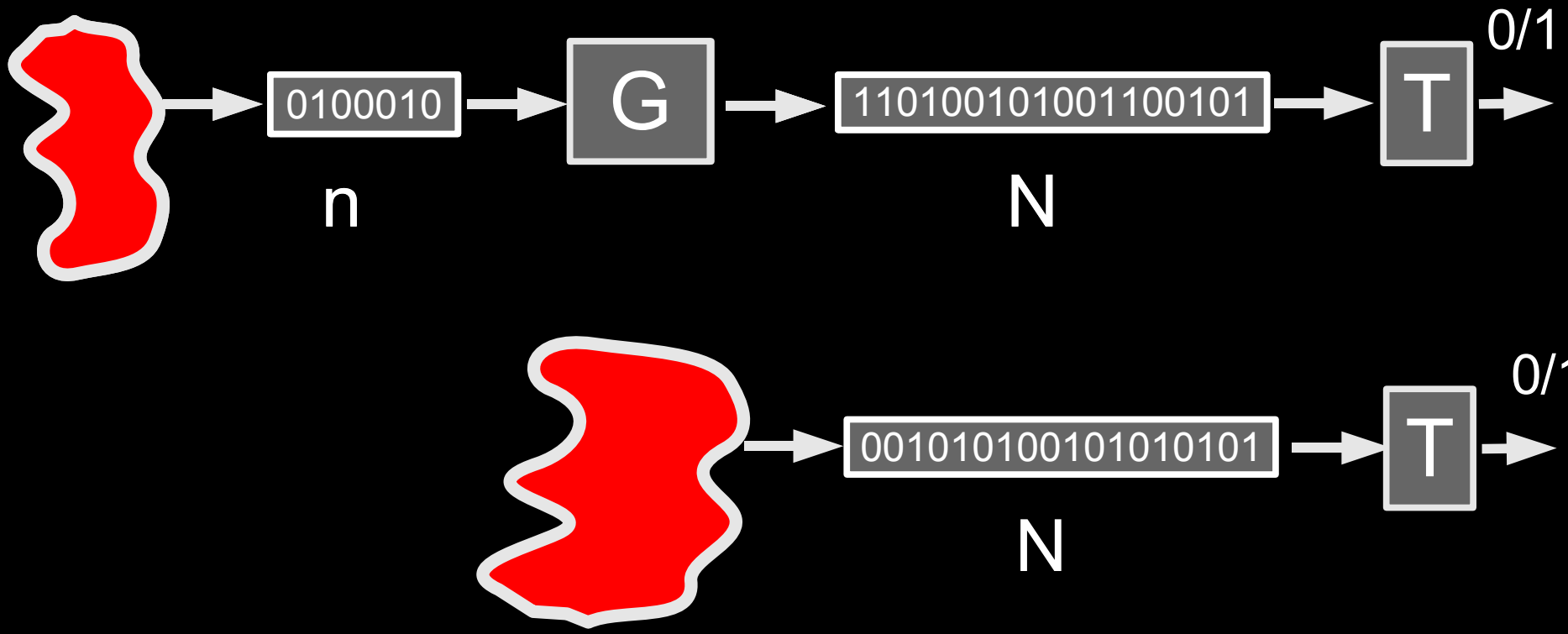
- If input to G is random then output has some statistical properties of uniform distribution:
  - On average ½ ones ½ zeros
  - Right variance
  - Right fraction of pairs, triples, . . .
  - Right typical length of runs of zeros and ones
  - . . .

- (suspend disbelief) Ideally all properties

All Properties

- For all statistical tests T

  $| \Pr[T(U_N) = 1] - \Pr[T(G(U_n)) = 1] | < .001$

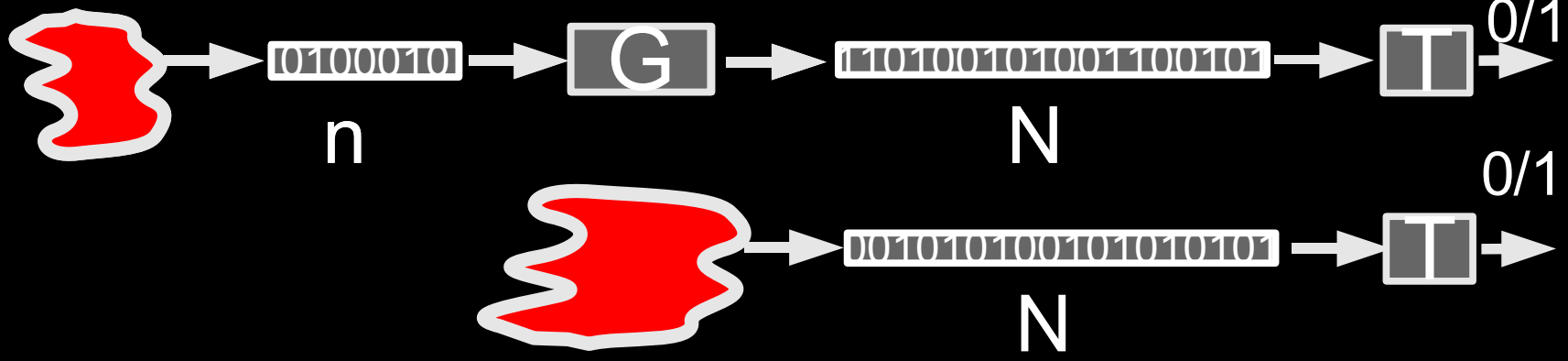


All Properties

- For all statistical tests T

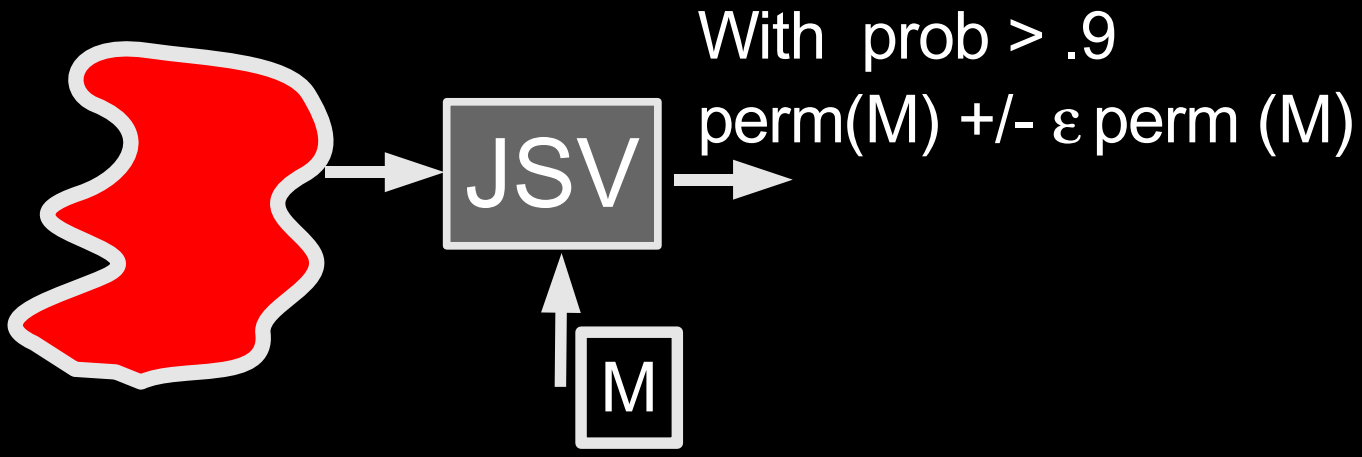
  $| \Pr[T(U_N) = 1] - \Pr[T(G(U_n)) = 1] | < .001$

- (disbelief still suspended) Suppose that G is computable in poly(N) time and n=O(log N) Call such G a *wonderful generator*



Using a Wonderful Generator

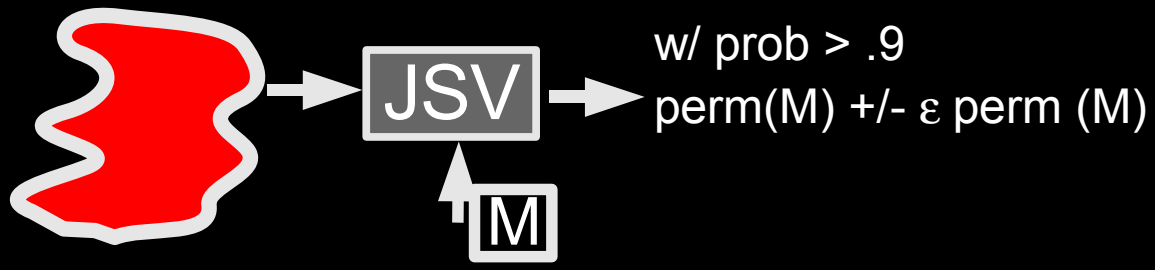
- Permanent approximation algorithm of Jerrum-Sinclair-Vigoda

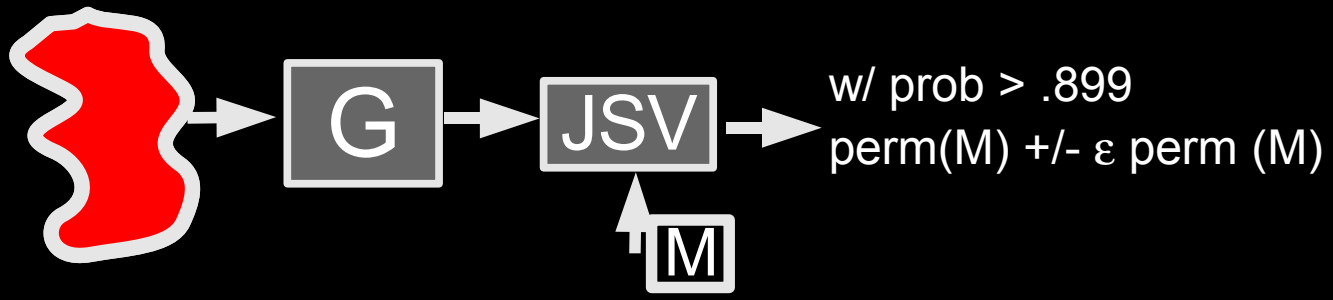


Note: M is a matrix, perm(M) is a number

Using a Wonderful Generator

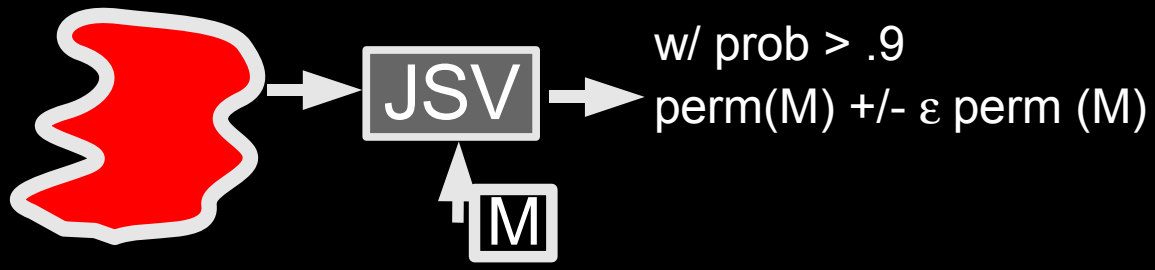
- Permanent approximation algorithm of Jerrum-Sinclair-Vigoda works nearly as well with G

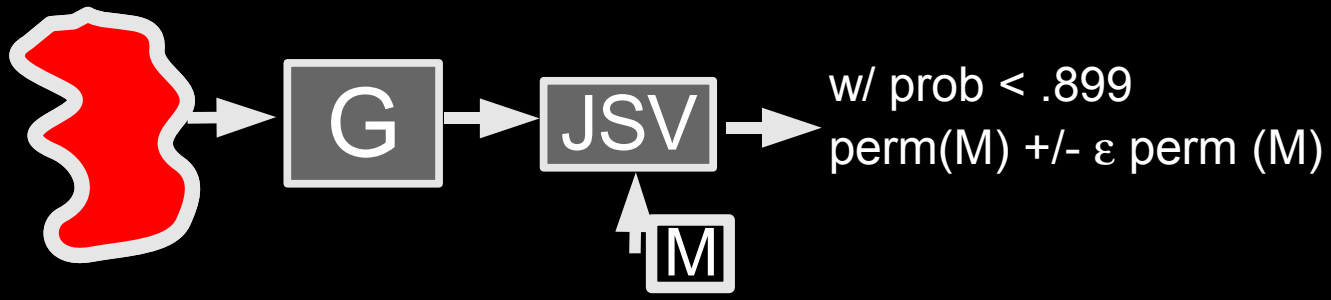




Proof

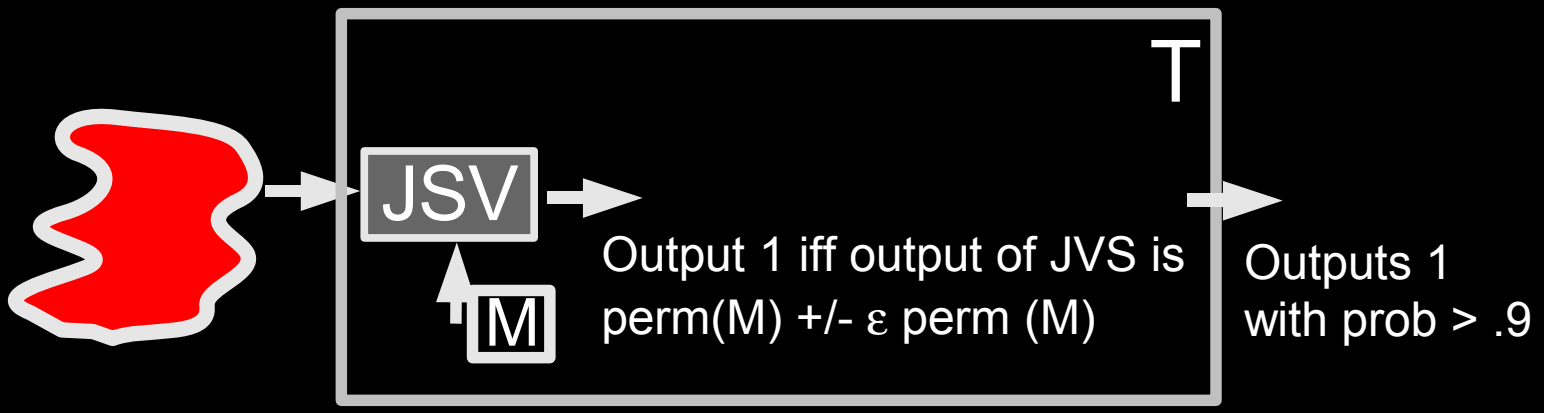
- Suppose not:

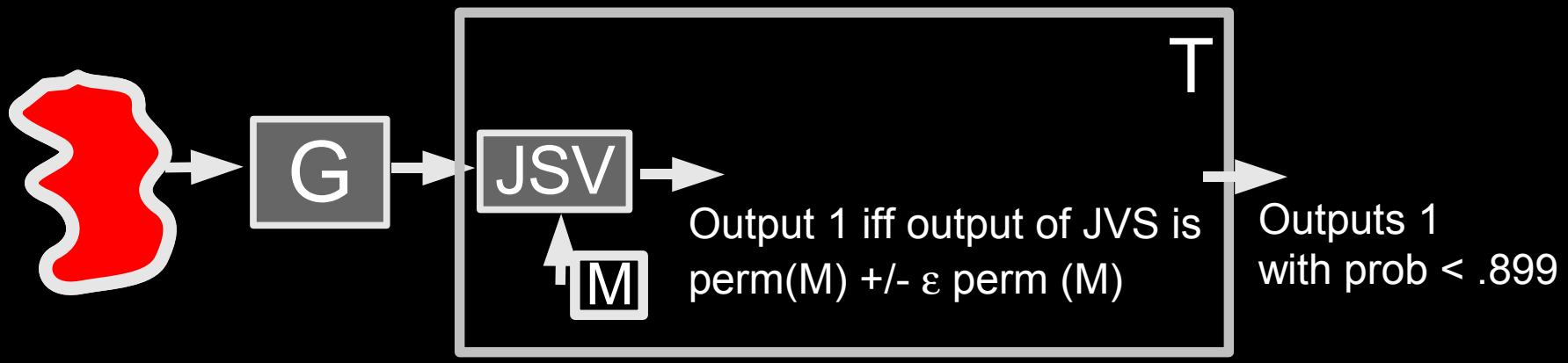




Proof

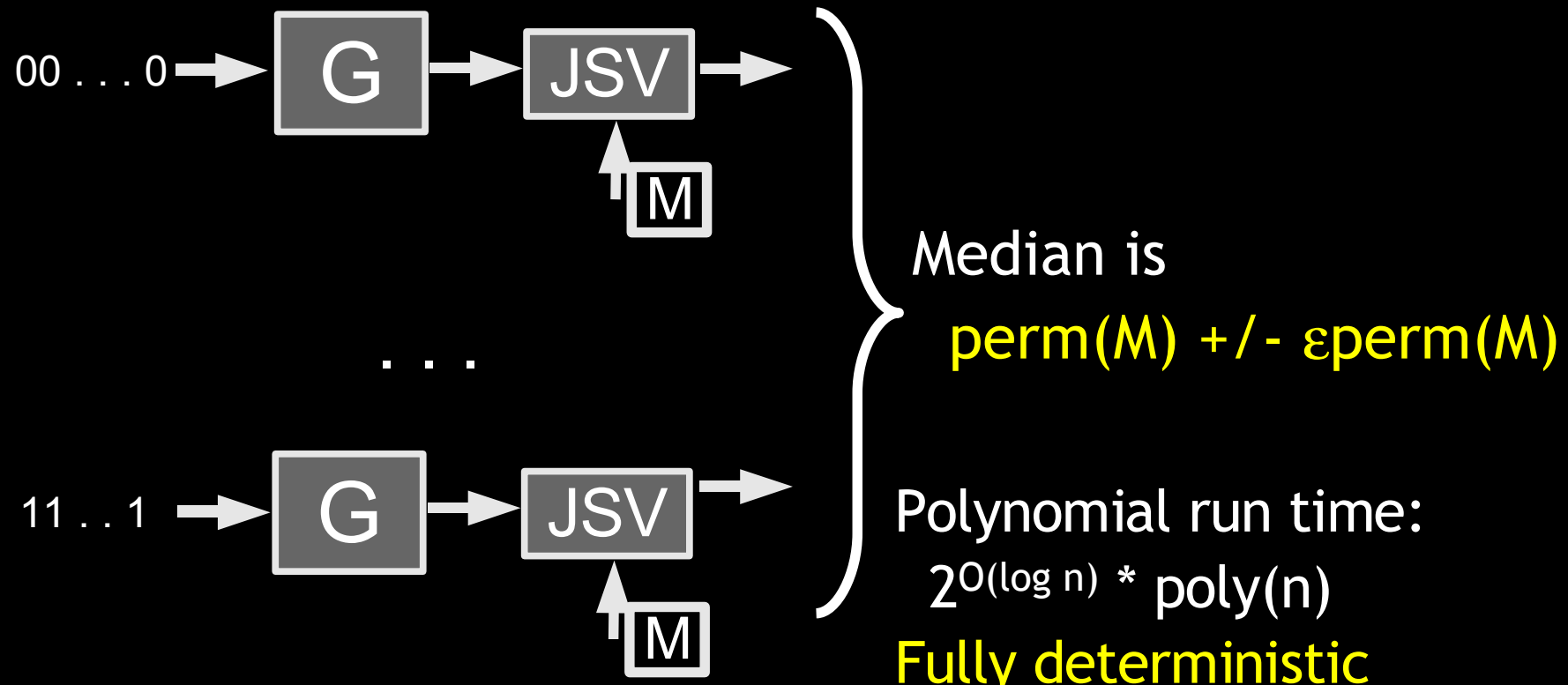
- Define property T contradicting G being generator:





Using a Wonderful Generator

- Now that we know that JVS algorithm works well with G, enumerate all possible inputs to G:



Generator for Algorithms

Given probabilistic algorithm A:

- Replace randomness of A with output of generator
- "Error probability" of A goes up by at most .01
  (Otherwise algorithm+input+definition of error are a statistical test that distinguishes G from U)
- Only O(log N) truly random bits are used
- Enumerate, then majority/maximum/median whichever is right

Generator for Constructions

- Suppose we prove the object O satisfying property P by showing
  Pr [P(O)] > ½
  when O is sampled from a”uniform distribution”

- Then one of the polynomially many possible outputs of generator must be an object with property O

But . . .

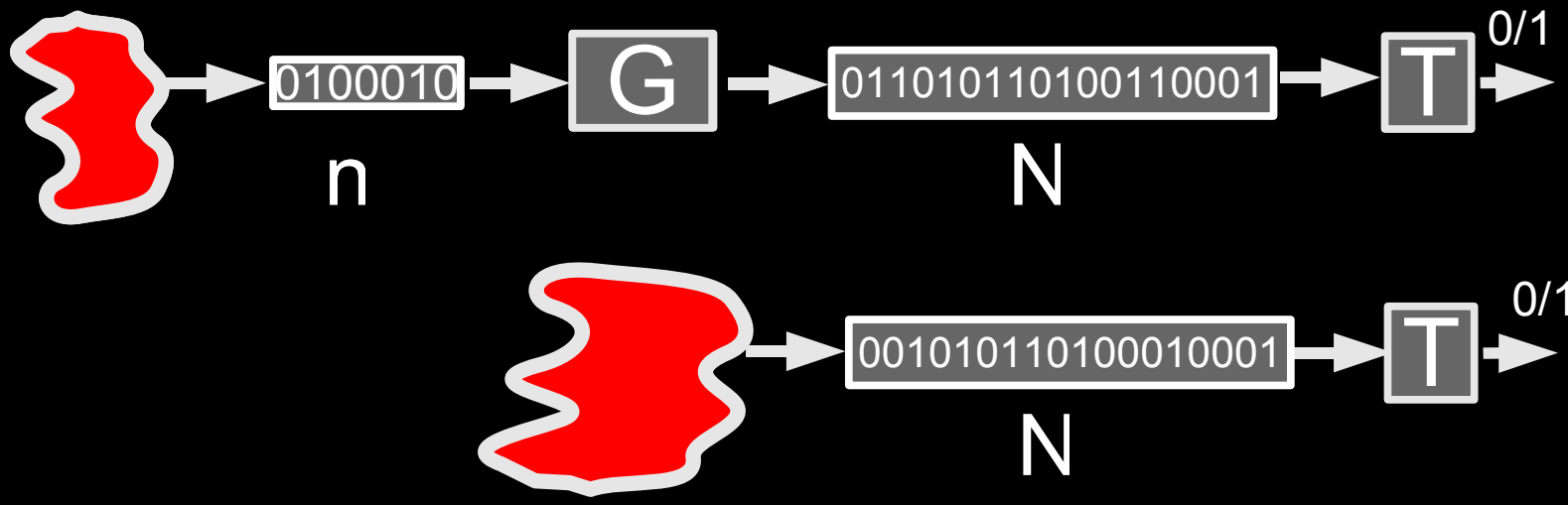
(suspension of disbelief is over)

wonderful generators do not exist

A Test That Always Fails

- We want: For all statistical tests T

  $| \Pr[T(U_N) = 1] - \Pr[T(G(U_n)) = 1] | < .001$



Define T(x)=1 iff x is possible output of G

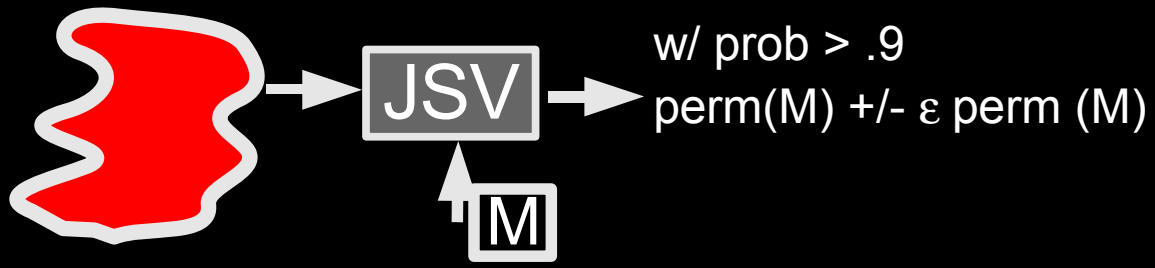
Almost as Good

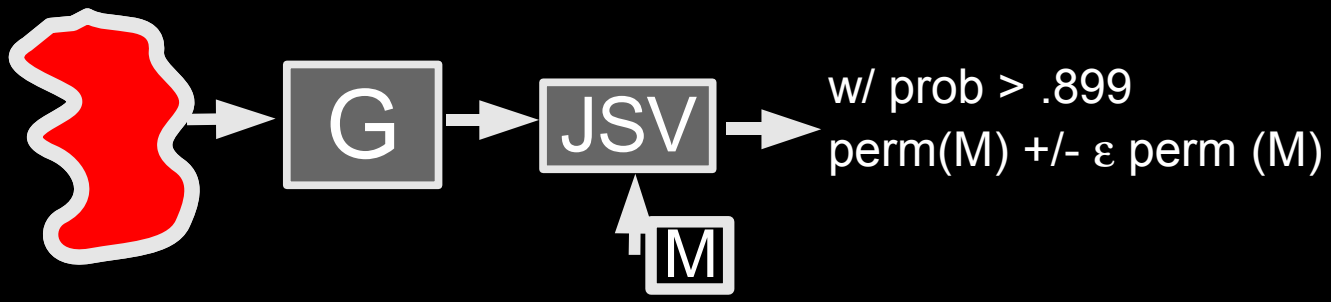
- In all application, suffices that no efficiently computable test T distinguishes G from Uniform
- Typically, T is algorithm+input+condition on output
- Define: G is (t,ε)–pseudorandom if
  for every T 'computable in time' t
  $$| \Pr [T(G(U_n))=1] - \Pr [T(U_N)=1] | \leq \varepsilon$$

e.g. N=1,000,000; n=64; t= 1 day; $\varepsilon = .01$
(More precisely, t is *circuit size* of T)

As Good

- If G is (t,.001)-pseudorandom, and in time t can run JVS algorithm and compare value with given range:





D-Generators

- A mapping $G^n \to G^N$ is a *derandomizing* generator if
  - the output is $(N^2, .01)$-pseudorandom
  - $n = O(\log N)$
  - computable in poly(N) time

- Thm: If a D-generator exists, then all randomized algorithms can be efficiently derandomized

If D-Generators exist . . .

- **P=BPP**
- Permanent and several counting problems have polynomial time deterministic approximation algorithms
- Given N, can find deterministically in poly(N) time a prime between 2^N and 2^{N+1}
- Given: probabilistic method existence proof of a checkable property;
  can get: explicit construction
  (assume noticeable probability, efficient samplability)

Conditional Existence

- [Blum-Micali Yao Goldreich-Levin 80s]

- Suppose there is a one-way permutation of exponential hardness
  - p_n: $\{0,1\}^n$ -> $\{0,1\}^n$ bijective
  - computable in poly(n) time
  - every algorithm of running time < $2^{\delta n}$ given $p_n(x)$, has prob < $2^{-\delta n}$ of finding x
- Then a D-generator exists

Nisan-Wigderson ('88) Theorem

- Suppose there is a decision problem that is
  - solvable in $2^{O(n)}$ time
  - but every circuit of size $< 2^{\delta n}$ is correct on $< \frac{1}{2} + 2^{-\delta n}$ fraction of inputs

  [NW Assumption]
- Then a D-generator exists
  (Map O(log N) bits to N bits, $(N^2, .01)$-pseudorandom)

Compare with BMY: same conclusion weaker assumption
(but BMY also stronger conclusion important for crypto)

Impagliazzo-Wigderson (1997)

A Theorem giving a reasonable condition under which D-generators exist:

- Suppose there is a decision problem that is
  - solvable in $2^{O(n)}$ time
  - not solvable by circuits of size $< 2^{\delta n}$
- Then [NW Assumption] is true (and a D-generator exists)

Proof of IW

- Assumption: a worst-case intractable problem
- Conclusion: the existence of an average-case intractable problem

- Original proof: 3 stages of “average-case hardness amplification”

- Alternative proof [Sudan Trevisan Vadhan 1999] error-correcting codes, sublinear time decoders

Worst-Case to Average-Case

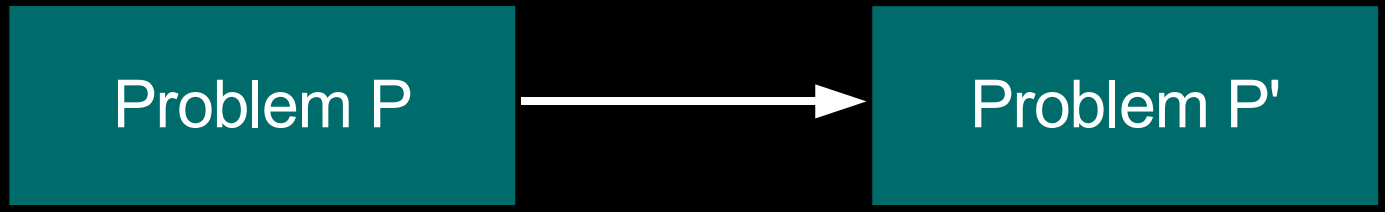
- Assumption: a worst-case intractable problem P
- Conclusion: the existence of an average-case intractable problem P'

Worst-Case to Average-Case

- Given P define P' such that
- If P' has good-on-average algorithm
- Then P has algorithm good on all inputs

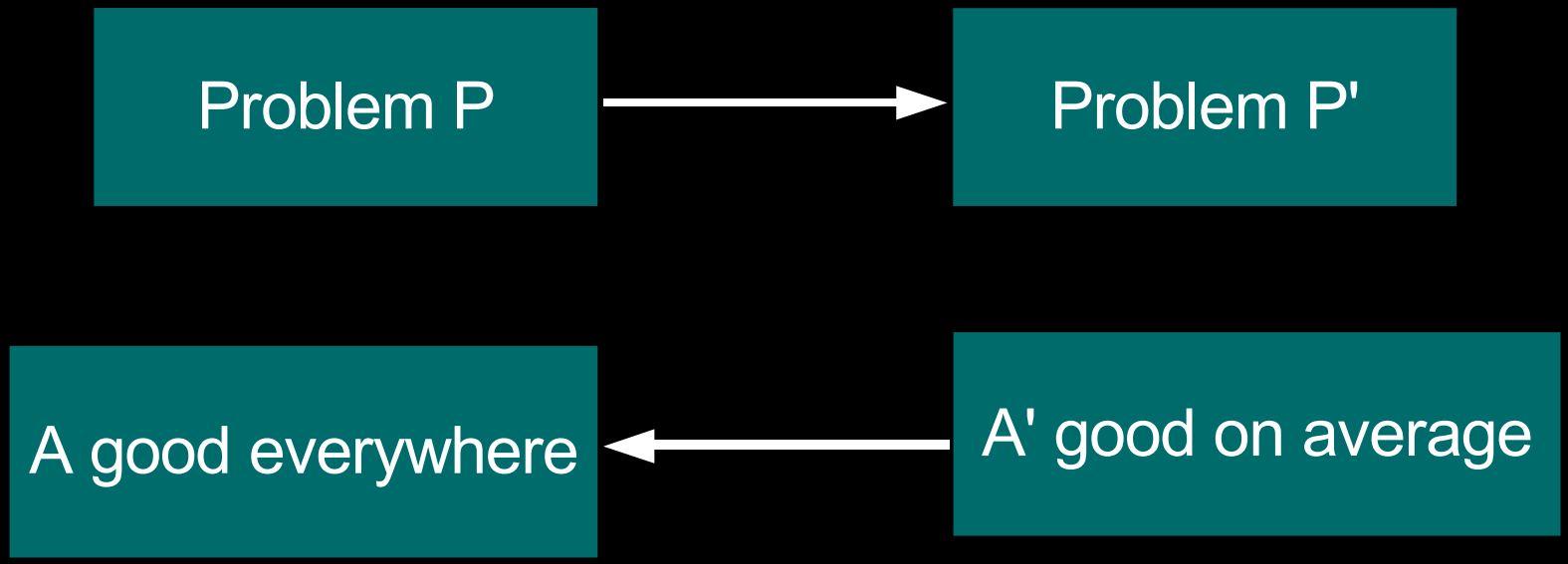
Worst-Case to Average-Case

- Given P define P' such that
- If P' has good-on-average algorithm
- Then P has algorithm good on all inputs



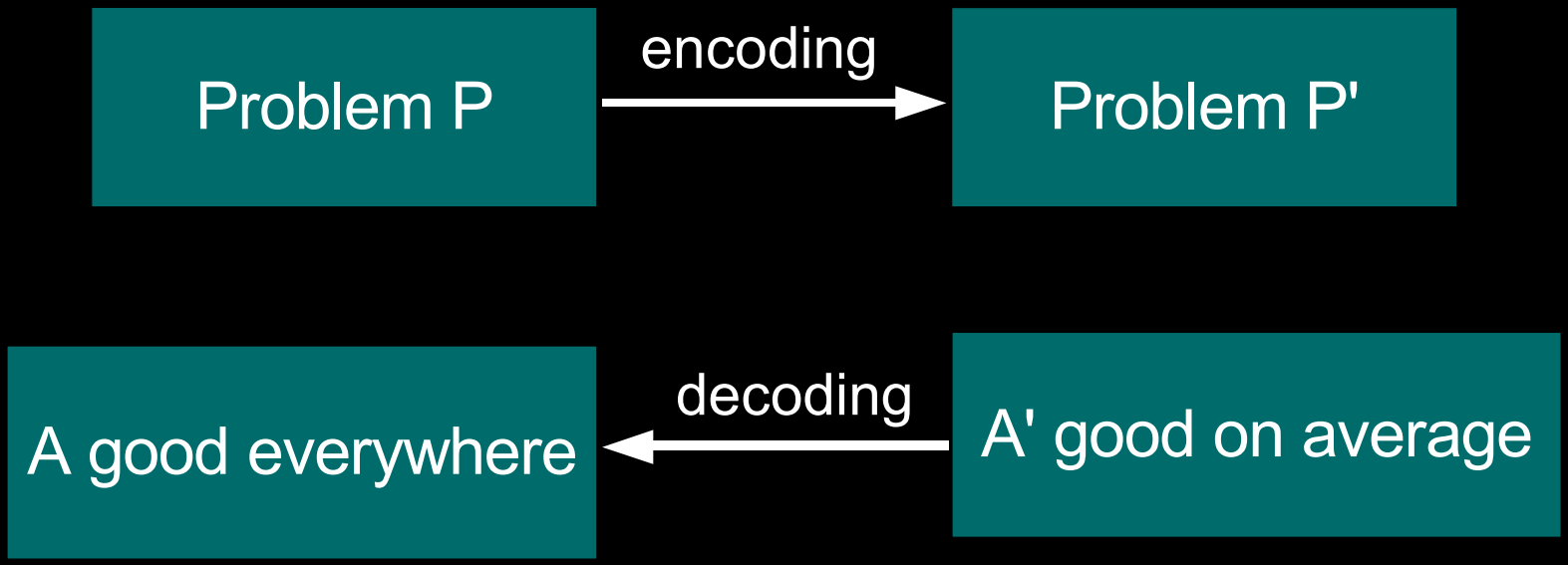
Worst-Case to Average-Case

- Given P define P' such that
- If P' has good-on-average algorithm
- Then P has algorithm good on all inputs



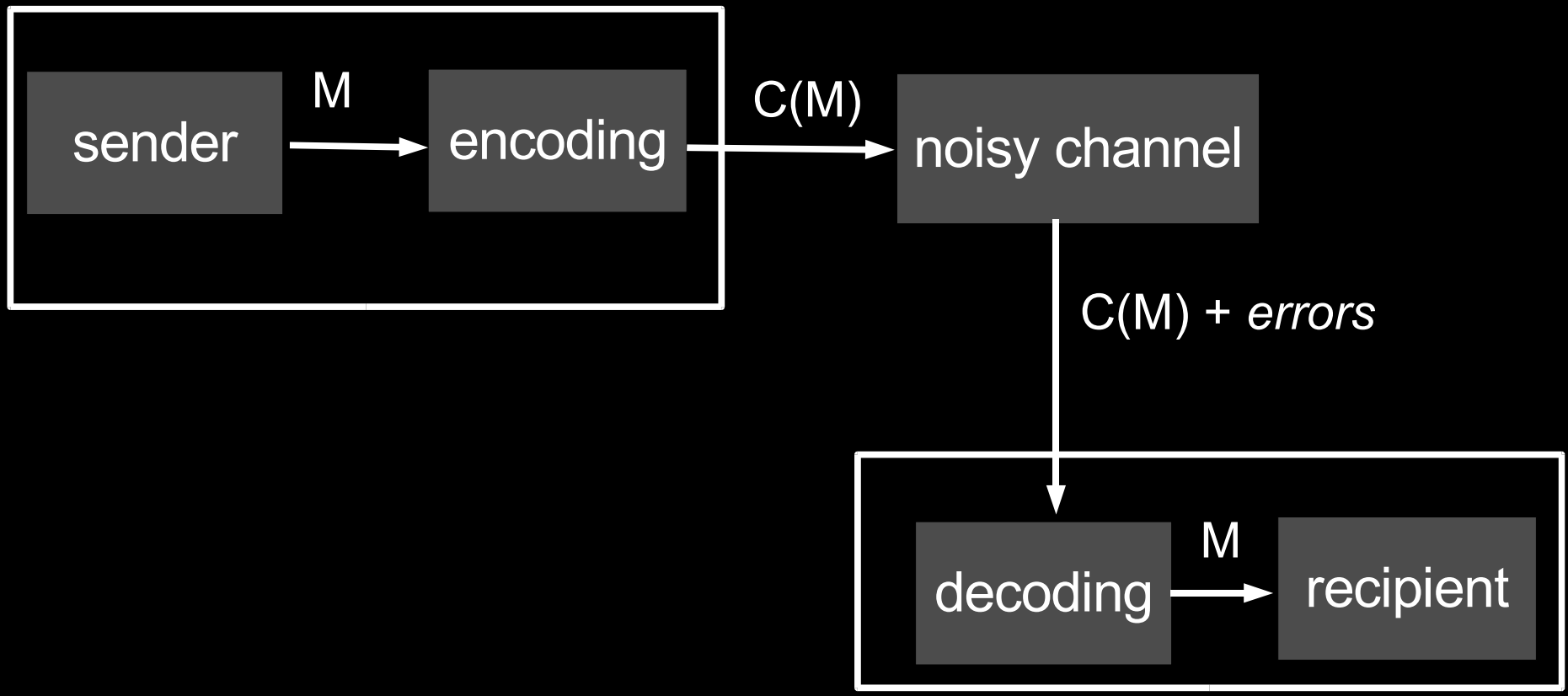
Worst-Case to Average-Case

- Given P define P' such that
- If P' has good-on-average algorithm
- Then P has algorithm good on all inputs



Error-Correcting Codes

Problem: reliable transmission over a noisy channel



Worst-Case to Average-Case

- Given P define P' such that
- If P' has good-on-average algorithm
- Then P has algorithm good on all inputs

- Coding theory dictionary:
  - P: message
  - P': encoding of P
  - Good-on-average alg for P': noisy transmission
  - Good-everywhere alg for P: decoding

A Glimpse Beyond

What about applications that require randomness?

- **Cryptography**
- Sublinear time algorithms (sampling,...)
- Simulations
- ...

Randomness Extraction

How to generate high-quality random bits from physical sources?

How to convert a distribution with biases and dependencies into a sequence of 'pure' random bits?

- [von Neumann Elias Blum Chor Goldreich Santha Vazirani Vazirani ...]
- (Seeded) Randomness Extractors [Zuckerman 1990]
- Connected to error-correcting codes, expander graphs, . . .

Another connection

- Nisan-Wigderson generator + error-control codes gives unconditional (seeded) extractors [T 1999]

Another connection

- Nisan-Wigderson generator + error-control codes gives unconditional (seeded) extractors [T 1999]

- Dictionary:

•Generator.....	•Randomness extractor
•Pseudorandom output.....	•Random output
•Worst-case complexity.....	•Entropy
•Transformation of worst-case complexity into average-case complexity..	•Error-correcting code

A Glimpse Beyond

- What about applications that require randomness
  - Seeded Randomness Extractors [Zuckerman 1990]
  - nearly optimal constructions
    [Lu Raz Reingold Shaltiel Ta-Shma Umans Vadhan Wigderson 99-03]
  - Some applications require (seedless) extractors different from Zuckerman's
  - Barak-Impagliazzo-Wigderson '04
    new approach using additive combinatorics
  - New Ramsey graphs, much current excitement
    [Barak Gabizon Kindler Rao Raz Shaltiel Shpilka Sudakov Zuckerman]

In Conclusion

- Under complexity-theoretic assumptions
  - All randomized algorithms can be made deterministic with polynomial time efficiency
  - Some existential proofs with probabilistic method can be made explicit
- Ideas in this field have led to unconditional combinatorial constructions, inspired derandomization of specific algs (not discussed)

http://www.cs.berkeley.edu/~luca/pacc

Google: luca pseudorandomness