JOURNAL OF COMPUTER AND SYSTEM SCIENCES 42, 346–398 (1991)

Average Case Completeness

YURI GUREVICH*

Electrical Engineering and Computer Science Department,
The University of Michigan, Ann Arbor,
Michigan 48109–2122

Received April 28, 1988; revised October 3, 1989

We explain and advance Levin's theory of average case completeness. In particular, we exhibit examples of problems complete in the average case and prove a limitation on the power of deterministic reductions. © 1991 Academic Press, Inc.

INTRODUCTION

Many NP hard problems are practically important and have to be solved in one way or another in spite of NP hardness. There are different approaches in the literature to this challenge: approximate algorithms, probabilistic algorithms, etc. The approach, adapted in this paper, is to forget about the worst case and to concentrate on the average case.

For simplicity, we speak about decision problems, rather than search problems, and restrict attention to algorithms that solve all instances of the problem in question. The first restriction is completely superfluous: the whole theory is readily generalizable to search problems. The second restriction may be relaxed as well.

We assume that a decision problem D comes together with a function μ that assigns probabilities to instances of D; the pair (D, μ) is called a randomized decision problem. How can one take advantage of probabilities? One possibility is to seek algorithms that almost always run in polynomial time. The common formalization of running almost always in Ptime is that for each n, the probability of hard instances of size n (where the running time exceeds the given polynomial in n bound) is bounded by an inverse polynomial of n. Powerful algorithms of that kind were devised for the Hamiltonian Circuit Problem; see [3] and references therein. Another approach is to seek algorithms whose expected running time is polynomial. An algorithm of that kind for Hamiltonian Circuit Problem with a fixed edge probability has been devised in [11]. Leonid Levin [16] suggested a natural liberalization of the second approach where an algorithm is considered fast if the expectation of some fixed root of the running time is polynomial. Algorithms that almost always run in Ptime (i.e., polynomial time) may be and often are slow in

* Partially supported by NSF Grant DCR 85-03275.

0022-0000/91 $3.00

Levin's sense. Levin's approach allows a nice reduction theory which is the subject of this paper.

The new reduction theory generalizes the reduction theory for NP problems. The role of NP plays a class RNP of randomized decision problems (D, μ) such that D is NP and the probability function μ satisfies a certain technical condition (see Section 1) that is usually satisfied in practice. In his exceedingly terse paper [16], Levin generalized polynomial time reductions to fit RNP problems and found a natural RNP complete problem, Randomized Tiling. To work correctly, a reduction should not diminish too much the probability of a given instance. As a result, reducing RNP problems is much more difficult than reducing NP problems. A priori, it is not clear that there exist complete RNP problems.

Levin's completeness proof is ingenious and complicated. The main part of the proof is devoted to establishing the completeness of a randomized (and bounded) version of the halting problem; the reduction of Randomized Halting to Randomized Tiling is relatively routine, but also not trivial. One contribution of this paper is a direct and simple proof of the Ptime completeness of Randomized Halting (Section 4).

David Johnson [13] provided some intuition behind Levin's definitions and proofs; he challenged readers to find additional complete RNP problems. The first additional Ptime complete RNP problems are presented in Sections 5 and 6 below. One of them is Randomized Post Correspondence Problem:

Instance. A nonempty list $(u_1, v_1), ..., (u_s, v_s)$ of pairs of binary strings, and the unary notation 1^n for a positive integer n.

Question. Do there exist a number $k \leqslant n$ and a function F from $[1 \cdots k]$ to $[1 \cdots s]$ such that the concatenation of strings $u_{F(1)}, ..., u_{F(k)}$ coincides with the concatenation of strings $v_{F(1)}, ..., v_{F(k)}$?

Probability. The probability function is given by the following experiment: Draw independently positive integers n and s with respect to the uniform probability distribution on positive integers, and then draw independently binary strings $u_1, v_1, ..., u_s, v_s$ with respect to the uniform probability on binary strings.

The uniform (or standard, or default) probabilities on positive integers and binary strings are described in Section 2.

We have also found that many apparently difficult RNP problems cannot, to all practical purposes, be proved Ptime complete for RNP. Let us explain this. Call a probability function μ *flat* if there exists $\varepsilon > 0$ such that $\mu(x) \leqslant 2^{-n^{\varepsilon}}$, i.e., $-\log \mu(x) \geqslant n^{\varepsilon}$, for all instances x of sufficiently large size n. Call a randomized decision problem (D, μ) *flat* if μ is flat. Let DEXP (resp. NEXP) be the class of decision problems decidable in deterministic (resp. nondeterministic) exponential time. In Section 8, we prove that if D is DEXP, μ is flat and (D, μ) is hard for RNP with respect to polynomial time reductions, then DEXP = NEXP. Thus, a flat problem cannot be proved Ptime complete for RNP unless DEXP = NEXP. The natural randomizations of usual NP complete problems very often are flat. For

example, every RNP graph problem is flat if the probability distribution on n-vertex graphs is determined by the edge-probability $f(n)$ with $n^{-2+\varepsilon} < f(n) < 1 - n^{-2+\varepsilon}$ for some constant $\varepsilon > 0$.

The idea of the incompleteness theorem is as follows. A NEXP problem D_0 can be turned into a very sparse RNP problem (D_1, μ_1) whose positive instances x have enormous (with regard to the size) probabilities. Given such an x, a reduction f of (D_1, μ_1) to a flat problem (D, μ) produces an instance $f(x)$ of a high probability and therefore a small size. It turns out that a deterministic exponential time procedure for D together with a polynomial time procedure for computing f give a deterministic exponential time procedure for D_0.

The incompleteness theorem survives the generalization to reductions computable in average polynomial time; actually, the incompleteness theorem of Section 8 is stated and proved for average polynomial time reductions. The theorem survives the generalization to Turing (as opposite to many-one) reductions. However, it does not survive the generalization to coin-flipping reductions. The proof fails because, instead of producing one instance $f(x)$ of a high probability and a small size, a randomizing reduction produces a multitude of instances of a small probability and a large size. Ramarathnam Venkatesan and his advisor Leonid Levin found [22] a natural randomized graph-coloring problem, which is flat and complete with respect to coin-flipping polynomial-time reductions; such reductions are considered in Section 9. Rich additional information on the theory of average case complexity can be found in [1]; also see [9].

An important question is whether the current state of RNP theory is sufficient to identify problems that are difficult on average. Why, in spite of the introduction of randomized reductions, there are still only a few RNP complete problems known? Is the setting not exactly right or are average-case completeness proofs inherently too difficult? It is possible that many problems difficult on average are not complete for the whole RNP but are complete for natural subclasses. In this connection, NP problems with small (log-size or, alternatively, polylog-size) witnesses cry for attention. The worst-case complexity for problems with small witnesses was a subject of study recently [19, 20]. But the case of statistically small witnesses is even more interesting. Note, for example, that in the case of uniform probability distribution over graphs with n vertices, the expected maximal clique size is about $2 \log n$.

This article contains a number of additional results and is organized as follows.

Section 1: The notion of polynomiality on average is defined and discussed. Then the analogs AP and RNP of the classes P and NP are introduced.

Section 2: Default probability functions on numbers and strings are introduced and discussed. Also, examples of RNP problems are given.

Section 3: Polynomial-time reducibility is defined and studied.

Section 4: A direct and simple proof of the polynomial-time completeness of Randomized Halting for RNP is given.

Section 5: Randomized Post Correspondence Problem is proved polynomial-time complete for RNP.

Section 6: Some additional RNP complete problems are given.

Section 7: Reducibility in average polynomial time is defined and studied.

Section 8: The incompleteness theorem is proved.

Section 9: Randomizing polynomial-time reductions are defined and studied.

Section 10: Sparse RNP problems are studied.

Appendix: The original completeness proof of Levin is reconstructed. (The Appendix is a result of cooperation of the author and his student David McCauley.)

In the meantime the reduction theory for average case complexity was substantially advanced and cleaned up somewhat; see [1, 25] and also [9, 26].

1. Polynomiality on Average; Classes *AP* and *RNP*

The main purpose of this section is to define the analogs for P and NP in the case of randomized decision problems. Some definitions are revised later in Section 9.

We start with terminology and notation. As usual, an *alphabet* is an ordered finite set of symbols, the letter Σ is reserved to denote alphabets, and Σ^* is the set of all Σ-strings. Order Σ^* first by length and then lexicographically; for brevity, that order is called lexicographical. Σ-strings are assigned natural numbers (starting from 0) with respect to the lexicographical order. The empty string is denoted e. The successor of a string x is denoted x^+. The alphabet $\langle 0, 1 \rangle$ is called the *binary alphabet*.

It is often assumed that, in principle, any decision problem D is the decision problem for some language L in some alphabet Σ:

Instance. A Σ-string w.

Question. Does w belong to L?

In applications, instances may be graphs or whatever, but usually there is no problem in coding them by strings.

For technical reasons, we need a more general notion of a decision problem over strings such that the domain (i.e., the set of instances) may be a proper (and not necessarily recognizable in polynomial time) subset of some Σ^*. We suppose that, in principal, every decision problem D is given by an alphabet $\Sigma(D)$ (or Σ_D), the domain $\mathrm{dom}(D) \subseteq \Sigma_D^*$, and a language $L(D)$ (or L_D) over $\Sigma(D)$:

Instance. An element w of $\mathrm{dom}(D)$.

Question. Does w belong to $L(D)$?

If D is a decision problem and $X \subseteq \Sigma_D^*$, then the *restriction* $D \mid X$ of D to X is the decision problem with alphabet $\Sigma(D)$, domain $D \cap X$, and language $L(D)$. Thus, an

binary strings and every $\mu^*(x) < 1$. Let $dx = 2^{-2|x|}$. By the definition of polynomial time computability with $k = 2|x| + 1$, there is a Ptime computable function $N'(x)$ such that every value of N' is a binary function and $|\mu^*(x) - N'(x)| < (dx)/2$. Round $N'(x)$ down to $2|x| + 1$ digits; if the last digit is a 1, then add $(dx)/2$. The result $N(x)$ is a binary fraction with at most $2|x|$ digits after the binary point, and $|\mu^*(x) - N(x)| < dx$.

Define

$$4\mu_1(x) = [\text{if } x \neq e \text{ then } N(x^+) - N(x) + 2\,dx, \text{ else } N(e^+) + 1].$$

Then $4\mu_1^*(x) = 1 + Nx + 2\Sigma_{e<y<x}\, dy$ if $x \neq e$, and therefore

$$\lim_{|x| \Rightarrow \infty} 4\mu_1^*(x) = 1 + 1 + 2 \sum_{n>1} 2^n/2^{2n} = 4.$$

Finally, note that $4\mu_1(x) > \mu(x)$. Indeed, $4\mu_1(e) > \mu^*(e^+) - de^+ + 1 > \mu(e)$, and if $x \neq e$ then

$$4\mu_1(x) = N(x^+) - N(x) + 2\,dx > (\mu^*(x^+) - dx) - (\mu^*(x) + dx) + 2\,dx = \mu(x).$$

Q.E.D.

For future references, note that, for the probability function μ_1 constructed in the proof of Lemma 1.6, each binary fraction $\mu_1(x)$, written without trailing zeroes, has at most $2 + 2|x^+| \leqslant 4 + 2|x|$ digits.

2. Standard Probability Functions and Examples of RNP Problems

In the first part of this section, we define standard (or default) probability functions on finite sets, the set of natural numbers, and the set of strings over a given alphabet. There are two reasons for us to introduce standard probability functions. One is to use them to define natural probability functions on more complicated objects; the use of standard probability functions hopefully supports the claim of naturality. The other reason is brevity. We can speak simply about a random natural number or a random binary string meaning the randomness with respect to the corresponding standard probability function.

The uniform probability function, assigning equal probabilities to all sample points, is our obvious choice for a standard probability function on any (non-empty) finite set. The choice of a default probability function on positive integers is not so obvious. We follow Levin [16]:

Definition. The standard probability of a positive integer n is proportional to n^{-2}.

(For simplicity, we ignore the empty string.) Additional arguments in favor of (ii) vs (i) may be found in [9]. Condition (ii) may be too restrictive as well. Consider, for example, a function f such that $f(x)=2^{|x|}$ if $|x|$ is even, and $f(x)=|x|$ otherwise. Suppose that, for each even n, the $\mu[H_n \leqslant 2^{-2|x|}$. One would expect that f is linear on μ-average, but condition (ii) is not satisfied. This leads us to the official definition:

DEFINITION. f is *linear on μ-average* if the expectation $\sum_{x \neq e} f(x) \cdot |x|^{-1} \cdot \mu(x)$ converges, and f is *polynomial on μ-average* if it is bounded by a polynomial of a function that is linear on μ-average.

Thus, f is polynomial on μ-average if and only if:

(iii) There exists $\varepsilon > 0$ such that $\sum_{x \neq e} (fx)^{\varepsilon} \cdot |x|^{-1} \cdot \mu(x) < \infty$, or

(iii') There exists an integer $k > 0$ such that $\sum_{x \neq e} (fx)^{1/k} \cdot |x|^{-1} \cdot \mu(x) < \infty$.

We say that ε (resp. k) *witnesses* the polynomiality of f on μ-average if (iii) (resp. (iii')) holds.

Condition (ii) has some advantage over condition (iii) because often one knows probability functions on instances of the same size and does not care about the probabilities of different sizes. The following proposition shows that, for many usual probability functions, the two conditions are equivalent.

PROPOSITION 1.1. *Let μ be a probability function on some Σ^* and suppose that there exists a polynomial p such that, for every n, either $\mu[H_n]=0$ or $\mu[H_n] \geqslant p(n)^{-1}$. Then conditions* (ii) *and* (iii) *are equivalent.*

Proof. It is easy to see that (ii) implies (iii). We prove the other implication. Suppose (iii). Then there exist ε and c such that

$$\sum_{n>0} \left[n^{-1} \cdot \sum_{|x|=n} (fx)^{\varepsilon} \mu(x) \right] = c < \infty.$$

We may restrict attention to $n > 0$ such that $\mu[|x|=n] > 0$. For each such n,

$$\sum_{|x|=n} (fx)^{\varepsilon} \mu(x) \leqslant cn,$$

$$\sum_{|x|=n} (fx)^{\varepsilon} \mu_n(x) \leqslant cn/\mu[|x|=n] \leqslant cnp(n).$$

Set $\delta = \varepsilon/2$. We may restrict attention to strings x such that $(fx)^{\delta} \geqslant p(n)$. Then $(fx)^{\delta} = (fx)^{\varepsilon} \cdot (fx)^{-\delta} \leqslant (fx)^{\varepsilon} \cdot p(n)^{-1}$ and therefore, for every n,

$$\sum_{|x|=n} (fx)^{\delta} \mu_n(x) \leqslant \left[\sum_{|x|=n} (fx)^{\varepsilon} \mu_n(x) \right] \cdot p(n)^{-1} \leqslant [cn \cdot p(n)] \cdot p(n)^{-1} = cn. \qquad \text{Q.E.D.}$$

The following sufficient condition for (iii) is useful sometimes:

(iv) There exists an integer $k>0$ such that $\sum_{x\neq e} f(x)\cdot|x|^{-k}\cdot\mu(x)<\infty$.

Condition (iv) implies condition (iii') with the same witness k. To prove this, note that we care only about those nonempty strings x where $(fx)^{1/k}\cdot|x|^{-1}>1$. On those strings $(fx)^{1/k}\cdot|x|^{-1}<f(x)\cdot|x|^{-k}$.

Until now, we looked into sufficient conditions for (iii). Here is a necessary condition:

(v) The expectation $\sum_{|x|>1}\log_{|x|}f(x)\cdot\mu(x)$ converges.

Why do we prefer condition (iii) to condition (v)? This question is related to another question, addressed in Section 2: Which probability functions on positive integers are natural? Condition (iii) fits well probability functions on positive integers which are inverse polynomials. Condition (v) is too liberal in that case. For example, suppose that $\mu(x)$ is proportional to $n^{-3}2^{-n}$, where $n=|x|$, so that $\mu[H_n]$ is proportional to the inverse polynomial n^{-3}. Then a fast-growing function $f(x)=|x|^{|x|}$ satisfies (v). Also, Proposition 1.1 fails if polynomiality on average is defined with respect to (v). This ends our discussion on the correct definition of polynomiality on average. A continuation of this discussion may be found in [9].

Next we give a useful criterion of polynomiality on average [22].

DEFINITION. A function ρ from some Σ^* to nonnegative reals is a *rarity function* for a probability function μ on Σ^* if the expectation of ρ is finite.

PROPOSITION 1.2. *Let f be a function from some Σ^* to nonnegative reals and μ be a probability function on Σ^*. The function f is polynomial on μ-average if and only if there exists a rarity function ρ for μ such that $f(x)$ is bounded by a polynomial of two arguments, $|x|$ and $\rho(x)$.*

Proof. It k witnesses that f is polynomial on average, define $\rho(x)=(fx)^{1/k}\cdot|x|^{-1}$, then $f(x)=(\rho(x)|x|)^k$. If $f(x)$ is bounded by a polynomial of $|x|$ and $\rho(x)$, then there exists a positive integer k such that, for sufficiently large x, we have $f(x)\leqslant(|x|\,\rho(x))^k$, so that $(fx)^{1/k}\cdot|x|^{-1}\leqslant\rho(x)$ and therefore k witnesses that f is polynomial on average. Q.E.D.

LEMMA 1.1. *Let μ be a probability function on some Σ^*, and f, g be functions from Σ^* to nonnegative reals, and r be a positive real. If f and g are polynomial on μ-average then so are* $\max(f,g)$, f^r, $f\times g$, *and* $f+g$.

Proof. Let $L=|x|$. We may suppose that, for some ε, both expectations $E[f^\varepsilon/L]$ and $E[g^\varepsilon/L]$ are finite. Let $h(x)=\max(f(x),g(x))$. Then

$$E[h^\varepsilon/L] = \sum_x (hx)^\varepsilon \cdot |x|^{-1} \cdot \mu(x)$$

$$= \sum_{fx \geqslant gx} (hx)^\varepsilon \cdot |x|^{-1} \cdot \mu(x) + \sum_{fx < gx} (hx)^\varepsilon \cdot |x|^{-1} \cdot \mu(x)$$

$$= \sum_{fx \geqslant gx} (fx)^\varepsilon \cdot |x|^{-1} \cdot \mu(x) + \sum_{fx < gx} (gx)^\varepsilon \cdot |x|^{-1} \cdot \mu(x)$$

$$\leqslant E[f^\varepsilon/L] + E[g^\varepsilon/L] < \infty.$$

The rest is obvious. Q.E.D.

DEFINITION. f is *polynomial on μ-average* on a subset X of Σ^* if there exists $\varepsilon > 0$ such that

$$\sum_{e \neq x \in X} (fx)^\varepsilon \cdot |x|^{-1} \cdot \mu(x) < \infty.$$

DEFINITION. A randomized decision problem (D, μ) is *decidable in APtime* if some Turing machine decides D within time polynomial on average with respect to μ. AP is the class of randomized decision problems decidable in APtime. A function f from some Σ_1^* to some Σ_2^* is computable in APtime with respect to a probability function μ_1 on Σ_1^* if some Turing machine computes f within time polynomial on average with respect to μ_1.

AP is the analog for P. The letter A stands for "average."

LEMMA 1.2. *Suppose that μ_1 is a probability function on some Σ_1^*, f is a function from Σ_1^* to some Σ_2^*, and $\mu_2(y) = \sum_{fx=y} \mu_1(x)$ is the induced probability function on Σ_2^*.*

1. *Let T be a function from Σ_2^* to nonnegative reals. If $|fx|$ is polynomial on μ_1-average and T is polynomial on μ_2-average then the composition $h = T \circ f$ is polynomial on μ_1-average.*

2. *Let g be a function from Σ_2^* to some Σ_3^*. If f is computable in APtime* wrt *μ_1 and g is computable in APtime* wrt *μ_2 then the composition $g \circ f$ is computable in APtime* wrt *μ_1.*

Proof. (1) Let k witness that T is polynomial on μ_2-average. For every positive $m \geqslant 1$,

$$\sum_{y \neq e} (Ty)^{1/k} \cdot |y|^{-1} \cdot \mu_2(y) < \infty$$

$$\to \sum_{y \neq e} (Ty)^{1/km} \cdot |y|^{-1/m} \cdot \mu_2(y) < \infty$$

$$\to \sum_{fx \neq e} (hx)^{1/km} \cdot |fx|^{-1/m} \cdot \mu_1(x) < \infty.$$

We can safely ignore strings x such that $x = e$ or $fx = e$. If $|fx|$ is polynomially bounded and m is such that $|fx|^{1/m} < |x|$ for sufficiently long x, then km witnesses that h is polynomial on μ_1-average:

$$\sum (hx)^{1/km} \cdot |x|^{-1} \cdot \mu_1(x) < \infty.$$

In the general case. Let m witness that $|fx|$ is polynomial on μ_1-average:

$$\sum |fx|^{1/m} \cdot |x|^{-1} \cdot \mu_1(x) < \infty.$$

Let

$$\alpha(x) = (hx)^{1/2km} \cdot |fx|^{-1/2m},$$

so that the expectation $E[\alpha^2]$, with respect to μ_1, is finite. Let

$$\beta(x) = |fx|^{1/2m} \cdot |x|^{-1/2},$$

so that the expectation $E[\beta^2]$, with respect to μ_1, is finite. Then the expectations $E[\alpha^2 + \beta^2]$ and $E[\alpha\beta]$ are finite. Hence

$$\sum (hx)^{1/2km} \cdot |x|^{-1/2} \cdot \mu_1(x) < \infty,$$

$$\sum (hx)^{1/2km} \cdot |x|^{-1} \cdot \mu_1(x) < \infty.$$

(2) The computation of $g \circ f$ splits into two parts: Computing $y = f(x)$ and then computing $g(y)$. We need to show only that the second part can be done in APtime with respect to μ_1. We know that $g(y)$ is computable in time $T(y)$ polynomial on μ_2-average. Now use (1). Q.E.D.

For a technical reason, we are interested in probability distributions that are Ptime computable. It is possible, as in [17] to restrict attention to probability distributions with rational values; such an approach is justified later in this section. But it seems to us more appropriate to extend the notion of Ptime computability to real-valued functions. For simplicity, we restrict attention to functions with values in the real interval $[0, 1]$.

DEFINITION (cf. [14]). A function f from some Σ^* to the interval $[0, 1]$ of reals is *computable in polynomial time* if there exists a polynomial time algorithm $A(x, 1^k)$ such that, for every Σ-string x and every positive integer k, $A(x, 1^k)$ is a binary fraction and $|f(x) - A(x, 1^k)| < (\frac{1}{2})^k$.

LEMMA 1.3 [Blass and Gurevich].

1. *If f and g are Ptime computable functions from some Σ^* to the real interval $[0, 1]$, then $f + g$, $f - g$, and $f \times g$ are Ptime computable as well.*

2. *Let f be a monotone function from some Σ^* to the real interval $[0, 1]$ and let $A(x, 1^k)$ witness the Ptime computability of f. There exists a witness $B(x, 1^k)$ to Ptime computability of f such that, for every k, B is monotone in x.*

Proof. (1) is easy.

(2) Without loss of generality, f is increasing. Fix k; to simplify notation, we omit the argument 1^k. View a Σ-string x as a positive integer (say, 1 plus the number of x in the lexicographical order of Σ-strings). Here is an algorithm computing the desired $B(x)$:

1. Find the least integer p such that $x \leqslant 2^p$.
2. For every $q \leqslant p$, set $B(2^q) = \max\{A(2^r) : r \leqslant q\}$.
3. Halt if $p = 0$; otherwise set $a = 2^{p-1}$ and $b = 2^p$.
4. While $B(x)$ is undefined do:

(a) If $B(a) = B(b)$ then set $B(x) = A(x)$ and halt, else set $c = \lfloor (b-a)/2 \rfloor$.

(b) If $A(c) \leqslant B(a)$ then set $B(c) = B(a)$, else if $A(c) > B(b)$ then set $B(c) = B(b)$, else set $B(c) = A(c)$.

(c) If $x \leqslant c$ then set $b = c$, else set $a = c$. Q.E.D.

Remark. The Ptime computability of f does not guarantee the computability (let alone Ptime computability) of the kth digit of fx. For, let M be a Turing machine that computes a function $b(x)$ from binary strings to $\{0, 1\}$ such that the sets $\{x : b(x) = 0\}$, $\{x : b(x) = 1\}$ are recursively inseparable. Let $T(x)$ be the time that M works on instance x; $T(x)$ is infinite if M does not halt on x. If M halts on x, let

$$f(x) = 0.0(01)^{T(x)}1 \qquad \text{and} \qquad g(x) = 0.0(10)^{T(x)}b(x).$$

Otherwise, let

$$f(x) = 0.0(01)^{\infty} \text{ and } g(x) = 0.0(10)^{\infty}.$$

Obviously, f and g are Ptime computable. Let $h = f + g$. If $b(x) = 0$ then $h(x) = 0.0...$ and if $b(x) = 1$, then $h(x) = 0.1$. Thus, computing the first digit of h (after the binary point) would separate the inseparable sets.

By Lemma 1.3, a probability function μ is Ptime computable if the corresponding probability distribution μ^* is Ptime computable. The converse is not necessarily true:

LEMMA 1.4 [2]. *There exists a Ptime computable probability function μ such that the probability distribution μ^* is not Ptime computable unless $P = NP$.*

Proof. Construct a Ptime computable binary relation R on binary strings such $|x| = |y|$ for all $(x, y) \in R$ and the language $L = \{x : \exists y(xRy)\}$ is NP complete. Construct a Ptime computable probability function ν on binary strings such that every $\nu(x)$ is a binary fraction, and $\nu(x) = 0 \leftrightarrow |x|$ is odd.

Define a probability function μ on binary strings w as follows. If $|w|$ is even then $\mu(w) = 0$. Suppose that $w = xby$, where $|x| = |y|$ and b is a binary bit. If $b = 0$ then $\mu(w) = [\text{if } xRy \text{ then } \nu(xy) \text{ else } 0]$, and if $b = 1$ then $\mu(w) = \nu(xy) - \mu(x0y)$. Obviously, μ is Ptime computable and

$$\sum \mu(w) = \sum_{|x|=|y|} \mu(x0y) + \mu(x1y) = \sum_{|x|+|y|} \nu(xy) = 1.$$

If μ^* is Ptime computable then L is in P:

$$\exists y(xRy) \leftrightarrow \mu^*(x1z) - \mu^*(x0z) \neq 0,$$

where $z = 0^{|x|}$. Q.E.D.

DEFINITION. Let μ_1, μ_2 be probability functions on strings in the same alphabet Σ. μ_2 *dominates* (resp. *weakly dominates*) μ_1 if there is a function f from Σ^* to non-negative reals such that $\mu_1(x) \leqslant f(x) \cdot \mu_2(x)$ and f is polynomially bounded (resp. polynomial on μ_1 average). It is possible to require that $\mu_1(x) = f(x) \cdot \mu_2(x)$. The probability distribution μ_2^* *dominates* (resp. *weakly dominates*) the probability distribution μ_1^* if μ_2 dominates (resp. weakly dominates) μ_1.

On first glance, the definition may look a little strange: μ_2 needs a factor to be equal to μ_1. But, considering for simplicity positive μ_1 and μ_2, note that if μ_2 dominates μ_1 then the ratio μ_1/μ_2 is bounded by f, whereas there is no a priori bound on the ratio μ_2/μ_1.

LEMMA 1.5. *If μ_1 is weakly dominated by μ_2 and (D, μ_2) is AP then (D, μ_1) is AP.*

Proof. Since (D, μ_2) is AP, D is decidable within time $T(x)$ such that

$$\sum (Tx)^{1/k} \cdot |x|^{-1} \cdot \mu_2(x) < \infty$$

for some k. We prove that

$$\sum (Tx)^{1/l} \cdot |x|^{-1} \cdot \mu_1(x) < \infty$$

for some l that is chosen later. Let g witness that μ_1 is dominated by μ_2, and let $X = \{x : g(x) \cdot (Tx)^{1/l} \leqslant (Tx)^{1/k}\}$. Then

$$\sum_{x \in X} (Tx)^{1/l} \cdot |x|^{-1} \cdot \mu_1(x)$$

$$= \sum_{x \in X} (Tx)^{1/l} \cdot g(x) \cdot |x|^{-1} \cdot \mu_2(x)$$

$$\leqslant \sum_{x \in X} (Tx)^{1/k} \cdot |x|^{-1} \cdot \mu_2(x) < \infty.$$

Let x range over the complement of X. Then $T^{1/k} < g \cdot T^{1/l}$, $T^{l-k} < g^{kl}$, and $T^{1/l} < g^{k/(l-k)}$. Since g is polynomial on μ_1-average, there is j such that

$$\sum (gx)^{1/j} \cdot |x|^{-1} \cdot \mu_1(x) < \infty.$$

Choose l such that $k/(l-k) < 1/j$. Q.E.D.

An alternative proof of Lemma 1.5 is given in Section 7.

DEFINITION. RNP is the class of randomized decision problems (D, μ) such that D is NP and μ is dominated or weakly dominated by a probability function ν with Ptime computable probability distribution ν^*.

RNP is our analog of NP for randomized decision problems. Actually, the restriction to NP decision problems in the above definition may be rightfully questioned, but in this paper we stick to it.

Note that the Ptime computability of a probability distribution requires the Ptime computability of the probabilities of only very special events $\{y : y < x\}$. Levin hypothesizes [13] that any natural probability function either has a polynomial time computable distribution, or else is dominated by a function that does. Johnson writes that it is not difficult to devise encodings that make "each of the distributions we have discussed in this column" polynomial time computable. Our experience supports Levin's hypothesis as well. However, there exist important probability functions that are not Ptime computable. In particular, information complexity (i.e., Kolmogorov complexity) gives rise to a recursively enumerable (in appropriate sense) probability function (say, on binary strings) that dominates any other recursively enumerable probability function [24]. That maximal probability function is not Ptime computable and is not dominated by any Ptime computable probability function.

An important generalization of Ptime computable probability distributions was introduced recently in [1]; they are so-called *samplable* distributions. See the discussion in [9] in this connection.

LEMMA 1.6. *For every probability function μ with a Ptime computable probability distribution μ^* there is a positive probability function μ_1 such that μ_1^* is Ptime computable and every value of μ_1 is a finite binary fraction and $\mu(x) = O(\mu_1(x))$.*

Proof. To simplify somewhat the exposition, we assume that μ is defined on

binary strings and every $\mu^*(x) < 1$. Let $dx = 2^{-2|x|}$. By the definition of polynomial time computability with $k = 2|x| + 1$, there is a Ptime computable function $N'(x)$ such that every value of N' is a binary function and $|\mu^*(x) - N'(x)| < (dx)/2$. Round $N'(x)$ down to $2|x| + 1$ digits; if the last digit is a 1, then add $(dx)/2$. The result $N(x)$ is a binary fraction with at most $2|x|$ digits after the binary point, and $|\mu^*(x) - N(x)| < dx$.

Define

$$4\mu_1(x) = [\text{if } x \neq e \text{ then } N(x^+) - N(x) + 2\,dx, \text{ else } N(e^+) + 1].$$

Then $4\mu_1^*(x) = 1 + Nx + 2\Sigma_{e<y<x}\, dy$ if $x \neq e$, and therefore

$$\lim_{|x| \Rightarrow \infty} 4\mu_1^*(x) = 1 + 1 + 2 \sum_{n>1} 2^n/2^{2n} = 4.$$

Finally, note that $4\mu_1(x) > \mu(x)$. Indeed, $4\mu_1(e) > \mu^*(e^+) - de^+ + 1 > \mu(e)$, and if $x \neq e$ then

$$4\mu_1(x) = N(x^+) - N(x) + 2\,dx > (\mu^*(x^+) - dx) - (\mu^*(x) + dx) + 2\,dx = \mu(x).$$

Q.E.D.

For future references, note that, for the probability function μ_1 constructed in the proof of Lemma 1.6, each binary fraction $\mu_1(x)$, written without trailing zeroes, has at most $2 + 2|x^+| \leqslant 4 + 2|x|$ digits.

2. Standard Probability Functions and Examples of RNP Problems

In the first part of this section, we define standard (or default) probability functions on finite sets, the set of natural numbers, and the set of strings over a given alphabet. There are two reasons for us to introduce standard probability functions. One is to use them to define natural probability functions on more complicated objects; the use of standard probability functions hopefully supports the claim of naturality. The other reason is brevity. We can speak simply about a random natural number or a random binary string meaning the randomness with respect to the corresponding standard probability function.

The uniform probability function, assigning equal probabilities to all sample points, is our obvious choice for a standard probability function on any (non-empty) finite set. The choice of a default probability function on positive integers is not so obvious. We follow Levin [16]:

Definition. The standard probability of a positive integer n is proportional to n^{-2}.

Discussion. If the desired standard probability function $\mu(n)$ decreases too quickly then too much weight is given to small instances. For example if $\mu(n) = 2^{-n}$ then the expectation of $2^{n/2}$ with respect to μ converges and $2^{n/2}$ appears to be bounded on average, which is undesirable. Proposition 1.1 justifies restricting attention to probability functions satisfying the assumption of the proposition. Further, it is natural to restrict attention to probability functions inversely proportional to polynomials. It is easy to check that if μ and ν are inverse polynomials such that both $\Sigma\mu(n)$ and $\Sigma\nu(n)$ converge then any function polynomial on μ-average is polynomial also on ν-average. Thus, in a sense, it is immaterial which specific inverse polynomial to choose. The choice of n^{-2} is natural.

There are natural probability functions that grow slower than probability functions given by inverse polynomials. Consider, for example, probability functions proportional to $n \cdot (\log n)^2$, $n \cdot \log n \cdot (\log \log n)^2$, etc. These functions seem less convenient, but they have their own advantages. For example, adapt, for a moment, the alternative definition of polynomiality on average based on condition (v) in Section 1: A function f is polynomial on μ-average if the μ-expectation of $\log_{|x|} f(x)$ converges. Then a relatively fast-growing function $f(x) = |x|^{\log_2 |x|}$ is not polynomial on average with respect to any of the probability functions in question, but it is polynomial on average with respect to, say, the probability function proportional to n^{-3}.

If the uniform probability distribution is an ideal (an unreachable ideal in the case of a countable infinite set of sample points), then one may be interested in even slower growing probability function. There is no such thing as the slowest growing probability function. The situation changes however if one restricts attention to recursively enumerable (in an appropriate sense [24]) probability functions and does not distinguish between probability function μ and ν such that $\mu(n) = O(\nu(n))$ and $\nu(n) = O(\mu(n))$. Then there is the slowed growing probability function; however, it is not dominated or weakly dominated by any Ptime computable probability function. End of discussion.

DEFINITION. In the case of natural numbers, the standard probability of a positive n is proportional to n^{-2}, and the standard probability of 0 is positive. (The exact value of the standard probability of 0 will be immaterial.)

DEFINITION. Let Σ be a k-letter alphabet. The standard probability function on Σ^* assigns the probability proportional to $n^{-2}k^{-n}$ to any strings of length n. (It corresponds to the following experiment: choose randomly a natural number n, and then choose randomly a string of length n.)

An alternative natural approach is to identify strings with natural numbers and use the standard probability function for natural numbers [16]. One should be a little careful though. Suppose, for example, that the alphabet in question is binary and assign to a binary string w the probability proportional to the inverse of the square of the number of w in the lexicographical order of binary strings. Then the

probability of the event $\{w : |w| = n\}$ is about 2^{-n} which is too little. Assigning the probability proportional to $n^{-1}(\log_2 n)^{-2}$ to the number n and the string of number n results in the probability of the event $\{w : |w| = n\}$ being roughly proportional to n^{-2}.

Remark. Sometimes, standard probability functions are called uniform even though they are not truly uniform.

In the rest of this section, we give some examples of RNP problems. The probability functions are described by means of appropriate experiments.

Randomized 3-Coloring

Instance. A graph on an initial segment $[0 \cdots (n-1)]$ of natural numbers.

Question. Is the graph 3-colorable?

Probability. Randomly choose a positive integer n, and then randomly choose a graph on $[0 \cdots (n-1)]$.

The Randomized 3-Coloring Problem happens to be AP. The usual backtracking solves it in about (surprise!) 197 steps on average [23]. The reason is that there are very simple and probable witnesses to non-colorability, like a clique of 4. The average time can be further cut down if the algorithm starts with a direct search for such witnesses.

DEFINITION. Consider a sample space of graphs on the segment $[0 \cdots (n-1)]$ of natural numbers where events "$\{u, v\}$ is an edge" are independent. Here u and v are distinct vertices. If each of these $n(n-1)/2$ events has the same probability p, we say that the probability function is given by the edge probability p. If $p = \frac{1}{2}$ then the probability function is uniform.

Randomized Cliques

Instance. A graph on an initial segment $[0 \cdots (n-1)]$ of natural numbers and a positive integer $k < n$.

Question. Is there a clique of size $> k$ in the graph?

Probability. Randomly choose a positive integer n, and then randomly choose a graph on $[0 \cdots (n-1)]$.

It is an open problem whether the Randomized Clique Problem is AP. See [21] in this connection. It is not difficult to devise a backtracking algorithm that inspects all cliques in lexicographical order and this way finds a clique of the maximal size. The expected run time of that algorithm is bounded by

$$(n \cdot e^2/l)^{(l-r)/2} \cdot \pi(n),$$

where e is the basis for natural logarithms, $l = \log_2 n$, $r = \log_2 l$, and π is a polynomial; a similar estimation is valid if the probability function on n-vertex graphs is

given by a fixed edge probability p, except the basis for logarithms is $1/p$ rather than 2.

Randomized Hamiltonian Circuits with Edge Probability p

Instance. A graph on $[0 \cdots (n-1)]$.

Question. Is there a Hamiltonian circuit in the graph?

Probability. Randomly choose a positive integer n, and then choose a graph on $[0 \cdots (n-1)]$ with respect-to the given edge probability p.

There is a decision algorithm for Randomized Hamiltonian Circuits with expected run time $O(n)$ for each fixed edge probability p [11]. The fact that Randomized Hamiltonian Circuits with edge probability $\frac{1}{2}$ is AP is proved in [3].

Randomized Tiling Problem over an Alphabet Σ

Some definitions are needed. A tile is a quadruple

$$\begin{array}{ccc} & v & \\ u & & w \\ & x & \end{array}$$

of Σ-strings. A function τ from the square $[0 \cdots (n-1)] \times [0 \cdots (n-1)]$ to a set T of tiles is a *T-tiling* of the square if

$$\text{left}[\tau(i+1, j)] = \text{right}[\tau(i, j)] \quad \text{and} \quad \text{bottom}[\tau(i, j+1)] = \text{top}(\tau(i, j)]$$

for all appropriate i and j. A function ρ from $[0 \cdots (j-1)]$ to T is a *T-row* of length j if each $\text{left}[\rho(i+1)] = \text{right}[\rho(i)]$. Now we are ready to formulate the problem.

Instance. A finite set T of tiles, the unary notation 1^n for a positive integer n, a positive integer $k < n$, and a T-row ρ of some length j such that either $j = k$ or else $j < k$ and T has no t with $\text{left}[t] = \text{right}[\rho(j)]$.

Question. Does there exist a T-tiling τ of the square $[0 \cdots (n-1)] \times [0 \cdots (n-1)]$ with $\tau(0, i) = \rho(i)$ for all $i < j$?

Probability. Choose T with respect to your favorite positive probability function. Choose randomly n, k and $\rho(0)$. If $\rho(i)$ has been chosen, $i < k-1$ and the set $T_i = \{t : t \in T \text{ and } \text{left}[t] = \text{right}[\rho(i)]\}$ is not empty, then choose $\rho(i+1)$ randomly from T_i.

Randomized Tiling is complete for RNP in an appropriate sense [16]; a reconstruction of Levin's proof can be found in the Appendix.

3. Ptime Reducibility

If P = NP then AP includes RNP. Hence it is hard to demonstrate an RNP problem which is not AP. Instead, one can develop a reduction theory for RNP problems and demonstrate complete RNP problems. RNP completeness of a randomized decision problem witnesses that the problem is hard in the average case. This section is devoted to polynomial time reducibility of RNP problems; the existence of a Ptime complete RNP problem is established in the next section. It is worth mentioning that the inclusion $\mathrm{RNP} \subseteq AP$ is not very likely either: by a theorem of Ben-David and Luby in Section 8 below, it implies that every problem decidable in nondeterministic exponential time is decidable in deterministic exponential time.

As usual, we say that a function f *reduces* a decision problem D_1 to a decision problem D_2 if, for every $x \in \mathrm{dom}(D_1)$, $x \in L(D_1)$ if and only if $f(x) \in L(D_2)$.

Definition.

1. A function f *transforms* a probability function μ_1 into a probability function μ_2 if $\mu_2(y) = \sum_{fx=y} \mu_1(x)$ for all sample points y in the domain of μ_2.

2. A function f *transforms* (D_1, μ_1) into (D_2, μ_2) if it reduces $D_1 | \{x : \mu_1(x) > 0\}$ to D_2 and transforms μ_1 into μ_2.

Lemma 3.1.

1. *Suppose that a function f transforms μ_1 into a restriction $\mu_2 | Y$ of μ_2, R is the range of f and $R_0 = \{f(x) : \mu_1(x) > 0\}$. Then $\mu_2 | R_0 = \mu_2 | Y$ and there exists $\nu \geqslant \mu_1$ such that f transforms ν into $\mu_2 | R$.*

2. *Suppose that a function f transforms (D_1, μ_1) into a restriction of (D_2, μ_2) and (D_2, μ_2) is AP. If f is computable in polynomial time or in time polynomial on μ_1-average then (D_1, μ_1) is AP.*

3. *Every RNP problem (D, μ) is Ptime transformable to some RNP problem (D_1, μ_1) over the binary alphabet.*

Proof. (1) The first claim is obvious. It is not true though that Y necessarily coincides with R_0; it can be a proper extension of R_0.

The desired ν is proportional to μ_1 on $\{x : \mu_1(x) > 0\}$. For every $y \in R - R_0$, $(\mu_2 | R)(y) = \sum_{fx=y} \nu(x)$.

(2) Every restriction of an AP problem is AP. For, suppose that a decision problem D is decidable in time $T(x)$ polynomial on average with respect to some probability function μ and let X be a collection of instances of D of some probability $\mu(X) > 0$. If k witness that T is polynomial on μ-average, then

$$\sum_{x \in X} (Tx)^{1/k} \cdot (\mu | X)(x) = \mu(X)^{-1} \cdot \sum_{x \in X} (Tx)^{1/k} \cdot \mu(x) < \infty.$$

Hence we may assume that f transforms (D_1, μ_1) to (D_2, μ_2) itself. Let A be a decision algorithm for D_2 whose run time is polynomial on μ_2-average. To decide an instance x of D_1, compute $f(x)$ and then apply A to $f(x)$. By Lemma 1.2, the run time of A on $f(x)$ is polynomial on μ_1-average.

(3) Let Σ be the alphabet of D. If Σ is unary and a is the only letter of Σ, define $f(a^n) = 1^n$; otherwise let f take the nth Σ-string to the nth binary string. The desired D_1 is the decision problem for the language $\{f(x) : x \in L(D)\}$, and the desired $\mu_1(y) = \mu(f^{-1}(y))$. If μ is dominated by some ν with Ptime computable ν^* and $\nu_1(y) = \nu(f^{-1}(y))$, then ν is dominated by ν_1 and ν_1^* is Ptime computable.

Q.E.D.

Let $\mu_1 \leqslant \mu_2$ denote that μ_1 is dominated by μ_2, and let $\mu_1 \xrightarrow{f} \mu_2$ denote that f transforms μ_1 into μ_2.

DEFINITION. *μ_2 dominates μ_1 with respect to a function f*, symbolically $\mu_1 \leqslant^f \mu_2$, if there exists some $\nu \geqslant \mu_1$ such that f transforms ν into a restriction of μ_2.

LEMMA 3.2. *μ_2 dominates μ_1 with respect to a one-to-one function f if and only if the probability function $\nu(x)$ proportional to $\mu_2(fx)$ dominates μ_1.*

Proof. First, suppose that μ_2 dominates μ_1 wrt f. Then there exists $\nu_1 \geqslant \mu_1$ such that f transforms ν_1 to a restriction $\mu_2 | Y$ of μ_2. We have

$$\nu_1(x) = (\mu_2 | Y)(fx) = \mu(Y)^{-1}\, \mu_2(fx) = \mu(Y)^{-1}\, \nu(x),$$

so that $\nu \geqslant \nu_1 \geqslant \mu_1$. Second, suppose that $\nu \geqslant \mu_1$. Since f transforms ν to μ_2, μ_2 dominates μ_1 wrt f. Q.E.D.

LEMMA 3.3. *Let f be a Ptime computable function from some Σ_1^* to some Σ_2^*.*

1. *If $\mu_1 \xrightarrow{f} \nu_2 \leqslant \mu_2$ for some ν_2 then $\mu_1 \leqslant^f \mu_2$.*
2. *Suppose that $f: \Sigma_1^* \to \Sigma_2^*$ is honest, i.e., $|x|$ is bounded by a polynomial of $|fx|$. If $\mu_1 \leqslant^f \mu_2$ then there is ν_2 such that $\mu_1 \xrightarrow{f} \nu_2 \leqslant \mu_2$.*

Proof. (1) Without loss of generality, we may suppose that $\mu_2(y) = 0$ for every y such that $\nu_2(y) = 0$. For, let μ be the restriction of μ_2 to $\{y : \nu_2(y) > 0\}$. Obviously, μ dominates ν_2. Suppose that some ν dominates μ_1 and f transforms ν to a restriction of μ. Then f transforms ν to a restriction of μ_2 and therefore $\mu_1 \leqslant^f \mu_2$.

Since μ_2 dominates ν_2, there exists a polynomially bounded function g such that $\nu_2(y) = g(y) \cdot \mu_2(y)$. Define:

$$\nu_1(x) = [\text{if } \nu_2(fx) > 0 \text{ then } \mu_1(x) \cdot (g(fx))^{-1}, \text{ else } 0].$$

Since $|fx|$ and $g(y)$ are polynomially bounded, $g(fx)$ is polynomially bounded and therefore $\mu_1 \leqslant \nu_1$. We check that f transforms ν_1 into μ_2. If $\nu_2(y) = 0$ then

$\sum_{fx=y} \nu_1(x) = 0 = \mu_2(y)$, and if $\nu_2(y) > 0$ then $g(y) > 0$ and $\sum_{fx=y} \nu_1(x) = (\sum_{fx=y} \mu_1(x)) \cdot (gy)^{-1} = \nu_2(y) \cdot (gy)^{-1} = \mu_2(y)$.

(2) By Lemma 3.1(1), there exists ν_1 such that $\mu_1 \leqslant \nu_1 \xrightarrow{f} \mu_2 \mid Y$, where Y comprises points $f(x)$ with $\mu_1(x) > 0$. Without loss of generality, $\mu_2 \mid Y = \mu_2$, for if $\mu_1 \xrightarrow{f} \nu_2 \leqslant \mu_2 \mid Y$ then $\mu_1 \xrightarrow{f} \nu_2 \leqslant \mu_2$. Define:

$$g(x) = [\text{if } \mu_1(x) > 0 \text{ then } \mu_1(x)/\nu_1(x), \text{ else } 1]$$

$$\nu_2(y) = \sum_{fx=y} \mu_1(x)$$

$$h(y) = [\text{if } \mu_2(y) > 0 \text{ then } \nu_2(y)/\mu_2(y), \text{ else } 1].$$

Obviously, g is polynomially bounded, $\mu_1 = g \cdot \nu_1$, f transforms μ_1 into ν_2, and $\nu_2 = h \cdot \mu_2$. We need only prove that h is polynomially bounded. Restrict attention to $y \in Y$. We have

$$\sum_{fx=y} \mu_1(y) = \nu_2(y) = h(y) \cdot \mu_2(y) = h(y) \cdot \sum_{fx=y} \nu_1(x) = h(y) \cdot \sum_{fx=y} (gx)^{-1} \cdot \mu_1(x).$$

Thus, $(hy)^{-1}$ is the conditional expectation $E[(gx)^{-1} \mid fx = y]$. Since g is polynomially bounded and f is honest, there exists a polynomial q such that $g(x) \leqslant q(|fx|)$. Then $(gx)^{-1} \geqslant 1/q(|fx|)$, and $(hy)^{-1} = E[(gx)^{-1} \mid fx = y] \geqslant 1/q(y)$, and $h(y) \leqslant q(y)$. Q.E.D.

Remark. The honesty condition cannot be dropped in Lemma 3.3(2). Consider a function $y = f(x)$ that takes a binary string x into the number $|x|$ written in the binary notation. For every $i > 1$, f transforms the probability function $\alpha_i(x)$ proportional to $|x|^{-i}$ to the probability function $\beta_i(y)$ proportional to y^{-i}. Since $\alpha_2 \leqslant \alpha_3$, $\alpha_2 \leqslant^f \beta_3$. But f transforms α_2 into β_2 which is not dominated by β_3.

DEFINITION. A Ptime computable function f *reduces* (D_1, μ_1) to (D_2, μ_2) if f reduces $D_1 \mid \{x : \mu_1(x) > 0\}$ to D_2 and $\mu_1 \leqslant^f \mu_2$.

LEMMA 3.4.

1. *If* (D_1, μ_1) *Ptime reduces to* (D_2, μ_2) *and* (D_2, μ_2) *is AP then* (D_1, μ_1) *is AP.*

2. *The Ptime reducibility relation on randomized decision problems is transitive.*

Proof. (1) Suppose that a Ptime computable function f reduces (D_1, μ_1) to (D_2, μ_2). Then there exists a probability function $\nu \geqslant \mu_1$ such that f transforms ν into a restriction of μ_2. Suppose that (D_2, μ_2) is AP. By Lemma 3.1(2), (D_1, ν) is AP. By Lemma 1.5, (D_1, μ_1) is AP.

(2) Suppose that f Ptime reduces (D_1, μ_1) to (D_2, μ_2) and g Ptime reduces (D_2, μ_2) to (D_3, μ_3). There exists a probability function $\nu_1 \geqslant \mu_1$ such that f transforms ν_1 into a restriction μ_2' of μ_2, and there exists a probability function $\nu_2 \geqslant \mu_2$ such that g transforms ν_2 to a restriction μ_3' of μ_3. If $\mu_1(x) > 0$ then $\nu_1(x) > 0$,

$\mu_2'(fx) > 0$ and $\mu_2(fx) > 0$; hence the composition $g \circ f$ reduces $D_1 | \{x : \mu_1(x) > 0\}$ to D_3. We have

$$\mu_1 \leqslant \nu_1 \xrightarrow{f} \mu_2' \leqslant \mu_2 \leqslant \nu_2 \xrightarrow{g} \mu_3'.$$

By Lemma 3.3(1), there exists $\nu \geqslant \nu_1$ such that f reduces ν to a restriction ν_2' of ν_2. Thus,

$$\mu_1 \leqslant \nu \xrightarrow{f} \nu_2' \leqslant \nu_2 \xrightarrow{g} \mu_3'.$$

Obviously, g transforms ν_2' to a restriction μ_3'' of μ_3'. Hence $g \circ f$ reduces μ_1 to μ_3. Q.E.D.

DEFINITION. A randomized decision problem (D, μ) is *Ptime hard* for RNP if every RNP problem Ptime reduces to (D, μ), and (D, μ) is *Ptime complete for* RNP if it is RNP and Ptime hard for RNP.

It is not obvious that there are Ptime complete problems for RNP.

4. RANDOMIZED HALTING PROBLEM

In this section, we prove that an arbitrary RNP problem reduces to a randomized version of the bounded halting problem for an appropriate nondeterministic Turing machine (shortly, NTM); for brevity, the adjective „bounded" is omitted. We restrict attention to NTMs with binary input alphabet (unless the contrary is said explicitly).

Randomized Halting Problem RH(M) for an NTM M

Instance. A binary string $w01^n$ with $n > |w|$.

Question. Is there a halting computation of M on w with at most n steps?

Probability. Proportional to $n^{-3}2^{-k}$, where $k = |w|$.

The probability function of RH(M) corresponds to the following experiment. First, randomly choose a positive integer n, then randomly choose a natural number $k < n$, and then randomly choose a binary string of length k.

DEFINITION. A positive integer n is *longevous* for an input w of an NTM M if every halting computation of M on w has $\leqslant n$ steps. A function $g(w)$ is a *longevity guard* for M if, for every input w, $g(w)$ is a number longevous for w. If g is a longevity guard for M, let RH(M, g) be the restriction of RH(M) to instances $w01^{g(w)}$.

THEOREM 4.1. *For every RNP problem (D, μ) there exist an NTM M and a longevity guard g for M such that (D, μ) Ptime reduces to RH(M, g).*

Proof. By the definition of RNP problems in Section 1, the probability function μ is dominated by some probability function μ_1 with Ptime computable distribution μ_1^*. By the definition of Ptime reducibility, (D, μ) Ptime reduces to (D, μ_1). By Lemma 3.4(2), we may assume that $\mu = \mu_1$. By Lemma 3.1(3), we may assume that instances of D are binary strings. By Lemma 1.6, we may assume that every value of μ is a positive binary fraction.

By the definition of RNP problems, the decision problem D is NP. Therefore there exists an NTM A_D such that:

- A_D has a halting computation on an arbitrary input w if and only if w is a positive instance of D, and
- A_D has a polynomially bounded longevity guard.

Let x' be the shortest binary string with $\mu^*(x) < 0 \cdot x'1 \leqslant \mu^*(x^+)$. Recall that x^+ is the successor of x in the lexicographical order. Then

$$0 \cdot x'1 - 2^{-|x'1|} \leqslant \mu^*(x) < \mu^*(x^+) < 0 \cdot x'1 + 2^{-|x'1|},$$

and therefore $2 \times 2^{-|x'1|} > \mu(x)$. Set

$$x'' = [\text{if } 2^{-|x|} > \mu(x) \text{ then } 0x, \text{ else } 1x'],$$

so that $2^{-|x''|} > \mu(x)/2$.

The desired reduction is

$$f(x) = x''01^{g(x'')},$$

where g is a longevity guard for the desired NTM M. Now we describe the desired NTM M. Given a binary bit b followed by a string w, M executes the following algorithm:

1. If $b = 0$ then
   if $2^{-|w|} \leqslant \mu(w)$ then loop forever else simulate A_D on w.
2. Find the unique x with $\mu^*(x) < 0 \cdot w1 \leqslant \mu^*(x^+)$.
3. If $2^{-|x|} > \mu(x)$ or $x' \neq w$ then loop forever, else simulate A_D on x.

M has a halting computation on x'' if and only if x is a positive instance of D. Ptime computability of μ^* is used on step 2. It is easy to see that M has a longevity guard g such that $g(x'')$ is bounded by a polynomial of $|x|$ (though not necessarily bounded by a polynomial of $|x''|$).

Finally, the probability function ν of $\mathrm{RH}(M, g)$ dominates μ with respect to f. For, $\nu(fx)$ is proportional to $g(x'')^{-3} 2^{-|x''|}$ which exceeds $g(x'')^{-3} \mu(x)/2$. Q.E.D.

COROLLARIES.

1. *There is an NTM M such that $RH(M)$ is Ptime complete for RNP.*

2. *Let ν be any positive probability function over NTMs. The following randomized decision problem is Ptime hard for RNP:*

Instance. An NTM M and an instance $w01^n$ of $\mathrm{RH}(M)$.

Question. Is there a halting computation of M on w with at most n steps?

Probability. Choose M with respect to ν and then choose an instance of $\mathrm{RH}(M)$ as above.

Proof. (1) Choose M to be a universal NTM. (2) Clear. Q.E.D.

Remark. Theorem 4.1 implies a similar theorem for the case of, say, ternary input alphabet. The proof illustrates how reductions of RNP problems differ from reductions of NP problems. The desired reduction transforms an instance $x01^m$ for the given $\mathrm{RH}(M, g)$ to an instance $y01^n$ for a new $\mathrm{RH}(M', g')$; here x is a binary string and y is a ternary string. Of course, x is also a ternary string, but y cannot be taken equal to x because the domination condition will be violated: The probability that a random ternary string happens to be binary approaches 0 exponentially (in the length of the string) fast. One possibility is to choose y in such a way that the number of x in the lexicographical order of binary strings equals the number of y in the lexicographical order of ternary strings.

In the rest of this section, we restrict attention to NTMs with a single tape, that is bounded on the left and unbounded to the right, and a single head; the input is left justified on the tape in the initial moment. Note that Theorem 4.1 survives the restriction. The following two lemmas are useful.

LEMMA 4.1. *Let F, G be Ptime computable functions from binary strings to binary strings such that $|F(w)| = O(\log_2 |w|)$ and $|G(w)| = O(\log_2 |w|)$. For every $RH(M_0, g_0)$, there exist an NTM M and a longevity guard g for M such that $RH(M_0, g_0)$ Ptime reduces to the restriction of $RH(M, g)$ to instances $w01^n$ where w starts with $F(w)$ and ends with $G(w)$.*

Proof. Given an input w, the desired M checks whether w has the form $F(w)\,uG(w)$. In the positive case, M simulates M_0 on u; otherwise it loops. The desired reduction takes $u01^m$ to $F(u)\,u(G(u)\,01^{p(m)}$, where p is an appropriate polynomial. The domination requirement is obviously satisfied. Q.E.D.

DEFINITION. An input w is *stable* for an NTM M if, for every natural number n, the following statements are equivalent:

1. There exists x such that M has a halting computation on wx with at most n steps, and
2. For every x, M has a halting computation on wx with at most n steps.

LEMMA 4.2. *For every $RH(M_0, g_0)$, there exist an NTM M and a longevity guard g for M such that $RH(M_0, g_0)$ Ptime reduces to the restriction of $RH(M, g)$ to stable instances (i.e., to instances $w01^{g(w)}$ where w is stable for M). Moreover, it may be*

required that 0 *and* 1 *are the only tape symbols of* M (*with* 0 *serving also as the blank*).

Remark. Note that a machine with binary input alphabet may have many tape symbols; in particular, the blank may differ from input symbols. The proof of Lemma 4.2 can be simplified if the restriction on the tape alphabet is removed.

Proof. Code tape symbols of M_0 with binary strings of some fixed length l such that the string 1^l, called \$ in this proof, is not a code. The desired M works as follows.

1. M verifies that the initial tape has a prefix

$$\$0a_1a_10a_2a_2\cdots0a_ka_k\$\$$$

for some k and some binary digits $a_1, ..., a_k$ with $a_1 = 1$; if not then M loops.

2. Let m be the positive integer with binary notation $a_1a_2\cdots a_k$, and u be the string $b_1b_2\cdots b_m$ such that the initial tape has a prefix $\$0a_1a_10a_2a_2\cdots0a_ka_k\$\$u$. Using the sequence of positions 2, 5, 8, ..., $3k-1$ of the string $\$\cdots\$\$ as a counter, M transforms

$$\$0a_1a_10a_2a_2\cdots0a_ka_k\$\$u \text{ into } u\$0a_1a_10a_2a_2\cdots0a_ka_k\$\$.$$

3. Using a counter again, M transforms

$$u\$0a_1a_10a_2a_2\cdots0a_ka_k\$\$ \text{ into } \$0^{3k}\$v_1v_2\cdots v_m\$,$$

where each v_i is the code for b_i.

4. Using the codes for tape symbols, M simulates M_0 pushing the rightmost \$ to the right if necessary.

The desired reduction is

$$f(u01^n) = \$0a_1a_10a_2a_2\cdots0a_ka_k\$\$u01^{p(n)},$$

where $a_1a_2\cdots a_k$ is the binary notation for $|u|$, and p is an appropriate polynomial. We ignore the case of $u = e$. It is obvious that the string $v = \$0a_1a_1\cdots0a_ka_k\$\$u$ is stable for M. To check the domination condition, note that $|v| = |u| + O(\log_2|u|)$. Q.E.D.

5. Randomized Post Correspondence Problem

In this section, a randomized version of the bounded Post Correspondence Problem (PCP) is defined and proved Ptime complete for RNP. PCP is a well-known undecidable decision problem [12]; it can be stated as follows.

Post Correspondence Problem

Instance. A nonempty list $L = \langle(u_1, v_1), ..., (u_s, v_s)\rangle$ of pairs of strings.

Question. Does there exist a function F from some nonempty interval $[1 \cdots k]$ of integers to the interval $[1 \cdots s]$ such that the concatenation of strings $u_{F(1)}, ..., u_{F(k)}$ coincides with the concatenation of strings $v_{F(1)}, ..., v_{F(k)}$?

If $u_{F(1)} \cdots u_{F(k)} = v_{F(1)} \cdots v_{F(k)}$, and $k > 0$ then F is called a *solution* of length k for the given instance L of PCP. According to Garey and Johnson [7] a bounded version of PCP has been proved NP complete by Constable, Hunt, and Sahni [6]. For brevity, we omit the adjective "bounded" in the following definition.

Randomized Post Correspondence Problem (RPCP)

Instance. A nonempty list $L = \langle (u_1, v_1), ..., (u_s, v_s) \rangle$ of pairs of binary strings, and the unary notation 1^n for a positive integer n.

Question. Is there a solution of length at most n for L?

Probability. Randomly and independently choose positive integers n and s, then randomly and independently choose binary strings $u_1, v_1, ..., u_s, v_s$.

In accordance with Section 2, the random choices are made with respect to the default, or standard, probability functions on positive integers and binary strings which were defined in Section 2. It is clear that RPCP is RNP. Call an instance $(L, 1^n)$ of RPCP robust if either L has no solution or it has a solution of length $\leqslant n$. Let RRPCP be the restriction of RPCP to robust instances.

THEOREM 5.1. *RRPCP is Ptime hard for RNP.*

Proof. The proof is an adaptation of the standard undecidability proof for PCP [12]; the difficulty is that the desired reduction should have the domination property.

Suppose that M is an arbitrary Ptime guarded NTM and g is a longevity guard for M. By Theorem 4.1, it suffices to reduce $\mathrm{RH}(M, g)$ to RRPCP. Let (D, μ) be the restriction of $\mathrm{RH}(M, g)$ to instances $w01^m$ such that w is not empty and starts with a 1; by Lemma 4.1, it suffices to reduce (D, μ) to RRPCP. Let σ be the number of control states of M and τ be the number of tape symbols of M.

LEMMA 5.1. *Let w be a nonempty binary string and l be the least even integer such that $2^{(l-6)/2} \geqslant |w| + \sigma + 2\tau + 2$. There exists a set S of binary strings of length l satisfying the following requirements.*

1. *No S-string is a substring of w.*
2. *If a nonempty suffix z of an S-string x is a prefix of an S-string y, then $z = x = y$.*
3. *Every S-string starts with* 01.
4. $|S| = \sigma + 2\tau + 2$.

Proof. Let R be the regular set $0100(00 + 11)^* 11$. The string w has $\leqslant |w|$ substrings of length l. The definition of l allows us to choose a set S of R-strings of length l that satisfies requirements (1) and (4). Then requirement (3) is satisfied.

To prove that requirement 2 is satisfied as well, suppose by contradiction that $x = a_1 \cdots a_l \in S$, $1 < i \leqslant l$, and $y = b_1 \cdots b_l = a_i \cdots a_{l+i-1} \in S$. Since x, y belong to R, they satisfy the following: If $1 < j \leqslant l$ and j is odd then $j < l$, $a_j = a_{j+1}$, and $b_j = b_{j+1}$. If $i = l$ then $0 = b_1 = a_l = 1$; hence $i < l$. Since $a_l = b_1 = 0$ and $a_{i+1} = b_2 = 1$, i is even. Since $i+1$ is odd, $a_{i+1} = a_{i+2} = 1$. But $a_{i+2} = b_3 = 0$. This gives the desired contradiction. Q.E.D.

LEMMA 5.2. *Every binary string x that does not start with* 01 *and is different from* 0 *is a concatenation of strings* 00, 000, 1, *and* 10.

Proof. We prove the lemma by induction on $|x|$. The case $|x| \leqslant 3$ is easy. Suppose that $|x| > 3$. It suffices to prove the existence of strings y, z such that $x = yz$, y is one of the 4 strings 000, 00, 1, 10 and z does not start with 01 and is different from 0.

If x starts with:	then the desired y is:
0000	00
0001	000
001	00
100	1
101	10
11	1

Q.E.D.

In Section 4, at the beginning and right before Lemma 4.1, we restricted the class of NTMs under consideration. Without loss of generality, we may suppose additionally that our M uses a blank symbol which is different from input symbols and that, on every step, the head of M prints a nonblank symbol in the currently scanned cell and moves one cell to the left or right. It follows that the nonblank portion of the tape is always an initial segment of the tape. In addition, we may suppose that there is only one halting state, and in any halting configuration the first, i.e., the leftmost, blank is observed.

Let w be an instance of D and let l and S be as in Lemma 5.1. Use σ members of S to code state symbols of M, and let s, h be the codes for the initial and the halting states of M respectively. Use 2τ additional members of S to assign two binary codes X' and X'' to each tape symbol X of M. In particular, we have $0'$, $0''$, $1'$, $1''$; let B' and B'' be the two codes for the blank symbol B of M. We use X^+ as a variable over (X', X''). Finally, let % and s_0 be the two remaining members of S.

For every w, let $L = L(w)$ be an instance of PCP comprising the following pairs of binary strings:

L0. $(\%, \%ws_0)$.

L1. The four pairs (u, v) such that $u \in \{000, 00, 1, 10\}$ and v is obtained from u by replacing symbols 0, 1 with strings $0'$, $1'$ respectively.

L2. The pair (s_0, sB').

L3. Pairs (X', X''), (X'', X') for every tape symbol X of M.

L4.1. Pairs $(pX^+, Y'q)$ for each instruction $[pX \to qYR]$ of M.
L4.2. Pairs $(Z^+pX^+, qZ'Y')$ for each instruction $[pX \to qYL]$ of M.
L4.3. Pairs $(pB^+, Y'qB')$ for each instruction $[pB \to qYR]$ of M.
L4.4. Pairs $(Z^+pB^+, qZ'Y'B')$ for each instruction $[pB \to qYL]$ of M.
L5. Pairs (X^+h, h).
L6. Pairs (hB^+B', B').

It is convenient to view the problem of solving L as a derivation problem with pairs L0–L6 as rules of inference. In this connection, we need a few definitions.

Two binary strings are *compatible* if one of them is a prefix of the other. Pairs (x_1, y_1) and (x_2, y_2) of strings are *equivalent* if there exist strings u, v, and x_3, y_3 such that $x_1 = ux_3$, $y_1 = uy_3$, $x_2 = vx_3$, and $y_2 = vy_3$. A pair (x, y) of binary stings is *unary* if either x or y is empty. It is easy to see that every pair of compatible strings is equivalent to a unary pair.

A pair (x_1, y_1) *yields* a pair (x_2, y_2) *in one step* if there is a pair (u, v) in L0–L6 such that (x_1u, y_1v) is equivalent to (x_2, y_2). The *yield* relation on pairs is the transitive closure of the yield-in-one-step relation. We identify a string x with the unary pair (e, x). This extends the yield relation to strings.

LEMMA 5.3. *$L(w)$ has a solution of length $1+k$ if and only if ws_0 yields the empty string e in k steps.*

Proof. L has a solution of length $1+k$ if and only if e yields e in $1+k$ steps. Since $(\%, \%ws_0)$ is the only compatible pair in L, e yields e in $1+k$ steps if and only if ws_0 yields e in k steps. Q.E.D.

For each binary string x, let x' (resp. x'') be the binary string obtained from x by replacing each 0 with 0′ (resp. 0″) and each 1 with 1′ (resp. 1″).

LEMMA 5.4.

1. *ws_0 yields s_0w' in $\leqslant |w|$ steps.*

2. *Any derivation of e from ws_0 splits into two parts: a derivation of s_0w' and a subsequent derivation of e from s_0w'.*

Proof. (1) Use Lemma 5.2 and rules L2.

(2) Consider the given derivation. First some pair (u, ws_0u') is derived by means of L1-rules and then some other rule (x, y) is applied to that pair. Obviously x belongs to S and ux is a prefix of ws_0u'.

Recall that S satisfies the four requirements of Lemma 5.1. If ux is a prefix of w then x is a substring of w which contradicts requirement 1. Hence $|ux| > |w|$. The string u' is a concatenation $s_1 \cdots s_k$ where each s_i is either 0′ or 1′. Let $w_0 = w$ and $w_{i+1} = w_is_i$ for $0 < i < k$, and let i be the least number such that $|w_i| < |ux| \leqslant |w_{i+1}|$. Then a nonempty suffix of x is a prefix of s_i. Since S satisfies requirement 2 of Lemma 5.1, $u = w_i$.

If $i>0$ then (u, ws_0u') is equivalent to a nonempty concatenation of strings $0'$ and $1'$. Only rules L3 are applicable to concatenations of strings $0'$ and $1'$, and all rules L3 are length preserving. It follows that, in the case $i>0$, the pair (u, ws_0u') does not derive e. Hence $i=0$, $u=w$ and (u, ws_0u') is equivalent to s_0w'. Q.E.D.

If x is a string of state or tape symbols of M, let x^+ denote any of the binary strings obtained from x by replacing each occurrence of every symbol by an S-string that codes the symbol.

LEMMA 5.5. *There exists exactly one derivation of length* $|w|+1$ *from* s_0w', *and the result of that derivation is* $sw''B'$.

Proof. Apply the L2-rule to derive $w'sB'$ from s_0w'. Then use $|w|$ applications of L3-rules to derive $sw''B'$ from $w'sB'$. The uniqueness is obvious. Q.E.D.

If at moment t (i.e., after t steps of computation), the state of M is q, the head of M is at cell number i, and the first blank is in cell number j, then the configuration of M at moment t may be represented by a string xqy, called the instantaneous description or ID, where x and y are the strings in the segments $[1\cdots(i-1)]$ and $[i\cdots j]$ of the tape respectively. We identify states of M with their binary codes. Thus, the initial ID of M on input w is swB.

LEMMA 5.6. *Let* $t'=\max(t, |w|)$.

1. *There exists a polynomial* p_1 *such that if an ID* xqy *is reachable from the initial ID* swB *in* t *steps then every* sw^+B^+ *yields some* x^+qy^+ *within* $p_1(t')$ *steps.*

2. *There exists a polynomial* p_2 *such that if* M *has a halting computation of length* t *then every* sw^+B^+ *yields the empty string within* $p_2(t')$ *steps.*

Proof. (1) An easy induction on t. The simulation of a step of M from a configuration $x_1q_1y_1$ comprises of $\leqslant |x_1|$ applications of L3-rules, followed by one application of an L4-rule, followed by $|y|-1$ applications of L3-rules. It remains to note that the length of the ID at moment t is bounded by $t'+1$.

(2) If a halting configuration xhB is reachable in t steps, then, by (1), some x^+hB^+ is derivable from any sw^+B^+ in $\leqslant p_1(t)$ steps. If $y=zX$, where X is a tape symbol of M, then y^+hB^+ yields z^+hB^+ by means of $|z|$ applications of rules L3, followed by one application of an L5-rule, followed by an additional application of an L3-rule. This shows that x^+hB^+ yields hB^+ and allows to estimate the derivation length. Finally, hB^+ yields e by means of one application of an L-rule. Q.E.D.

It is easy to see that any derivation from any sw^+B^+ can use only rules L3–L6. For, consider the collection K of strings x and unary pairs (x, e) such that x is a concatenation of the codes for state and tape symbols. K contains all strings sw^+B^+, and is closed under rules L3–L6, and only rules L3–L6 are applicable to members of K.

LEMMA 5.7.

1. *Every string, derived from any* sw^+B^+ *by means of rules* L3-L4 *has the form* $x_2^+ qy^+ x_1^+$, *where* x_1x_2qy *is a reachable ID of* M, *or the form* $y_2^+ x^+ qy_1^+$, *where* xqy_1y_2 *is a reachable ID of* M.

2. *If some* sw^+B^+ *yields* e *then* M *halts on* w.

Proof. (1) Induction on the length of the derivation.

(2) Suppose that sw^+B^+ yields e. Since no L3 or L4 rule shortens strings, rules L5 or L6 should be used in the derivation. Hence sw^+B^+ yields some string $x_2^+ hu^+ x_1^+$ or $y_2^+ x^+ hy_1^+$. By (1), there is a halting computation of M on w. Q.E.D.

LEMMA 5.8. *There exists a polynomial* p *such that, for every instance* $w01^m$ *of* D, *the following statements are equivalent*:

1. M *has a halting computation of length* $\leqslant m$ *on* w.
2. $L(w)$ *has a solution of length* $\leqslant p(m)$.
3. $L(w)$ *has a solution.*
4. M *has a halting computation on* w.

Proof. First we prove that (1) implies (2) for an appropirate p. Suppose that M has an m-step halting computation on w. By Lemma 5.3, we need to show that ws_0 yields e in $<p(m)$ steps for some polynomial p. By Lemma 5.4(1), ws_0 yields s_0w' in a number of steps which is at most $|w|$ and therefore less than m. By Lemma 5.5, s_0w' yields $sw''B'$ in $|w|+1 \leqslant m$ steps. Now use Lemma 5.6.

Obviously, (2) implies (3).

To prove that (3) implies (4), suppose that $L(w)$ has a solution. By Lemma 5.3, ws_0 yields e. By Lemma 5.4(2), s_0w' yields e. By Lemma 5.5, $sw''B'$ yields e. Now use Lemma 5.7(2).

Since (D, μ) is a restriction of $\mathrm{RH}(M, g)$, m is longevous for w. Hence (4) implies (1). Q.E.D.

Let p be as in Lemma 5.8. The desired reduction reduction of (D, μ) to RRPCP is

$$f(w01^m) = (L(w),\ 1^{p(m)}).$$

By Lemma 5.8, $w01^m$ is a positive instance of D if and only if $f(w01^m)$ is a positive instance of RRPCP. It remains to check that the probability function ν of RRPCP dominates μ. Since (D, μ) is a restriction of $\mathrm{RH}(M, g)$, $\mu(w01^m)$ is proportional to $m^{-3}2^{-|w|}$. We must prove that, for some polynomial r, $r(m) \times \nu(f(w01^m))$ exceeds $m^{-3}2^{-|w|}$.

Let $\delta(x)$ be the default probability of a binary string x. Let u range over the binary strings of $L(w)$ different from $\%ws_0$. $\nu(f(w01^m))$ is the product of $p(m)^{-2}$ and $\delta(\%ws_0)$ and all $\delta(u)$. It suffices to prove that:

- There exists a polynomial r_1 such that $r_1(m) \times p(m)^{-2} > m^{-3}$,
- there exists a polynomial r_2 such that $r_2(m) \times \delta(\%ws_0) > 2^{-|w|}$, and
- there exists a polynomial r_3 such that $r_3(m) \times \delta(u) > 1$ for all u.

All three claims are easy. Use the fact the length l of S-strings is $O(\log_2 |w|)$. Theorem 4.1 is proved. Q.E.D.

COROLLARY. *RPCP is Ptime complete for RNP.*

Remark 5.1. The reason for introducing robust instances was to make the completeness proof a little easier. It is possible also that the robustness may be helpful in reducing RPCP to other problems. In this connection, let us note that the definition of robust instances $(L, 1^n)$ may be strengthened by requiring that every solution for L should be of length $\leqslant n$. Theorem 5.1 remains true and the particular reduction, described in the proof of Theorem 5.1, is fine. Lemma 5.7 should be strengthened by asserting that longer derivations correspond to longer computations.

Remark 5.2. In the classical reduction of the halting problem to PCP [12], an input w of the given Turing machine appears in a coded form in the corresponding instance of PCP. We must use an essentially uncoded form of w in order to take care about probabilities. Rules L2 are used to rewrite w in a coded form. The four L2-rules cannot be replaced by two simpler rules $(0, 0')$ and $(1, 1')$ because the new rules may be applicable in inappropriate situations.

In the rest of this section, we slightly modify the proof of Theorem 5.1 and prove the RNP hardness of another form of RPCP; that result is used in the next section. Let x^{-1} denote the reverse of binary string x.

LEMMA 5.1′. *Let w be a nonempty binary string and l be the least even integer such that $2^{(l-6)/2} \geqslant 2|w| + \sigma + 2\tau + 2$. There exists a set S of binary strings of length l satisfying the four requirements of Lemma* 5.1 *plus the following two additional requirements*:

5. *For no S-string x, x^{-1} is a substring of w.*
6. *If x, y, z are s-strings then z^{-1} is not a substring of xy.*

Proof. Let R be as in the proof of Lemma 5.1. The number of substrings of w of length l plus the number of substrings of w^{-1} of length l is at most $2|w|$. The definition of l allows us to choose a set S of R-strings of length l that satisfies requirements (1), (4), and (5). Requirement (3) is obviously satisfied. The same proof as before establishes that S satisfies requirement (2).

By contradiction, suppose that x, y, z witness that S fails to satisfy requirement (6). Let $xy = a_1 \cdots a_{2l}$ and $z^{-1} = b_1 \cdots b_l = a_i \cdots a_{l+i-1}$. Since z^{-1} starts with 11 and x starts with 0100, $i > 4$. If i is odd then xy has a 1 in the odd position $l+i-2$ followed by a 0 in the even position $l+i-1$ which is impossible. Hence i is even.

By induction on j, $i \leqslant j \leqslant l+1$, we check that $a_j = 1$. If $j = i$ then $a_j = b_1 = 1$. If j is odd and $a_j = 1$ then $a_{j+1} = 1$ because $x, y \in R$. If j is even and $a_j = 1$ then $j-i+1$ is odd, $j-i+1 < j-4+1 < l-2$ and $b_{j-i+1} = 1$; hence $b_{j\ i} = 1$ because $z \in R$ and hence $a_{j+1} = 1$. In particular, $a_{l+1} = 1$ which is impossible. Q.E.D.

The proof of Theorem 5.1 remains valid if Lemma 5.1 is replaced with Lemma 5.1′.

Define the length of a pair (x, y) of binary strings to be the difference $|y| - |x|$. Call an instance L of PCP *positively biased* if $|u_1 \cdots u_k| \leqslant |v_1 \cdots v_k|$ whenever $u_1 \cdots u_k$ and $v_1 \cdots v_k$ are compatible and each (u_i, v_i) belongs to L_i.

LEMMA 5.9. *The instances of* PCP *constructed in the modified proof of Theorem* 5.1 *are positively biased.*

Proof. By contradiction, suppose that an instance $L(w)$ is not positively biased. Then e yields a negative pair N. Then ws_0 yields N. An argument similar to the proof of Lemma 5.4(2) establishes that $s_0 w'$ yields N. By Lemma 5.5, $sw''B'$ yields N. As it has been proved, any derivation from $sw''B'$ uses only rules L3–16. Length-decreasing rules should be used in order to derive N. All length decreasing rules involve h. By Lemma 5.7, some $u^+ hv^+$ yields N. Here uv is a string of tape symbols. It is easy to see that $u^+ hv^+$ does not yield any negative pair. Q.E.D.

We say that an instance L of PCP is *palindrome sensitive* if there is no palindrome $u_1 \cdots u_k v_k^{-1} \cdots v_1^{-1}$ where each (u_i, v_i) belongs to L and $|u_1 \cdots u_k| \neq |v_1 \cdots v_k|$.

THEOREM 5.2. *The restriction of RPCP to instances* $(L, 1^n)$ *such that* L *is palindrome sensitive is hard for RNP.*

Proof. It suffices to check that instances $L(w)$ constructed in the modified proof of Theorem 5.1 are palindrome sensitive. Without loss of generality, we may suppose that $|w| \geqslant l$. By contradiction suppose that pairs $(u_1, v_1), \ldots, (u_k, v_k)$ witness the failure of palindrome sensitivity of some $L(w)$. We know that $u_1 = \%$ and $v_1 = \%ws_0$. Let $u = u_2 \cdots u_k$ and $v = v_2 \cdots v_k$. By Lemma 5.9, $|\%u| < |\%ws_0 v|$. Hence there exists a palindrome x such that $ux = ws_0 v$ and ws_0 yields x by means of rules L1–L6. It follows that v is a nonempty concatenation of S-strings. It is easy to see that x starts with a reverse r_1 of an S-string. By the choice of S, r_1 cannot be a substring of w. Hence w is a proper prefix of ur_1.

First suppose that u is a proper prefix of w, so that x has a suffix $s_0 v$ and therefore it has a prefix $r_1 r_2$, where r_2 is a reverse of an S-string. But then r_2 is a substring of the concatenation $s_0 v$ of S-strings which contradicts the choice of S.

Thus, w is a prefix of u. Then x is a suffix of $s_0 v$. If only L2-rules, which increase the positive balance, were used to derive x then $|x| \geqslant 2l$ and x is the concatenation of a suffix of an S-string and at least two s-strings. By the choice of S, such a

concatenation cannot be a palindrome. Thus, at least one of the rules L3–L6 was used to derive x. The left string of any such rule starts with an S string. By the choice of S (use requirements 1 and 2), x is a concatenation of S-strings. Hence x is not a palindrome. Q.E.D.

6. Additional RNP Complete Problems

Randomized Palindrome Problem

Instance. A context-free grammar with productions

$$T \to u_1 T v_1 \mid \cdots \mid u_s T v_s \mid e,$$

and the unary notation 1^n for a positive integer number n. Here u_i and v_i are binary strings, and e is the empty string.

Question. Is it possible to derive a nonempty palindrome (in terminal symbols 0 and 1) in at most n steps?

Probability. Randomly and independently choose positive integers n and s, then randomly and independently choose binary strings $u_1, v_1, \ldots, u_s, v_s$.

Theorem 6.1. *Randomized Palindrome Problem is Ptime complete for RNP.*

Proof. It is obvious that the problem is RNP. To prove that it is hard for RNP, we reduce the palindrome sensitive version of PCP (see Theorem 5.2) to Randomized Palindrome Problem. Given a palindrome sensitive instance $L = \langle (u_1, v_1), \ldots, (u_s, v_s) \rangle$ of PCP and some 1^n, the desired reduction produces a grammar G with productions

$$T \to u_1 T v_1^{-1} \mid \cdots \mid u_s T v_s^{-1} \mid e,$$

and the unary notation for $n+1$. The domination requirement is obvious. We must check that L has a solution of length $\leqslant n$ if and only if G produces a nonempty palindrome in at most $n+1$ steps.

It is clear that every solution

$$u_{F(1)} \cdots u_{F(k)} = v_{F(1)} \cdots v_{F(k)}$$

for L gives rise to a $(k+1)$-step derivation of the Palindrome

$$u_{F(1)} \cdots u_{F(k)} v_{F(k)}^{-1} \cdots v_{F(1)}^{-1}.$$

Suppose that G produces a nonempty Palindrome

$$u_{F(1)} \cdots u_{F(k)} v_{F(k)}^{-1} \cdots v_{F(1)}^{-1}$$

in $k+1$ steps. Since L is palindrome sensitive,

$$u_{F(1)} \cdots u_{F(k)} = v_{F(1)} \cdots v_{F(k)}.$$ Q.E.D.

It is easier to find complete RNP problems of logical nature. In this connection, we give two relatively straightforward theorems.

Let ϕ be a first-order sentence with order relation $<$, a unary predicate symbol P, and a collection σ of additional predicate symbols. Restrict attention to finite structures S with order such that the universe of S is an initial segment $[0 \cdots n-1]$ of natural numbers and the order is standard. Define the *randomized satisfiability problem* RSAT(ϕ):

Instance. The unary notation for a positive integer n, a natural number $k < n$, and a unary relation P_0 on $[0 \cdots k-1]$.

Question. Is there a model for ϕ on $[0 \cdots n-1]$ such that P coincides with P_0 on $[0 \cdots k-1]$?

Probability. The probability of the given instance is proportional to $n^{-3}2^k$ and corresponds to the following experiment: Randomly choose n, then randomly choose k, then randomly choose P_0.

Obviously, RSAT (ϕ) is RNP.

THEOREM 6.2. *For every RNP problem* (D, μ), *there exists a first-order sentence* $\phi(P)$ *such that* (D, μ) *reduces to* $RSAT(\phi)$.

Proof. Use Theorem 4.1. Q.E.D.

Remark. Utilizing known undecidability proofs, one can put severe syntactical restrictions on ϕ.

Let ψ be a sentence in the first-order language of arithmetic enriched with an additional unary relation P. Define the *randomized arithmetical satisfiability problem* RAS(ψ):

Instance. The unary notation for a positive integer n, a natural number $k < n$, and a unary relation P_0 on $[0 \cdots k-1]$.

Question. Is there an extension P of P_0 to $[0 \cdots n-1]$ such that $\psi(P)$ holds in the arithmetic modulo n?

Probability. Proportional to $n^{-3}2^k$.

Obviously, every RAS($\psi(P)$) is RNP.

THEOREM 6.2. [Gurevich and Shelah, 11]. *Every RNP problem Ptime reduces to some* $RAS(\psi)$.

Proof. Use Theorem 6.1. Q.E.D.

7. APtime Reducibility

In this section, the expression $\mu_1 \leqslant \mu_2$ denotes that μ_1 is weakly dominated by μ_2. The notions of weak domination and rarity function were defined in Section 1.

LEMMA 7.1. *Let* $\mu_1 \leqslant \mu_2$.

1. *If* ρ *is a rarity function for* μ_2, *then there exists* $\varepsilon > 0$ *such that* ρ^ε *is a rarity function for* μ_1.

2. *Every function polynomial on* μ_2*-average is polynomial on* μ_1*-average.*

3. *Every function computable in APtime with respect to* μ_2 *is so with respect to* μ_1.

4. *If* (D, μ_2) *is AP then* (D, μ_1) *is AP.*

5. *If* $\mu_2 \leqslant \mu_3$ *then* $\mu_1 \leqslant \mu_3$.

Proof. (1) Since $\mu_1 \leqslant \mu_2$, there exists a linear on μ_1-average function g such that some g^j witnesses that $\mu_1 \leqslant \mu_2$. Let $\varepsilon = 1/(j+1)$. We prove that ρ^ε is a rarity function for μ_1. It suffices to prove that $\sum_{\rho(x)^\varepsilon > g(x)} \rho(x)^\varepsilon \mu_1(x)$ finite. But $\rho(x)^\varepsilon > g(x)$ if and only if $\rho(x)^{j/(j+1)} > g(x)^j$ if and only if $\rho(x)^{1-1/(j+1)} > g(x)^j$ if and only if $\rho(x) > \rho(x)^\varepsilon g(x)^j$. Further,

$$\sum_{\rho(x)^\varepsilon > g(x)} \rho(x)^\varepsilon \mu_1(x) \leqslant \sum_{\rho(x)^\varepsilon > g(x)} \rho(x)^\varepsilon g(x)^j \mu_2(x) \leqslant \sum_x \rho(x) \mu_2(x) < \infty.$$

(2) is exactly Lemma 1.5. We give here an alternative proof. Suppose that a function f is polynomial on μ_2-average. By Proposition 1.2, there exists a rarity function ρ for μ_2 such that $f(x)$ is bounded by a polynomial of $|x|$ and $\rho(x)$. By (1), some $\rho^{1/k}$ is a rarity function for μ_1. Obviously, $f(x)$ is bounded by a polynomial of $|x|$ and $\rho(x)^{1/k}$. By Proposition 1.2, f is polynomial on μ_1-average.

(3) By (2), time polynomial in μ_2-average is polynomial in μ_1 average.

(4) To decide D means to compute the characteristic function of D. Use (3)

(5) Let f and g witness that $\mu_1 \leqslant \mu_2$ and $\mu_2 \leqslant \mu_3$ respectively. By (2), g is polynomial on μ_1-average. By Lemma 1.1, the product $f(x)\, g(x)$ is polynomial on μ_1-average. But the product witnesses that $\mu_1 \leqslant \mu_3$:

$$f(x)\, g(x)\, \mu_3(x) \leqslant f(x)\, \mu_2(x) \leqslant \mu_1(x). \qquad \text{Q.E.D.}$$

DEFINITION. A randomized decision problem (D_1, μ_1) is weakly transformable into a randomized decision problem (D_2, μ_2) if some APtime computable function reduces $D_1 | \{x : \mu_1(x) > 0\}$ to D_2, and transforms μ_1 into μ_2.

LEMMA 7.2. *If an APtime function f transforms (D_1, μ_1) into a restriction of (D_2, μ_2) and (D_2, μ_2) is AP then (D_1, μ_1) is AP.*

Proof. This is Lemma 3.1(2). Q.E.D.

LEMMA 7.3.

1. *μ_2 dominates μ_1 with respect to a one-to-one APtime function f if and only if the probability function $\nu(x)$ proportional to $\mu_2(fx)$ weakly dominates μ_1.*

2. *Suppose that a function f transforms μ_1 into some ν_2 which is weakly dominated by μ_2. If f is polynomial on μ_1-average then μ_2 dominates μ_1 with respect to a function f.*

Proof. (1) Similar to the proof of Lemma 3.2.

(2) As in the proof of Lemma 3.3(1), we may assume that $\mu_2(y) = 0$ whenever $\nu_2(y) = 0$. Let g witness that $\nu_2 \leqslant \mu_2 : g$ is polynomial on ν_2-average and $\nu_2(y) = g(y) \cdot \mu_2(y)$. Define ν_1 as in the proof of Lemma 3.3(1). We have $\mu_1(x) = g(f(x)) \cdot \nu_1(x)$. By Lemma 1.2(1), the function $g(f(x))$ is polynomial on μ_1-average, and therefore $\mu_1 \leqslant \nu_1$. It remains to check that f transforms ν_1 into μ_2; this is done exactly as in the proof of Lemma 3.3(1). Q.E.D.

DEFINITION. A function f *reduces* (D_1, μ_1) to (D_2, μ_2) *in APtime* if f reduces $D_1 | \{x : \mu_1 x > 0\}$ to D_2, f is computable in time polynomial on μ_1-average, and μ_2 dominates μ_1 with respect to f.

LEMMA 7.4.

1. *If (D_1, μ_1) APtime reduces to (D_2, μ_2) and (D_2, μ_2) is AP then (D_1, μ_1) is AP.*

2. *APtime reducibility is transitive.*

Proof. (1) Suppose that a function f reduces (D_1, μ_1) to (D_2, μ_2) in APtime and (D_2, μ_2) is AP. Then there exists a probability function $\nu \geqslant \mu_1$ such that f transforms ν into a restriction of μ_2. Suppose that (D_2, μ_2) is AP. At this point the similarity to the proof of Lemma 3.4(1) ends. We cannot use Lemma 7.2 to deduce that (D_1, ν) is AP because we do not know whether f is polynomial on ν-average; we know only that f is polynomial on μ_1-average.

Define $X' = \{x : \mu_1(x) \geqslant \nu(x)\}$, $X'' = \{x : \mu_1(x) < \nu(x)\}$, $\mu_1' = \mu_1 | X'$ and $\mu_1'' = \mu_1 | X''$. It suffices to prove that both (D_1, μ_1') and (D_1, μ_1'') are AP.

The case of μ_1'. Let $\nu' = \nu | X'$. We have $\nu' \leqslant \mu_1' \leqslant \mu_1$. By Lemma 7.1, f is polynomial on ν'-average. The function f transforms ν' to a restriction of μ_2. By Lemma 7.2, (D_1, ν') is AP. Since $\mu_1 \leqslant \nu$, $\mu_1' \leqslant \nu'$. By Lemma 7.1, (D_1, μ_1') is AP.

The case of μ_1''. Define $\nu_2(y) = \sum_{fx=y} \mu_1''(x)$. Then $\nu_2(y) \leqslant \sum_{fx=y} \nu(x)$ and therefore $\nu_2 \leqslant \mu_2$. By Lemma 7.1, (D_2, ν_2) is AP. By Lemma 7.2, (D_1, μ_1'') is AP.

(2) The proof is similar to that of Lemma 3.4(2). Q.E.D.

8. Incompleteness

We give a sufficient condition for a randomized decision problem to be incomplete for RNP with respect to APtime reductions.

Definition. A probability function μ on some Σ^* is *flat* if there exists a real number $\varepsilon > 0$ such that

$$\mu(x) \leqslant 2^{-n^\varepsilon}, \qquad \text{i.e., } -\log_2 \mu(x) \geqslant n^\varepsilon$$

for every Σ-string x of sufficiently big length n. A randomized decision problem (D, μ) is *flat* if μ is.

The intuition is that all values of a flat probability function are relatively small; none of them juts out.

In this section, the term "exponential" is used in a broader sense, and a function f from some Σ^* to nonnegative reals is *exponential* (or *exponentially bounded*) if there is a polynomial p with $f(x) \leqslant 2^{p(|x|)}$. The decision problem D for a language $L(D)$ over some alphabet $\Sigma(D)$ is DEXPtime (resp. NEXPtime) if some exponential-time deterministic (respectively nondeterministic) Turing machine decides D. Obviously, every NP problem is DEXPtime.

Theorem 8.1. *Let (D, μ) be a flat randomized decision problem where D is DEXPtime. If (D, μ) is APtime hard for RNP then NEXPtime = DEXPtime.*

Proof. We assume that (D, μ) is APtime hard for RNP and show that an arbitrary NEXPtime decision problem D_0 is DEXPtime decidable. Without loss of generality, instances of D_0 are binary strings. Let x range over binary string and $n = |x|$. We turn D_0 into a randomized decision problem (D_0, μ_0) by assigning to each x the default probability $\mu_0(x)$ proportional to $n^{-2}2^{-n}$.

Fix a polynomial $p(n) > n$ such that some $2^{p(n)}$-time-bounded NTM decides D_0. For every binary string x, let x' be the binary string of length $2^{p(n)}$ obtained from $x0$ by adding a tail of ones. Let D_1 be the decision problem for the language $\{x' : x \in L(D_0)\}$, and let μ_1 be the probability function on binary strings such that $\mu_1(x') = \mu_0(x)$ and $\mu_1(y) = 0$ if there is no x such that $y = x'$.

Lemma 8.1. *If a function g from binary strings to nonnegative reals is polynomial on μ_1-average then $g(x')$ is exponential in n.*

Proof. It suffices to consider the case when g is linear on average. For some c, we have

$$c > \sum_y g(y) \cdot |y|^{-1} \cdot \mu_1(y) = \sum_x g(x') \cdot |x'|^{-1} \cdot \mu_1(x').$$

Hence, for each x, $g(x') < c \cdot 2^{p(n)} \cdot n^2 2^n$. Q.E.D.

Since (D_1, μ_1) is RNP, there is an APtime reduction f of (D_1, μ_1) to (D, μ). This gives the following decision algorithm for D_0: Given x, compute x', then compute $f(x')$, and then solve the instance $f(x')$ of D. We need to prove only that the instance $f(x')$ of D is decidable in time exponential in n. Since D is DEXPtime, it suffices to show that $|f(x')|$ is bounded by a polynomial of n. Since f is APtime, μ weakly dominates μ_1 with respect to f; i.e., there exist a probability function ν and a polynomial on μ_1-average function g such that $g(x')\,\nu(x') = \mu_1(x')$ and f transforms ν into a restriction $\mu|Z$ of μ of some probability c. Then

$$g(x') \cdot \mu(fx')/c = g(x') \cdot (\mu|Z)(fx') = g(x') \cdot \sum_{ft' = fx'} \nu(t') \geqslant g(x') \cdot \nu(x') = \mu_1(x').$$

Since $g(x')$ is exponential in n (Lemma 8.1) and $(\mu_1(x'))^{-1}$ is exponential in n, there exists some polynomial $q(n)$ such that $\mu(fx') > 2^{-q(n)}$, i.e., $-\log_2 \mu(fx') < q(n)$. Since μ is flat, there is k such that

$$|fx'|^{1/k} \leqslant -\log_2 \mu(fx') < q(n), \qquad \text{i.e., } |fx'| < q(n)^k. \qquad \text{Q.E.D.}$$

The following lemma illustrates how prevalent flat probability functions are.

LEMMA 8.2. *Let μ be any probability function on graphs such that, for each n, the restriction of μ to graphs with n vertices is determined by edge-probability $f(n)$ such that*

$$n^{-2+\varepsilon} < f(n) < 1 - n^{-2+\varepsilon}$$

for some fixed $\varepsilon > 0$. Then μ is flat.

Proof. Let $r = n^{2-\varepsilon}$. For every graph G with n vertices,

$$\mu(G) < (1-1/r)^{n(n-1)/2} = [(1-1/r)^r]^{n(n-1)/(2r)} e^{-n(n-1)/(2r)}.$$

Hence

$$-\log_e \mu(G) > n(n-1)/(2r) \approx n^{\varepsilon}/2. \qquad \text{Q.E.D.}$$

COROLLARY. *Let μ be any probability function on graphs such that, for each n, the restriction of μ to graphs with n vertices is determined by an edge-probability $f(n)$.*

The μ-randomization of Hamiltonian Circuit Problem is not APtime hard for RNP unless NEXPtime = DEXtime.

Proof sketch. Choose a sufficiently small $\varepsilon > 0$ and design algorithms which solve HCP in expected polynomial time if the edge probability is at most $n^{-2+\varepsilon}$ or at least $1 - n^{-2+\varepsilon}$. Use these algorithms to reduce the μ-randomization of HCP to a flat problem. Q.E.D.

The randomized halting problems RH(M) for an NTM M is not flat because an input w for M may be very short comparative to the prescribed number n of steps. For every $r > 1$, the restriction of RH(M) to inputs $w01^n$ such that $|w|^r > n$ is flat. Similarly, the Randomized Post Correspondence Problem and Randomized Tiling Problem are not flat, but their natural restrictions are flat. For example, the restriction of Randomized Tiling to inputs $\langle T, 1^n, k, \rho \rangle$ such that ρ is of length $\geqslant n^{1/r}$ for some fixed r is flat.

The following theorem of Ben-David and Michael Luby [4] shows that the question DEXPtime = ? NEXPtime is related to the question whether AP includes RNP. See [1] in this connection.

THEOREM 8.2. *If AP includes RNP then DEXPtime = NEXPtime.*

Proof. Let E be the decision problem for some NEXPtime language $L(E)$. Let x range over instances of E and n be the number of x in the lexicographical order of instances of E. Let D be the decision problem for language $L(D) = \{1^n : x \in L(E)\}$ over the unary alphabet, and let $\mu(1^n)$ be the standard probability of natural number n. Obviously, (D, μ) is RNP.

Suppose that AP includes RNP. Then some algorithm A decides D in time T polynomial on μ-average. We prove that E is DEXPtime. It suffices to prove that $T(1^n)$ is bounded by a polynomial of n. For, in this case, the obvious algorithm for E—given x, compute 1^n, and then use A to solve 1^n—works in time bounded by an exponential function of $|x|$.

Let k witness that T is polynomial on μ-average:

$$\sum (T(1^n))^{1/k} \cdot |1^n|^{-1} \cdot \mu(1^n) \leqslant \infty.$$

There is a constant c such that for all n:

$$(T(1^n)^{1/k} \cdot n^{-1} \cdot n^{-2} \leqslant c \qquad \text{i.e., } T(1^n) \leqslant (cn^3)^k. \qquad \text{Q.E.D.}$$

9. RANDOMIZING REDUCTIONS

The proof of the incompleteness theorem, Theorem 8.1, does not give any indication that flat RNP problems are easier on average. The incompleteness theorem seems to hint that Ptime and even APtime reductions are not sufficiently strong. It

is natural at this point to raise the question of polynomial-time (or average polynomial-time) Turing reductions [7]. However, the incompleteness theorem survives the transition from many-one to Turing reductions; we omit the proof. Levin found a way to deal with the phenomenon of flatness [17]. He proposed to use randomizing (coin-flipping) Ptime reductions (RPtime reductions). A flat problem RP time complete for RNP can be found in [22].

In this section, we give a possible formalization of a simple version of RPtime reductions and then prove RPtime completeness of a flat version of Randomized Halting Problem for RNP. For simplicity, we restrict attention to decision problems in the binary alphabet. The proof of Lemma 3.1(3) shows how strings in larger alphabets can be coded by binary strings in a manner that respects probabilities.

One may want to use more liberal reductions that are randomized, Turing, and APtime at the same time. Different aspects of coin-flipping may be liberalized as well. Coins may be biased, the number of coin flips need not necessarily be polynomially bounded, etc. Also, reductions may be allowed to be incorrect in rare cases. We prefer to use the simplest reductions sufficient for our purposes.

DEFINITION. A *dilator* is a Ptime computable function from binary strings to natural numbers. If p is a dilator then the *p-dilation* (D_p, μ_p) of a randomized decision problem (D, μ) is the following randomized decision problem:

Instance. A pair (x, y) of binary strings where $|y| = p(x)$.

Question. Is x a positive instance of D?

Probability. $\mu_p(x, y) = \mu(x) \cdot 2^{-|y|}$.

In the rest of this section, p, q, and r are dilators. Note that the definition of dilations of randomized decision problems defines also dilations of decisions problems and of probability functions. To simplify notation, the q-dilation of the p-dilation of a randomized decision problem (D, μ) is denoted (D_{pq}, μ_{pq}). The following lemma shows that the effect of such double dilation can be achieved by a single dilation.

LEMMA 9.1. *For all dilators p and q, there exist a dilator r and a function f which transforms* (D_r, μ_r) *to* (D_{pq}, μ_{pq}).

Proof. Construct a dilator r such that $r(x) \geqslant p(x) + q(x, y)$ for all y with $|y| = p(x)$. If $|w| = r(x)$, set $f(x, w) = ((x, u), v)$ where uv is the initial segment of w such that $|u| = p(x)$ and $|v| = q(x, u)$. We need to check only that f transforms μ_r to μ_{pq}. Let x, u, and v be binary strings such that $|u| = p(x)$ and $|v| = q(x, u)$, and let $k = r(x) - (p(x) + q(x, u))$. Every preimage of $((x, u), v)$ with respect to f has the form (x, uvy), where $|y| = k$. Obviously,

$$\sum_{|y|=k} \mu_r(x, uvy) = \sum_{|y|=k} \mu_{pq}(x, uv) \cdot 2^{-k} = \mu_{pq}((x, u), v). \qquad \text{Q.E.D.}$$

Until now, the analog of P in the average complexity theory was the class AP of randomized decision problems (D, μ) such that some deterministic Turing machine decides D within time polynomial on μ-average. The use of randomizing algorithms gives rise to a more liberal analog of P.

DEFINITION. RAP is the class of randomized decision problems (D, μ) such that some dilation of (D, μ) is AP.

RAP is a class of randomized decision problems (D, μ) such that some randomizing (coin-flipping) Turing machine decides D within time polynomial on μ-average. For simplicity, we require that the coin is unbiased, that the number of coin tosses is polynomially bounded, and that all computations on the same input generate the same number of coin tosses. Does RAP properly include AP? We do not know.

DEFINITION. A randomized decision problem (D, μ) RPtime reduces to a randomized decision problem (E, ν) if some dilation of (D, μ) Ptime reduces to (E, ν).

An RPtime reduction is a coin-flipping Ptime reduction.

LEMMA 9.2.

1. *If ν dominates μ then, for every p, ν_p dominates μ_p.*

2. *If a Ptime computable function f transforms (D, μ) to a restriction of (E, ν) then, for every p, some Ptime computable function g transforms some (D_q, μ_q) to a restriction of (E_p, ν_p).*

3. *If (D, μ) RPtime reduces to (E, ν) then, for every q, some dilation of (D, μ) Ptime reduces to (E_q, ν_q).*

4. *RPtime reducibility is transitive.*

5. *If a randomized decision problem RPtime reduces to an RAP problem then it also is RAP.*

Proof. (1) Let f witness that ν dominates μ, so that $\mu(x) = f(x) \cdot \nu(x)$. Then

$$\mu_p(x, y) = \mu(x) \cdot 2^{-|y|} = f(x) \cdot \nu(x) \cdot 2^{-|y|} = f(x) \cdot \nu_p(x, y).$$

(2) Without loss of generality, f transforms μ to ν itself. For, let ν' be the restriction of ν such that f transforms μ to ν'. Then ν'_p is a restriction of ν_p. It follows that, if some g transforms some (D_q, μ_q) to (E_p, ν'_p) or to a restriction of (E_p, ν'_p) then g transforms (D_q, μ_q) to a restriction of (E_p, ν_p).

Choose the desired q such that $q(x) \geqslant p(fx)$. For every instance (x, y) of D_q, let $g(x, y) = (fx, z)$ where z is the prefix of y of length $p(fx)$. Obviously, g reduces D_q to E_p. We prove that g transforms μ_q to ν_p. Let (u, z) be an instance of E_p. Every preimage of (u, z) with respect to g has the form (x, zv) where $f(x) = u$ and

$|v| = q(x) - p(u)$. Let x range over $f^{-1}(u)$, $k = q(x) - p(u)$, and v range over binary strings of length k.

$$\sum_{x,v} \mu_q(x, zv) = \sum_x \mu(x) \cdot 2^{-|z|} \cdot \sum_v 2^{-k} = \nu_p(u, z).$$

(3) Suppose some (D_p, μ_p) Ptime reduces to (E, ν). Then there exist a probability function β and a Ptime computable function f such that β dominates μ_p and f transforms (D_p, β) to a restriction of (E, ν). By (2), there exist a dilator r and a Ptime computable transformation of (D_{pr}, β_r) to a restriction of (E_q, ν_q). It remains to prove that some dilation of (D, μ) Ptime reduces to (D_{pr}, β_r).

By (1), β_r dominates μ_{pr}; hence (D_{pr}, μ_{pr}) Ptime reduces to (D_{pr}, β_r). It remains to prove that some dilation of (D, μ) Ptime reduces to (D_{pr}, μ_{pr}). Now use Lemma 9.1.

(4) Suppose some (A, α) RPtime reduces to (B, β) which RPtime reduces to (C, γ). Then some (B_q, β_q) Ptime reduces to (C, γ). By (3), some dilation (A_p, α_p) Ptime reduces to (B_q, β_q) and therefore to (C, γ).

(5) Suppose that (D, μ) RPtime reduces to (E, ν) and a q-dilation of (E, ν) is AP. By (3), some (D_p, μ_p) Ptime reduces to (E_q, ν_q) and therefore is AP. Q.E.D.

RPtime reductions allow us to have prettier versions of randomized halting problems. Fix any function $\pi(i)$ from natural numbers to natural numbers such that:

- π is Ptime computable,
- π is nondecreasing; i.e., $i \leqslant j$ implies $\pi(i) \leqslant \pi(j)$,
- the function $\pi^{-1}(n) = \min_i(\pi(i) \geqslant n)$ is polynomially bounded.

For an NTM M, let $\mathrm{RH}_\pi(M)$ be the following version of the randomized halting problem for M:

Instance. A binary string w of some length l.

Question. Is there a halting computation of M on with at most $\pi(l)$ steps?

Probability. The probability of an instance w is the default probability $l^{-2}2^{-l}$ of the binary string w.

THEOREM 9.1. *Every RNP problem (D, μ) RPtime reduces to $RH_\pi(M)$ for some NTM M.*

Proof. Let M be an NTM and g be a longevity guard for M. By Lemma 4.2, we may suppose that (D, μ) is the restriction of $\mathrm{RH}(M, g)$ to stable instances.

Construct a dilator p such that $\pi(|w| + p(w01^n)) \geqslant n$. It suffices to prove that (D_p, μ_p) Ptime reduces to $\mathrm{RH}_\pi(M)$. The desired reduction is

$$f(w01^n, y) = wy.$$

First we check that f takes positive instances to positive instances and negative instances to negative instances. Let w be a stable input for M, $n = g(w)$, and y be a binary string of length $p(w01^n)$. An instance $(w01^n, y)$ of D_p is positive if and only if there exist a halting computation of M on w of length $\leqslant n$. Since n is longevous for w and $\pi(|wy|) \geqslant n$, M has a halting computation on w of length $\leqslant n$ if and only if it has a halting computation on w of length $\leqslant \pi(|wy|)$. Since w is stable, M has a halting computation of length $\leqslant \pi(|wy|)$ on w if and only if it has a halting computation of length $\leqslant \pi(|wy|)$ on wy if and only if wy is a positive instance of $\mathrm{RH}_\pi(M)$.

Next we check the domination condition. Let $(w01^n, y)$ be an instance of D_p, $|w| = l$, $|y| = k$, and $m = k + l$. Then $\mu_p(w01^n, y) = n^{-3}2^{-l}2^{-k} = n^{-3}2^{-m}$, whereas the probability of $f(w01^n, y)$ is $m^{-2}2^{-m}$, and the domination condition is obvious. Q.E.D.

Using Theorem 9.1 instead of Theorem 4.1, one can construct prettier versions of Randomized Post Correspondence Problem and Randomized Tiling Problem that are complete for RNP with respect to RPtime reductions.

10. Sparse Problems

Definition. A probability function μ on strings in some alphabet is *sparse* if there is a polynomial bound $p(n)$ on the number of strings x of length n such that $\mu(x) > 0$. A randomized decision problem (D, μ) is a *sparse RNP* problem if μ is weakly dominated by some sparse probability function ν with a Ptime computable probability distribution ν^*.

In this section, the term "exponential" is used in a more narrow sense. A function T from some Σ^* to nonnegative reals is *exponentially bounded* or, for brevity, *exponential* if there is a constant c with $T(x) \leqslant c^{|x|}$. A function f from some Σ_1^* to some Σ_2^* is *EXPtime computable* if some exponential-time Turing machine computes f. The decision problem D for some language $L(D)$ is *DEXPtime* (resp. *NEXPtime*) if some exponential-time Turing machine (resp. nondeterministic Turing machine) decides D.

Definition [Lew, 18]. A decision problem D EXP*time reduces* to a decision problem E if there exist an EXPtime computable function f and a constant c such that f reduces D to E and $|fx| \leqslant c \cdot |x|$.

The bound on $|fx|$ ensures the following desired feature of EXPtime reductions: If a NEXPtime D EXPtime reduces to a DEXPtime E then D is DEXPtime.

Definition. Let D be a NEXPtime decision problem and c be an integer such that some *NTM* accepts $L(D)$ in time $B(n) = c^n$. The *companion* of D with respect to B is the randomized decision problem (E, μ) such that $L(E) =$

$\{w01^{B(|w|)} : w \in L(D)\}$ and $\mu(w01^{B(|w|)})$ is the standard probability of string w (so that $\mu(y) = 0$ if y does not have the form $w01^{B(|x|)}$).

Any companion is a sparse RNP problem.

LEMMA 10.1. *Let (E, μ) be the companion of a NEXPtime decision problem D with respect to a bound $B(n)$. There is a polynomial p such that $p(|y|)\,\mu(y) \geq 1$ for all strings y of the form $w01^{B(|w|)}$.*

Proof. Clear.

LEMMA 10.2. *If a NEXPtime decision problem D_1 EXPtime reduces to a NEXPtime decision problem D_2 then any companion of D_1 Ptime reduces to any companion of D_2.*

Proof. Let f and c witness the EXPtime reducibility, and (E_i, μ_i) be an RNP companion of D_i. Let x be an instance of D_1, x' be the corresponding instance of E_1, and $y = f(x)$ and y' be the corresponding instance of E_2. The function $F(x') = y'$ reduces E_1 to E_2. It is computable in time exponential in $|x| + |y|$. Since $|y| \leq c|x|$, $F(x')$ is computable in time exponential in $|x|$, hence in time polynomial in $|x'|$. To prove that μ_2 dominates μ_1 with respect to F, use Lemma 10.1.

Q.E.D.

THEOREM 10.1. *Let E be any decision problem EXPtime complete for NEXPtime. Any companion E_0 of E is APtime complete for the class of sparse RNP problems.*

Proof. For every NTM M with binary input alphabet, define the *exponential halting problem* EH(M) for M as follows:

Instance. A binary string w.

Question. Is there a halting computation of M on w with at most $2^{|w|}$ steps?

Let (D, μ_0) be an arbitrary sparse RNP problem. We need to prove that it reduces in APtime to E_0. It suffices to prove that there is some NTM M_1 with binary input alphabet such that (D, μ_0) APtime reduces to a companion of EH(M_1). For, by Lemma 10.2, this companion of EH(M_1) APtime reduces to E_0, and therefore (D, μ_0) APtime reduces to E_0.

By Lemma 7.4(1), we may suppose that μ_0^* is Ptime computable. By the proof of Lemma 3.1(3), we may suppose that instances of D are binary strings. By Lemma 1.6, there is a positive probability function μ such that μ^* is Ptime computable and every value of μ is a binary fraction and $\mu_0(x) = O(\mu(x))$. Given (D, μ), construct x'' and M as in the proof of Theorem 4.1. M has a longevity guard g, the function $f(x) = x''01^{g(x'')}$ reduces (D, μ) to RH(M), $\mu(x)\,2^{-|x''|} < 2$, and $g(x'')$ is bounded by a polynomial of $|x|$.

Given an input $u01^i$, the desired machine M_1 simulates M on u. Let $F(x) = x''01^i01^j$, where $i = \lceil \log_2 g(x) \rceil$ and $j = 2^{|x''|+1+i}$. We show that F reduces

(D, μ_0) to the $B(n)$-companion (D_1, μ_1) of $\mathrm{EH}(M_1)$, where $B(n) = 2^n$. $F(x)$ belongs to $L(D_1)$ if and only if $x''01^i$ belongs to $\mathrm{EH}(M_1)$ if and only if there is a halting computation of M_1 on $x''01^i$ of length at most j if and only if there is a halting computation of M on x'' of length $\leqslant j$ if and only if there is a halting computation of M on x'' of length $\leqslant g(x)$ if and only if x belongs to $L(D)$. Thus, F reduces D to D_1.

To show that F is APtime computable, it suffices to check that the function $g(x) \cdot 2^{|x''|}$ is polynomial on average with respect to μ_0. In the discussion on polynomiality on average in Section 1, we formulated condition (iv) sufficient for polynomiality on average. Thus if suffices to prove that, for some k,

$$\sum_{\mu_0(x)>0} \mu_0(x) \cdot g(x) \cdot 2^{|x''|} \cdot |x|^{-k} < \infty.$$

We have $\mu_0(x)\, 2^{|x''|} = O(\mu(x)\, 2^{|x''|}) = O(1)$. Hence it suffices to prove that

$$\sum_{\mu_0(x)>0} g(x) \cdot |x|^{-k} < \infty,$$

which is true for a sufficiently large k depending on g and a polynomial witnessing the sparsity of μ_0.

Finally, use Lemma 10.1 to check that μ_1 dominates μ_0 with respect to F. Q.E.D.

Some NEXPtime complete problems can be found in [15, 18].

Appendix: Perfect Rounding and Randomized Tilling

This is a recast of report [10] with a reconstruction of Levin's completeness proof [16] for Randomized Tiling. When an undergrad David McCauley asked me for a challenge, he was invited to share the hard work of deciphering the exceedingly terse paper of Levin. David worked mostly on perfect rounding which is in the heart of Levin's original completeness proof for (a version of) Randomized Halting. Even though the new completeness proof in Section 4 above is short and straightforward, we find the ideas of the original proof very interesting and potentially useful; a reconstruction of the original proof is presented in Section A below. In Section B, which is independent from Section A, Randomized Halting is reduced to Randomized Tilling.

A. Perfect Rounding and Randomized Halting

Each binary fraction r in the half-open real interval $[0, 1)$ has a representation of the form $0.x$ where x is a binary string. If x has no trailing zeroes then the representation is called *standard* and $|x|$ is called the *length* $\mathrm{lh}(r)$ of r. If $0 \leqslant a < b < 1$ and I is an interval $[a, b]$, $[a, b)$, $(a, b]$, or (a, b), let Shortest (I) be any binary fraction of the minimal length in I.

LEMMA A1. *Shortest* (I) *is unique, and there are four algorithms, one for each of the four kinds of intervals, that construct Shortest* (I) *from the standard representations* $0.x$ *and* $0.y$ *for* a *and* b.

Proof. If $c < d$ are two binary fractions of the same length k then $c + 2^{-k}$ is a shorter binary fraction in (c, d). In the case of $(a, b]$, the desired algorithm works as follows:

1. If x is a perfix of y, find the longest string u of zeroes such that xu is a prefix of y and set $z = xu1$.

2. If x is not a prefix of y, find the greatest common prefix u of x and y and set $z = u1$.

Other cases are similar. Q.E.D.

Let μ, ν be probability functions over binary strings, and let M, N be the corresponding probability distributions μ^* and ν^*. Call μ *normalized* if $\mu(e) = \frac{1}{2}$, every $\mu(x)$ is a positive binary fraction of length at most $5 + 2|x|$, and M is Ptime computable. The bound $5 + 2|x|$ is somewhat accidental.

LEMMA A2. *Every* μ *with a Ptime computable* M *is dominated by a normalized* ν.

Proof. By virtue of Lemma 1.6 and its proof, we may suppose that every $\mu(x)$ is a positive binary fraction of length at most $4 + 2|x|$, and M is Ptime computable. Set $N(x) = \frac{1}{2} + M(x)/2$. Q.E.D.

Recall that the successor of a string x in the lexicographical order is denoted x^+. The predecessor of a string $x \neq e$ is denoted x^-.

Call μ *semirounded* if it is normalized and, for every $x > e$, Shortest$[M(x), M(x+)]$ is either $M(x)$ or $M(x^+)$.

LEMMA A3. *Every normalized* μ *is dominated by some semirounded* ν.

Proof. For every nonempty x, let Nx be the shortest binary fraction in the half-open interval $((Mx^- + Mx)/2, (Mx + Mx^+)/2]$. It is clear that the corresponding ν is a normalized probability function. We check that ν is semirounded. Let r be a binary fraction in the open interval (Nx, Nx^+). If $r > (Mx + Mx^+)/2$ then $\mathrm{lh}(r) > \mathrm{lh}(Nx^+)$ by the choice of Nx^+. If $r \leqslant (Mx + Mx^+)/2$ then $\mathrm{lh}(r) > \mathrm{lh}(Nx)$ by the choice of Nx.

Finally, we prove that $4\nu(x) > \mu(x)$ for all x, and therefore ν dominates μ. This is clear if $Nx \leqslant Mx$ or $Nx^+ \geqslant Mx^+$. Suppose $Mx < Nx < Nx^+ < Mx^+$. Let $k = \mathrm{lh}(Nx)$ and $l = \mathrm{lh}(Nx^+)$.

Case 1. $k \geqslant l$. $\mu(x) = Mx^+ - Mx = 2[(Mx + Mx^+)/2 - Mx] < 2(Nx^+ - Mx)$. Further, $Nx^+ - Mx = Nx^+ - Nx + Nx - Mx = \nu(x) + (Nx - Mx)$. Finally,

$Nx - Mx < 2^{-k}$ because Nx is the shortest binary fraction in $[Mx, Nx]$, and $2^{-k} \leqslant \nu(x)$ because $\nu(x)$ is the diffeence of two distinct binary fractions of length at most k.

Case 2. $l \geqslant k.\mu(x) = Mx^+ - Mx = 2[Mx^+ - (Mx^+ Mx^+)/2] \leqslant 2(Mx^+ - Nx)$. Further, $Mx^+ - Nx = Mx^+ - Nx^+ + Nx^+ - Nx = (Mx^+ - Nx^+) + \nu(x)$. Finally, $Mx^+ - Nx^+ < 2^{-l}$ because Nx^+ is the shortest binary fraction in $[Nx^+, Mx^+]$, and $2^{-l} \leqslant \nu(x)$ because $\nu(x)$ is the difference of two distinct binary fractions of length at most l. Q.E.D.

Call μ *perfectly rounded* if it is normalized and Mx is the shortest binary fraction in the open interval (Mx^-, Mx^+) for all $x > e$.

LEMMA A4. *Every semirounded probability function μ is dominated by a perfectly rounded one.*

Proof. The proof of Lemma A4 splits into several claims. Define:

$$[S(M)](x) = [\text{if } x > e \text{ then Shortest}(Mx^-, Mx^+), \text{ else } 0],$$

$S(M, x)$ abbreviates $[S(M)](x)$,

$$[S(\mu)](x) = S(M, x^+) - S(M, x),$$

$S(\mu, x)$ abbreviates $[S(\mu)](x)$.

Claim 1. $S(\mu)$ is a positive probability function.

Proof. We must check only that $F = S(M)$ is strictly increasing. By contradiction suppose that $Fx^+ \leqslant Fx$ for some x. Then $Mx < Fx^+ \leqslant Fx < Mx^+$. Taking into account the choice of Fx and the fact that $Mx < Fx$, we have $\text{lh}(Fx) < \text{lh}(Mx)$. Similarly, $\text{lh}(Fx^+) < \text{lh}(Mx^+)$. Thus neither Mx nor Mx^+ is the shortest binary fraction in $[Mx, Mx^+]$ which contradicts the semiroundedness of μ. Q.E.D.

Claim 2. $S(\mu)$ is semirounded.

Proof. Let $\nu = S(\mu)$. We check that Shortest $[Nx, Nx^+]$ is either Nx or Nx^+.

Case $Nx \leqslant Mx$ and $Mx^+ \leqslant Nx^+$. Let r be an arbitrary binary fraction in $[Nx, Nx^+]$. If r is in (Mx, Mx^+) then $\text{lh}(r) \geqslant \text{lh}(Mx) \geqslant \text{lh}(Nx)$ or $\text{lh}(r) \geqslant \text{lh}(Mx^+) \geqslant \text{lh}(Nx^+)$. If r is in $[Nx, Mx]$ then $\text{lh}(r) \geqslant \text{lh}(Nx)$ by the choice of Nx; if r is in $[Mx^+, Nx^+]$ then $\text{lh}(r) \geqslant \text{lh}(Nx^+)$ by the choice of Nx^+.

Case $Nx > Mx$. The interval $[Nx, Nx^+]$ is a part of (Mx, Mx^{++}); hence $Nx^+ = \text{Shortest}[Nx, Nx^+]$.

Case $Nx^+ < Mx^+$. The interval $[Nx, Nx^+]$ is a part of (Mx^-, Mx^+); hence $Nx = \text{Shortest}[Nx, Nx^+]$. Q.E.D.

Claim 2 justifies the repetitive use of the operator S (the *shaking operator*). The obvious abbreviations $S^k(M, x)$ and $S^k(\mu, x)$ are used.

Claim 3. Let r be a binary fraction and $k = \mathrm{lh}(r)$.

1. If $S^k(M, x) = r$ then $S^l(M, x) = r$ for all $l > k$.
2. If $a = S^k(M, x) < r < S^k(M, x^+) = b$ then both a and b are shorter than r.

Proof. By induction on k. If $k = 1$ then $r = \frac{1}{2}$ and the claim is obvious. Suppose that $k > 1$. First assume that $r = S^k(M, x)$. By contradiction suppose that $S^l(M, x) = q \neq r$ for some q and some $l > k$. Then $\mathrm{lh}(q)$ equals some $j < k$. By the induction hypothesis, $S^k(M, x) = S^j(M, x) = q$ which is impossible.

Next assume that $a = S^k(M, x) < r < S^k(M, x^+) = b$. Since $S^k(M)$ is semirounded, either a or b is shorter than r. Without loss of generality, a is shorter than r. By the induction hypothesis, $S^{k-1}(M, x) = a$. Since $S^k(M, x^+)$ is the shortest binary fraction in the interval $(S^{k-1}(M, x), S^{k-1}(M, x^{++}))$, b is the shortest binary fraction in $(a, b]$ and therefore b is shorter than r. Q.E.D.

Claim 4. Let x be a string of length l, $m = 5 + 2l$, and $n \geqslant m$. Then $S^m(M, x) = S^n(M, x)$.

Proof. Let $r = S^m(M, x)$. Since $S^m(M)$ is normalized, $\mathrm{lh}(r) \leqslant m$. Now use Claim 3. Q.E.D.

Claim 4 justifies introducing an iterated shaking operator S^∞:

$$[S^\infty(M)](x) = S^\infty(M, x) = S^{5+2|x|}(M, x),$$

$$[S^\infty(\mu)](x) = S^\infty(\mu, x) = S^\infty(M, x^+) - S^\infty(M, x).$$

Claim 5. $S^\infty(\mu)$ is a positive probability function.

Proof. First we check that $F = S^\infty(M)$ is strictly increasing. Let $x < y$ and $m = \max(5 + 2\mathrm{lh}(x), 5 + 2\mathrm{lh}(y))$. Then $F(x) = S^m(M, x) < S^m(M, y) = F(y)$. Next we check that for every positive real δ there is a binary string x with $Fx > 1 - \delta$. Pick any binary fraction $r > 1 - \delta$ and let $k = \mathrm{lh}(r)$. Since $S^k(M)$ is a probability function, there is x such that $S^k(M, x) = r$ or $a = S^k(M, x) < r < S^k(M, x^+) = b$ for some a, b. Now use Claim 3. In the first case, $S^\infty(M, x) = r$, and in the second case, $S^\infty(M, x^+) = b > r$. Q.E.D.

Claim 6. $S^\infty(M)$ is Ptime computable.

Proof. The idea of the proof is simple: the shaking operator works locally and every $S^m(M, x)$ is computable from an appropriate array of M-values for binary strings close to x in the canonic ordering of binary strings. To implement the idea, we need a couple of definitions.

Recall that binary strings are numbered by natural numbers with respect to the lexicographical order of strings; in particular, the empty string e is the string of number 0. If x is the nth string and m is an arbitrary integer, let $x + +m$ be the string of number $n + m$ if $n + m > 0$ and the empty string otherwise. Further, let $x - -m = x + +(-m)$.

A sequence $A = [A_1, ..., A_k]$ of binary fractions is called an *array* if A is a strictly increasing sequence possible augmented with a prefix of zeroes. In other words, A is an array if and only if, for every $i < k$, either $A_i = 0$ or $A_{i+1} > A_i$. If F is a strictly increasing probability distribution, x is a binary string and $m < n$ then the sequence $[F(x++m), ..., F(x++n)]$ is an array.

An array $A = [A_1, ..., A_k]$ is *semirounded* if for every $i < k$, either A_i or A_{i+1} is the shortest binary fraction in the closed interval $[A_i, A_{i+1}]$. If $A = [A_1, ..., A_k]$ is a semirounded array and $k \geqslant 3$, let $S(A)$ be the array $B = [B2, ..., B_{k-1}]$ such that each $B_i = [\text{if } A_i > 0 \text{ then Shortest}(A_{i-1}, A_{i+1}), \text{ else } 0]$; think about $S(A)$ as the result of shaking A. The proof of Claim 2 can be easily adapted to show that $S(A)$ is semirounded if A is.

It is easy to see that if F is a semirounded probability distribution and $A = [F(x--(l-1)), ..., F(x++(l+1))]$ for some x and some $l > 0$ then $S(A) = [S(F, x--l), ..., S(F, x++l)]$. Hence the one-element array $[S^l(M, x)]$ is the result of shaking the array $[M(x--l), ..., M(x++l)]$ l times. This gives a Ptime algorithm for computing $S^l(M, x)$. Now set $l = 5 + 2|x|$. Q.E.D.

Claim 7. $S^\infty(\mu)$ is perfectly rounded.

Proof. Let $\nu = S^\infty(\mu)$. Given x, let $m = 5 + 2|x^+|$. By the definition, $S^{m+1}(M, x)$ is the shortest binary fraction between $S^m(M, x^-)$ and $S^m(M, x^+)$. But $S^{m+1}(M, x) = Nx$, $S^m(M, x^-) = Nx^-$ and $S^m(M, x^+) = Nx^+$. Thus, Nx is the shortest binary fraction between Nx^- and Nx^+. Q.E.D.

Claim 8. Let $\nu = S(\mu)$ and x be a binary string with $\nu(x) < \mu(x)$. Then $\nu(x) = 2^{-m}$ where $m = \max(\text{lh}(Nx), \text{lh}(Nx^+))$, and $2\nu(x) > \mu(x)$.

Proof. Since M is semirounded, either Mx or Mx^+ is the unique shortest binary fraction in $[Mx, Mx^+]$. Let $k = \text{lh}(Mx)$ and $l = \text{lh}(Mx^+)$. By virtue of symmetry, we may suppose that $k < l$. Then $\mu(x) = 2^l$ and $Nx \leqslant Mx$. Moreover, $Nx = Mx$; for, if $Nx < Mx$ then $\nu(x) > Mx - Nx \geqslant 2^{-k} \geqslant \mu(x)$. Since $\nu(x) < \mu(x)$, $Nx^+ < Mx^+$. Let $m = \text{lh}(Nx^+)$. Then $Mx^+ - Nx^+ < 2^{-m}$ and $Nx^+ - Mx = 2^{-m}$. Hence $\mu(x) = Mx^+ - Mx < 2 \cdot 2^{-m} \leqslant 2\nu(x)$. Q.E.D.

Claim 9. $S^\infty(\mu)$ dominates μ.

Proof. Let $\nu = S^\infty(\mu)$. Given a binary string x, we prove that $2\nu(x) \geqslant \mu(x)$. Let $\mu_i x = S^i(M, x^+) - S^i(M, x)$. If every $\mu_i(x) \geqslant \mu_{i-1}(x)$, then there is nothing to prove. Let i be any positive integer with $\mu_i(x) < \mu_{i-1}(x)$. By Claim 8, $\mu_i(x) = 2^{-m} > 2\mu_{i-1}(x)$ where $m = \max(\text{lh}(S^i(M, x^+)), \text{lh}(S^i(M, x)))$. Thus, $\mu_i(x)$ is minimal when i is the minimal positive integer with $\mu_i x < \mu_{i-1}(x)$. For the minimal i, $\mu_i x > 2\mu_{i-1}(x) \geqslant \mu(x)$. Q.E.D.

Thus, $S^\infty(\mu)$ is the desired perfectly rounded probability function, and Lemma A4 is proved. Q.E.D.

THEOREM A1. *Every RNP problem reduces to an RNP problem (D, μ) over binary strings such that μ is perfectly rounded.*

Proof. Use Lemmas A2, A3, and A4. Q.E.D.

LEMMA A5. *If μ is a perfectly rounded probability function then for every binary string x there exists a positive integer $j \leqslant 5 + 2|x|$ such that $\mu(x) = 2^{-j}$ and $2^j M(x)$ is integer.*

Proof. Let $a = M(x)$ and $b = M(x^+)$, so that $\mu(x) = b - a$. Since $a = \text{Shortest}[a, b)$, every binary fraction in (a, b) is longer than a. Since $b = \text{Shortest}(a, b]$, every binary fraction in (a, b) is longer than b. If a is shorter than b, then the desired j equals $\text{lh}(b)$; otherwise $j = \text{lh}(a)$. Since μ is normalized, $j \leqslant 5 + 2|x|$. Q.E.D.

THEOREM A2. *Every RNP problem reduces to the randomized halting problem $RH(A)$ for some NTM A.*

Proof. It suffices to prove that every RNP problem (D, μ) over binary strings such that μ is perfectly rounded reduces to some $\text{RH}(A)$. Define $m(x) = M(x)/\mu(x)$. By Lemma A.5, $m(x)$ is a nonnegative integer. Since $M(e) = 0$ and $\frac{1}{2} \leqslant M(x) < 1$ for $x \neq e$, $\mu(x)$ is easily computable from $m(x)$: If $m(x) = 0$ then $\mu(x) = \frac{1}{2}$, and if $m(x) > 0$ then $\mu(x)$ is the unique positive integer j with $\frac{1}{2} \leqslant m(x)\, 2^{-j} < 1$.

Since D is NP, there is a Ptime-bounded nondeterministic Turing machine, called the D-machine below, that accepts $L(D)$. Given a binary string w representing some $m(x)$, the desired Turing machine A finds the appropriate x and then simulates the D-machine on x; if w does not represent any $m(x)$ then A does not halt on w. Specifically, A executes the following algorithm:

1. If $w = e$ or w starts with a 0 but is different from a 0, then loop. If $w = 0$ then set $x = e$ and g to (4).

2. Find the integer k represented by w, the integer j with $\frac{1}{2} \leqslant k2^{-j} < 1$, and the shortest string 1^l such that $M(1^{l+1}) \geqslant k2^{-j}$.

3. Use binary search to find the lexicographically maximal binary string x such that $1^l < x \leqslant 1^{l+1}$ and $M(x) \leqslant k2^{-j}$. If $M(x) < k2^{-j}$ then loop.

4. Simulate the D-machine on x; halt only if the D-machine accepts.

A curious thing is that A is not necessarily Ptime bounded because the representation of $m(x)$ may be much shorter than x. However, there is a polynomial p such that if A has a halting computation on the representation of $m(x)$ then it has one with at most $p(|x|)$ steps. The desired reduction is $f(x) = w01^n$ where w represents $m(x)$ and $n = p(|x|)$.

We check that the probability function ν of $\text{RH}(A)$ dominates μ with respect to f. Since f is one-to-one, it suffices to check that $\nu(fx)$ dominates $\mu(x)$. We may restrict attention to the case $m(x) > 0$. Let $l = |m(x)|$. Note that $2^l \geqslant m(x)$. Thus, $\nu(fx) = (6/\pi^2) \cdot p(|x|)^{-3} \cdot 2^{-l} \leqslant (6/\pi^2) \cdot p(|x|)^{-3}/m(x) \leqslant (6/\pi^2) \cdot p(|x|)^{-3} \cdot 2\mu(x)$.

Q.E.D.

B. *Randomized Tiling*

DEFINITION. An NTM A *survives* n steps on input w if there exists a computation of A on w with at least n steps.

Randomized Survival Problem RS(A) for a Given NTM A

Instance. A binary string $w01^n$ where $n > |w|$.

Question. Does A survive n steps on w?

Probability. Choose randomly a positive integer n, a natural number $k < n$, and a binary string w of length k.

Recall that a number n is longevous for an input w of an NTM A if every halting computation of A on n has $\leqslant n$ steps. In other words, n is longevous for w with respect to A if and only if every computation of A on w with $\geqslant n+1$ steps is nonhalting. If g is a longevity guard for A, let $\mathrm{RS}(A)|g$ be the restriction of $\mathrm{RH}(A)$ to instances $w01^{g(w)+1}$.

LEMMA B1. *For every NP problem D there exists an NTM B_D with a polynomially bounded longevity guard g such that B_D survives $g(w)+1$ steps on an input w if and only if w is a positive instance of D.*

Proof. Since D is NP, there exist an NTM A_D and a polynomially bounded function g such that A_D has a halting computation on an input w if and only if w is a positive instance of D. The desired B_D simulates A_D and keeps track of time; if A_D halts after at most $g(w)$ steps then B_D loops forever, otherwise B_D halts after $g(w)$ steps. Q.E.D.

LEMMA B2. *For every RNP problem (D, μ) there exist an NTM M and a longevity guard g for M such that (D, μ) Ptime reduces to RS(M, g).*

Proof. Similar to that of Theorem 4.1. Instead of A_D, use machine B_D of Lemma B.1. In the description of the algorithm of M, replace "loop forever" by "halt". Q.E.D.

We redefine the notion of stability introduced in Section 4. Instead of the stability for halting, we are interested here in the stability for survival.

DEFINITION. Let A be an NTM. An input w is *stable* for A if, for every natural number n, the following statements are equivalent:

1. There exists x such that A survives n steps on wx,
2. for every x, A survives n steps on wx.

The following lemma is the analog of Lemma 4.2.

LEMMA B3. *For every $RS(M_0, g_0)$, there exist an NTM M and a longevity guard g for M such that $RS(M_0, g_0)$ Ptime reduces to the restriction of RS(M, g) to stable*

instances (i.e. to instances $w01^{g(w)+1}$ *where w is stable for M). Moreover, it may be required that* 0 *and* 1 *are the only tape symbols of M (with* 0 *serving also as the blank).*

Proof. Similar to the proof of Theorem 4.2. Q.E.D.

THEOREM B1. *Randomized Tiling is Ptime complete for RNP.*

bf Proof. Let A be an NTM such that 0 and 1 are the only tape symbols of A (with 0 serving also as the blank). Let π be a longevity guard for A, and (D, μ) be the restriction of $\mathrm{RS}(A)|\pi$ to stable instances. By Lemmas B2 and B3, it suffices to reduce (D, μ) to Randomized Tiling.

Let T contain the following tiles (T1)–(T5).

(T1)

$$\begin{array}{ccc} & \$pa & \\ \$ & & 0 \\ & \$ & \end{array}$$

where p is the initial state of A and a is either 0 or 1.

(T2)

$$\begin{array}{ccc} & a & \\ 0 & & 0 \\ & \$ & \end{array}$$

where a is either 0 or 1.

(T3)

$$\begin{array}{ccc} \$c & & c \\ \$ \quad 1 & & b \quad b \\ \$c & & c \end{array}$$

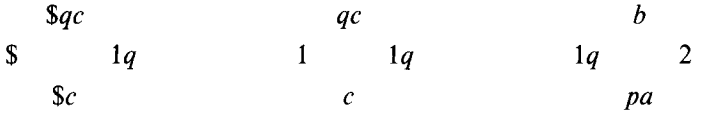
where b is 1 or 2, and c ranges over the tape symbols of A.

(T4) For each instruction $[pa \to qbR]$ of A,

$$\begin{array}{ccc} \$b & b & qc \\ \$ \quad 2q & 1 \quad 2q & 2q \quad 2 \\ \$pa & pa & c \end{array}$$

where c ranges over the tape symbols of A.

(T5) For each instruction $[pa \to qbL]$ of A,



where c ranges over the tape symbols of A.

LEMMA B4. *Suppose that τ is a T-tiling of $[0 \cdots (n-1)] \times [0 \cdots (n-1)]$ and $\tau(0, 0)$ is one of the two T1-tiles. Then:*

1. *Every $\tau(0, j+1)$ is in T2. Every $\tau(i+1, j)$ is in T3, T4, or T5.*

2. *The left string of $\tau(i, j)$ is \$ if and only if the top string of $\tau(i, j)$ contains a proper prefix \$ if and only if $j = 0$.*

3. *For every i there is at most one j such that the top string of $\tau(i, j)$ has a state symbol.*

4. *For every i there is $j \leqslant i$ such that the top string of $\tau(i, j)$ contains a state symbol.*

5. *For every i, let $\$w_i$ be the concatenation of the top strings of $\tau(i, 0), \ldots, \tau(0, n-1)$. Then each w_i is an ID (instantaneous description) of A.*

Proof. (1) Only T2-tiles have the string 0 on the left. Every tile in T1 or T2 has the string \$ on the bottom, but no T-tile has the string \$ on the top.

(2) The first equivalence is proved by direct inspection of T. The second is proved by induction on i. If $i = 0$, use (1). If $i > 0$, use (1) and the fact that for every tile in T3, T4, and T5, the bottom string has a proper prefix \$ if and only if the top string does.

(3) Induction on i. The case $i = 0$ is obvious. Assume $i > 0$. By contradiction, suppose that $j < k$ and the top strings of both (i, j) and (i, k) have state symbols. The right string of (i, j) is either $1q$ or 2. In either case, $\text{top}[\tau(i, j+1)]$ does not contain a state symbol and $\text{right}[\tau(i, j+1)] = 2$. But for every T-tile, if the left string is 2 then the right string is 2. Hence (i, k) has 2 on the left which is impossible.

(4) An obvious induction on i.

(5) Use (3) and (4). Q.E.D.

Given an instance $(\alpha_0 \cdots \alpha_{k-1}, 1^n)$ of RS(A), the desired reduction f produces an instance $\langle T, 1^n, k, \rho \rangle$ of Randomized Tiling where $\rho(0)$ is the T1-tile with α_0 on the top, and each $\rho(j+1)$ is the T2-tile with α_{j+1} on the top.

LEMMA B5. *A survives n steps on a stable input $u = \alpha_0 \cdots \alpha_{k-1}$ if and only if $\langle T, 1^n, k, \rho \rangle$ is a positive instance of Randomized Tiling.*

Proof. Let τ be a T-tiling of the square such that $\tau(0, j) = \rho(j)$ for $j < k$, and let

strings w_i be as in Lemma B4(5). Then w_0 is the initial configuration of A on some input ux. Check by induction on i that each w_i is the ith configuration of A on the input wx. Thus, A survives n steps on ux. Since u is stable, A survives n steps on u. The "only if" implication is easy. Q.E.D.

LEMMA B6. *The probability function ν of Randomized Tiling dominates the probability function μ with respect to f.*

Proof. It is easy to see that $\nu(f(u01^n))$ is proportional to $\mu(u01^n)$. (It is important that there are exactly two choices for each $\rho(j)$.) Q.E.D.

Theorem B1 is proved. Q.E.D.

ACKNOWLEDGMENTS

David McCauley put much work and ingenuity into the reconstruction of Levin's ideas on perfect rounding. Leonid Levin generously explained us his ideas. Numerous discussions with Andreas Blass and Saharon Shelah were very useful as well as enjoyable; the reader will note that some results are authored or coauthored by Andreas Blass or Saharon Shelah. Shai Ben-David and Michael Luby kindly allowed me to publish here a theorem of theirs. Short discussions with David Harel, Quentin Stout, and Martin Tompa were helpful. The material of this paper was taught at the University of Michigan in Winter 1987 and at Stanford in Spring 1989; I am grateful to the students and especially to Tomasz Radzik.

REFERENCES

1. S. BEN-DAVID, B. CHOR, O. GOLDREICH, AND M. LUBY, On the theory of average case complexity, *in* "Proc. 21st Annual ACM Symposium on Theory of Computing, ACM," pp. 204–216, 1989.
2. A. BLASS, Private communication.
3. B. BOLLOBAS, T. I. FENNER, AND A. M. FRIEZE, An algorithm for finding Hamilton cycles in a random graph, *in* "Proc. 17th Annual ACM Symposium on Theory of Computing, ACM," 1985, pp. 430–439.
4. S. BEN-DAVID AND M. LUBY, Private communication.
5. S. BEN-DAVID, B. CHOR, O. GOLDREICH, AND M. LUBY, On the theory of average case complexity, *in* "Proc. 21st Symp. on Theory of Computing, ACM," pp. 204–216, 1989.
6. R. L. CONSTABLE, H. B. HUNT, AND S. K. SAHNI, "On the Computational Complexity of Scheme Equivalences," Report No. 74–201, Dept. of Computer Science, Cornell University, Ithaca, NY, 1974.
7. M. R. GAREY AND D. S. JOHNSON, "Computers and Intractability," Freeman, New York, 1979.
8. Y. GUREVICH, Complete and incomplete randomized NP problems, *in* "Proc. 28th Annual Symp. on Found. of Computer Science, IEEE," pp. 111–117, 1987.
9. Y. GUREVICH, The challenger–Solver game: Variations on the theme of $P = ?NP$, *Bull. Eur. Assoc. Theoret. Comput. Sc.*, Oct., 1989.
10. Y. GUREVICH AND D. M. CAULEY, "Average Case Complete Problems," Unpublished manuscript, April 1987.
11. Y. GUREVICH AND S. SHELAH, Expected computation time for Hamiltonian path problem, *SIAM J. Comput.* **16**, No. 3 (1987), 486–502.
12. J. E. HOPCROFT AND J. D. ULLMAN, "Introduction to Automata Theory, Languages and Computation," Addison–Wesley, Reading, MA, 1979.

13. D. S. JOHNSON, The NP-completeness column, *J. Algorithms* **5** (1984), 284–299.
14. KER-I KO, On the definition of some complexity classes of real numbers, *Math. Systems Theory* **16** (1983), 95–109.
15. P. G. KOLAITIS AND M. Y. VARDI, The decision problem for the probabilities of higher-order properties, *in* "STOC 1987."
16. L. LEVIN, Average case complete problems, *SIAM J Comput.* **15** (1986), 285–286.
17. L. LEVIN, Private communication.
18. H. R. LEWIS, Complexity results for classes of quantificational formulas, *J. Comput. System Sci.* **21** (1980), 317–353.
19. N. MEGIDDO AND U. VISHKIN, "On Finding a Minimum Dominating Set in a Tournament," IBM Research Report RJ 5745, July 1987.
20. N. MEGIDDO, "Are the Vertex Cover and the Dominating Set Problems Equally Hard?" IBM Research Report RJ 5783, August 1987.
21. P. DINH DIEU, L. CONG THANH AND L. TUAN HOA, Average polynomial time complexity of some NP-complete problems, *Theoret. Comput. Sc.* **46** (1986), 219–237.
22. R. VENKATESAN AND L. LEVIN, Random instances of a graph coloring problem are hard, *in* "Proc. 20th Symp. on Theory of Computing, ACM, 1988."
23. H. S. WILF, Some examples of combinatorial averaging, *Amer Math. Monthly* **92** (1985), 250–261.
24. A. K. ZVONKIN AND L. LEVIN, The complexity of finite objects and the algorithmic concepts of information and randomness, *Russian Math. Surveys* **25/26** (1970), 83–124.
25. R. IMPAGLIAZZO AND L. LEVIN, No better ways to generate hard instances than picking uniformly at random, *FOCS* 1990, 812–821.
26. A. BLASS AND Y. GUREVICH, On the reduction theory for average case complexity, *in* "4th Workshop on Computer Science Logic" (E. Börger *et al.*, Eds.), Springer Lecture Notes in Computer Science, to appear.