INFORMATION AND COMPUTATION 104, 159–174 (1993)

On Dice and Coins: Models of Computation for Random Generation

DAVID FELDMAN

Department of Mathematics, University of New Hampshire, Durham, New Hampshire 03824

RUSSELL IMPAGLIAZZO[1]

Computer Science Department, University of Toronto, Toronto, Ontario, Canada M5S 1A4; and Department of Computer Science, University of California, San Diego, La Jolla, California 92093

MONI NAOR[1]

IBM Almaden Research Center, 650 Harry Road, San Jose, California 95120; and Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel

NOAM NISAN[2]

Department of Computer Science, Hebrew University, Jerusalem 91904, Israel

STEVEN RUDICH[1]

School of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213

AND

ADI SHAMIR

Department of Applied Mathematics, The Weizmann Institute of Science, Rehovot 76100, Israel

To examine the concept of random generation in *bounded*, as opposed to *expected*, polynomial time, a model of a probabilistic Turing machine (PTM) with the ability to make random choices with any (small) rational bias is necessary. This

[1] Supported by NSF Grant DCF-85-13926.
[2] Supported by a grant from Digital Equipment Corporation.

0890-5401/93 $5.00

ability is equivalent to that of being able to simulate rolling any k-sided die (where $|k|$ is polynomial in the length of the input). We would like to minimize the amount of hardware required for a machine with this capability. This leads to the problem of efficiently simulating a family of dice with a few different *types* of biased coins as possible. In the special case of simulating one n-sided die, we prove that only two types of biased coins are necessary, which can be reduced to one if we allow irrationally biased coins. This simulation is efficient, taking $O(\log n)$ coin flips. For the general case we get a tight time vs number of biases tradeoff; for example, with $O(\log n)$ different biases, we can simulate, for any $i < n$, an i-sided die in $O(\log n)$ coin flips.

1. INTRODUCTION

Probabilistic Turing machines (PTM), as defined in [Gil77], have access only to a coin which comes up heads or tails with equal probability, what we call a fair coin. There is no important difference between having a fair coin and having a biased coin when the PTM is used for language recognition. The best one can hope for in probabilistic language recognition is a short expected time and a coin which comes up heads with probability p, the bias of the coin, can indeed be simulated by a fair coin in short expected time. However, when the PTM is used to generate a combinatorial structure according to some distribution [Jer86], a natural question arises: can we do better than short (polynomial) expected time, i.e., can we have a random generator for some distribution which will always terminate after a short time. (An example of this type of random generation is found in the introduction to Section 3 of this paper; an algorithm which is inherently only expected time is Karp and Luby's random generation of satisfying instances for DNF formulae [Kar83].) There is a conceptual difference between these two kinds of random generation, in that (at least in the light of our results) a bounded-time algorithm can be thought of as a mapping from random strings to a set of combinatorial structures in a way that all structures in the set have pre-images of equal total weight.

If one has only a fair coin, then even the simple task of choosing with equal probability between 3 numbers, i.e., rolling a 3-sided die, has no algorithm that always terminates. Therefore additional hardware is required. We assume that for any bias p, a device is available that produces a head with a probability p. Our goal is to minimize the number of such devices (coins) required to generate samples according to certain distributions in bounded time. When a distribution is naturally associated with a certain device, such as an n-sided die, and we are generating samples from it by other means, we say we are simulating the device or its distribution.

We start in Section 2 with the simple problem of simulating an n-sided die for a fixed n. As we have mentioned, one cannot, in general, simulate an n-sided die having anly a fair coin. In Section 2.1 it is shown that if we have a coin of bias $1/n$ and a fair coin, then with at most $2 \log n$ coin flips we can simulate an n-sided die. Conversely, if we consider only coins whose bias is rational, then two coins are necessary, unless n is a power of two. This is shown in Section 2.2. If we do allow coins of irrational bias, then for any n, an n-sided die can be simulated with only one type of coin. This is shown in Section 2.3.

We then turn in Section 3 to the question of having a Turing machine which can at any step make a choice of any rational probability. Jerrum *et al.* [Jer86] argue against having such machines as models of computation since they violate the philosophy of Turing machines, because not every step can be implemented with a fixed amount of hardware and in a fixed time. We show, however, that by fixing in advance $\log n$ types of coin, any coin of rational bias whose denominator is less than n can be simulated in $O(\log n)$ coin flips. This is proven in Section 3.1. In Section 3.2 we give a tight lower bound, in that we show that any set of coins which enables us to simulate any coin of bias j/k, where $k < n$ with d coin flips, has size $\Omega(\log n/\log d)$. For each point on this curve we have a matching upper bound. Thus, if we have an algorithm which decides the bias with which to make a choice based on the computation so far, we can replace this with an algorithm which acts deterministically, but which has access to $\log n$ different size $T \log n$ random tapes over different alphabets. (T is a time bound for the first algorithm, n, the size of the maximum denominator of a bias used by the algorithm.) Unfortunately, unless the jth bit of $n!/i$ is easily computable, the simulation we have is non-uniform (see Section 3.1 for a discussion).

1.1. *Notation*

A *coin of bias p* is a device that, upon request, randomly outputs either *heads* or *tails* with a fixed probability p for heads. We assume that the outputs of such coins are independent.

Let D be a probability distribution on a finite set S. For a set of coins C to *simulate* D there must be a certain finite binary decision tree t. The leaves of t must be labeled with elements of S. Other nodes of t must be labeled with coins in C. The depending branches labeled one *heads* and one *tails* and D must be the distribution on S obtained from t by generating samples as follows. Start at the root. At internal nodes, flip the associated coin and proceed along the branch indicated until arriving at a leaf labeled by S. If the depth of t is d then it is said to use *d coin flips.*

We can restate our goal as follows. Given distributions $D_1, D_2, \ldots, D_n$

and a depth d, find the minimum size of a set of coins C that can simulate each D_i in d or fewer coin flips.

A *fair n-sided die*, for a natural number n, is the distribution which assigns equal probability for each integer in $1, ..., n$. A (fair) coin is a (fair) 2-sided die.

A sequence of coin flips can be viewed as a binary number by assigning to heads the value 1 and to tails the value 0.

Throughout the paper we use $a \bmod b$ to refer to the remainder in the division of a by b.

2. Simulating a Single Die

In this section we discuss the problem of simulating an n-sided die with as few coins as possible. We first restrict ourselves to rational coins, and give a simple procedure which, for any n, requires only two types of rationally biased coins to simulate an n-sided die by flipping the coins at most $\lceil 2 \log n \rceil + 1$ times. In Section 2.2 we show that this is the best possible; i.e., that except for powers of 2 you cannot simulate any n-sided die with just one type of rational coin. Then in Section 2.3 we show that for any n we can simulate an n-sided die by using a single coin whose bias is an algebraic number.

2.1. *Simulating with Rational Coins*

We give a simple procedure to simulate a fair n-sided die with two rationally biased coins using at most $\lceil 2 \log n \rceil + 1$ coin flips. The two coins used are a fair coin and one of bias $1/n$. Any such procedure determines a map from the set of sequences of coin flips into the numbers from 1 to n. Thus, we can think of our algorithm as a way to divide the possible sequences of coin flips into n bins, one for each number between 1 and n. The first $n-1$ bins have a similar construction, and the last bin, the "remainder" bin, is used to balance the probabilities. We use the same scheme in Section 2.3 to show how a single coin with an algebraic number can be used to simulate any die.

Consider sequences consisting of a single flip of the $1/n$ coin and k flips of the fair coin, where $k = \lceil 2 \log n \rceil$. There will be two types of sequences, one where the $1/n$ coin is tails and the other where the $1/n$ coin is heads. We have 2^k sequences of the first type, each with probability $(n-1)/(n2^k)$, and 2^k sequences of the second type having probability $1/(n2^k)$ each.

We plan to make the sum of the probabilities of the sequences assigned to each bin equal $1/n$. Let d be $\lfloor 2^k/(n-1) \rfloor$ and r be $2^k \bmod (n-1)$, i.e., d and r are are integers with $0 \leqslant r < n-1$ and $d(n-1)+r = 2^k$. We assign to

each of the first $n-1$ bins d sequences of the first type and r of the second type. The last bin will contain the remaining r sequences of the first type, and $2^k - r(n-1)$ sequences of the second type. This imposes a restriction on k, namely $r(n-1)$ must be no greater than 2^k. To satisfy this restriction it is enough for k to be at least $\lceil \log(n^2) \rceil = \lceil 2 \log n \rceil$.

The sum of the probabilities of the sequences in each of the first $n-1$ bins is

$$d \cdot \frac{n-1}{n2^k} + r \cdot \frac{1}{n2^k} = 1/n$$

The sum of the probabilities of the sequences in the last bin is

$$r \cdot \frac{n-1}{n2^k} + (2^k - r(n-1)) \cdot \frac{1}{n2^k} = 1/n.$$

Hence we have

THEOREM 1. *For any n, a fair n-sided die can be simulated using 2 types of biased coins and at most $\lceil 2 \log n \rceil + 1$ coin flips.*

The algorithm we get is:

(1) Flip the $1/n$ coin once and the fair coin $k = \lceil 2 \log n \rceil$ times. Let v be the number that represents the value of the sequence of the k fair coin flips when viewed as a binary number.

(2) If the $1/n$ coin is tails then if $v \leqslant d \cdot (n-1)$ then output $(v \bmod (n-1)) + 1$ else output n.

(3) If the $1/n$ coin is heads then if $v \leqslant r(n-1)$ output $(v \bmod (n-1)) + 1$ else output n.

2.2. *One Rational Coin Does Not Suffice*

THEOREM 2. *If n is not a power of 2, it is impossible to simulate a fair n-sided die using only one rationally biased coin.*

Proof. We prove the theorem for n a prime > 2; the general result follows immediately from this, since if we could simulate an n-sided die with a single coin, we could simulate a p-sided die for any divisor p of n with the same coin by taking the result mod p. Thus, if our claim is true, 2 is the only possible prime divisor of n.

Assume we could simulate with d flips an n-sided die with one coin of bias p/q, p and q relatively prime integers. This means that we can partition the set of all sequences of "heads" and "tails" of length d into n subsets,

$S_1, S_2, \ldots, S_n$, each of probability $1/n$, according to the output our algorithm gives after flipping such a sequence. For each S_j, define c_{ij} as the number of sequences in S_j with exactly i "heads." Then we have

$$\sum_{i=0}^{d} c_{ij} \cdot \left(\frac{p}{q}\right)^i \cdot \left(1 - \frac{p}{q}\right)^{d-i} = \frac{1}{n}$$

for each subset S_j.

From this, it immediately follows that n divides q, since n is prime and $1/n$ can be written as a sum of fractions all of whose denominators are powers of q. Let $q = rn$. Since $n > 2$ and each sequence appears in at most one S_j, there is an S_j with neither the sequence of all heads nor that of all tails as an element, i.e., $c_{0j} = c_{dj} = 0$. Then we can rewrite the last equation as

$$\frac{1}{(r \cdot n)^d} \cdot \sum_{i=1}^{d-1} c_{ij} \cdot p^i \cdot (rn - p)^{d-i} = \frac{1}{n},$$

which implies that

$$\sum_{i=1}^{d-1} c_{ij} \cdot p^i \cdot (rn - p)^{d-i} = r^d \cdot n^{d-1}.$$

Taking this mod p we have

$$0 = r^d \cdot n^{d-1} \bmod p.$$

but p is relatively prime to $q = rn$, and hence relatively prime to r and to n. So r and n are invertible mod p and we get $0 = 1 \bmod p$. This implies that $p = 1$. Similarly, $rn - p = 1$, so $r \cdot n = 2$, contradicting $n > 2$.

2.3. *One Algebraic Coin Suffices*

We now turn to the question of simulating an n-sided die when we allow the bias to be irrational, and specifically algebraic. We show that for any n, there exists a bias which is a solution to an integer polynomial equation of degree $2 \log n$, so that with a coin of that bias we can simulate an n-sided die by flipping it $O(\log n)$ times.

Suppose we have a coin with bias b, and we flip the coin k times. We try to divide all sequences, as in Section 2.1, between n bins, the first $n-1$ bins having a similar construction and the last one being a remainder bin. For each $0 \leqslant i \leqslant k$ we will try to divide all sequences with i heads evenly between the bins. All sequences with i heads have the same probability. So, for any $0 \leqslant i \leqslant k$, we put in each of the first $n-1$ bins $\lfloor \binom{k}{i}/(n-1) \rfloor$

sequences and in the last bin $\binom{k}{i} \bmod (n-1)$ sequences. Let $L(b)$ be the sum of the probabilities of the sequences assigned to the last bin as a function of b:

$$L(b) = \sum_{i=0}^{k} c_i \cdot b^i \cdot (1-b)^{k-i} \qquad \text{where} \quad c_i = \binom{k}{i} \bmod (n-1).$$

If, for some b, $L(b) = 1/n$ then the sum of the probabilities of the sequences assigned to each of the first $n-1$ bins is $1/n$ as well, since all other bins were constructed symmetrically. We show the existence of such a b using continuity. Clearly $L(0) = 1$,

$$L\left(\frac{1}{2}\right) = \sum_{i=0}^{k} c_i \frac{1}{2^i} \cdot \left(1 - \frac{1}{2}\right)^{k-i} = \frac{1}{2^k} \cdot \sum_{1=0}^{k} c_i$$

If k is large enough, i.e., $k \geqslant 3 \log n$, we have $L(\frac{1}{2}) < 1/n$. Since L is a continuous function, there is $0 \leqslant b_0 \leqslant \frac{1}{2}$ such that $L(b_0) = 1/n$. That b_0 is exactly the bias we need in order to simulate an n-sided die. Therefore we have

THEOREM 3. *For every n, a fair n-sided die can be simulated using $O(\log n)$ flips of a single coin whose bias is an algebraic number.*

The algorithm we get is:

(1) flip the coin of bias $b_0 k = 3 \log n$ times.

Let S be the sequence we got, and assume S has i tails. Let $r(S)$ denote the rank of S in a lexicographical order of all sequences of length k with i tails. $r(S)$ can be easily computed in time linear in k.

(2) If $r(S) > \binom{k}{i} - c_i$ then, output n.

Otherwise, output $(r(S) \bmod (n-1)) + 1$.

3. *Simultaneous Simulation of Many Coins*

Our goal in this section is to design machines that can simulate all rational coins. Having such machines is necessary for the random generation in bounded time of more complicated combinatorial structures. For instance, consider the problem: given a graph, generate a spanning tree of the graph in such a manner that all spanning trees of the graph are output with the same probability.

Since an efficient procedure for counting the number of spanning trees in

a graph exists [Eve79], and since the problem is self-reducible [Jer86] we have the following algorithm:

Let $G(V, E)$ be the graph, and let T be the spanning tree we are building. $ST(G)$ will denote the number of spanning trees G has. For an edge $e \in E$, let G_e denote the graph we get from G by contracting e, that is, removing the edge e and identifying its endpoints.

(1) If G is a graph composed of one node halt. T is a spanning tree.

(2) Select an arbitrary edge $e \in E$.

(3) With probability $ST(G_e)/ST(G)$ execute step a; otherwise, execute step b.

(a) Add e to T and let G be G_e.

(b) Remove e from the graph G. Thus, E becomes $E - \{e\}$.

(4) Go to 1.

If we have to run the algorithm on a PTM, then Step 3 can be implemented only with short *expected* time; but if we have a machine that has access to any rational biased coin, then this algorithm can run on it in bounded time. However, as [Jer86] argued, adding to a Turing machine the capacity to access all rationally biased coins violates the philosophy of Turing machines, in that not every step can be implemented with a fixed amount of hardware in fixed time. Therefore, we are interested in machines that can *simulate* any bias, using as few as possible different coins.

If we restrict ourself to rational coins, then no finite set of rationally biased coins can simulate all rational biases, since we noted in Section 2.2 that every prime must divide some bias. Therefore, we have to be less ambitious, and consider machines that can simulate all rational coins whose denominators are smaller than some fixed n. Consider a non-uniform model of computation, such as circuits, where the size of the input is fixed. We try to be efficient in the *length* of n, since then, by having polynomially (in the size of the input) many coins, any bias that arises in an algorithm, such as the above one, could be simulated efficiently. (In such an algorithm, the length of the biases that we will need to simulate will be polynomial in the size of the input.) In Section 3.2 we show that by fixing $O(\log n)$ coins any bias of denominator less than n can be simulated using $O(\log n)$ coin flips.

In Section 3.2, we show that the results of Section 3.1 are optimal by giving tight bounds on the tradeoff between the number of coins and the number of coin flips.

In Section 3.3, we show that, even if we allow irrational biases, any machine that can simulate all rational biases must have an infinite set of biased coins available to it.

3.1. *Upper Bounds for Efficient Simulation of Many Coins*

In this section, we show how, for any fixed n, one can determine in advance $O(\log n)$ coins, with which every rational bias whose denominator is less than n can be simulated by flipping $O(\log n)$ coins.

The idea of this simulation is that the set of biases that can be simulated quickly is "closed" under averaging. Assume we have a set of biased coins with which we can simulate two biases p_1 and p_2 with at most d coin flips. Flip a fair coin; if heads, simulate the bias p_1, if tails, simulate the bias p_2. This gives us a way of simulating a bias of $(p_1+p_2)/2$ with at most $d+1$ coin flips.

We can generalize this idea. Assume that we have a set of biased coins C and a sequence of biases $B=\{b_1, b_2, ..., b_n\}$ such that we can simulate any bias from B with coins from C by at most f coin flips and that we can simulate an m-sided die with at most g coin flips. Then we can simulate the average of B,

$$\frac{1}{m}\sum_{i=1}^{m} b_i,$$

with at most $f+g$ coin flips. This is done by simulating the m-sided die, and then simulating a coin of bias b_j, where j is the result of the first roll. We output the result of the second simulation.

Let k be $\lfloor \log(n!) \rfloor$. In order to show that we can simulate any coin of bias p/q such that $q \leqslant n$, it is enough to show that, for any q, we can simulate a $k \cdot q$-sided die, since we can decide that the outcome of the coin is heads or tails depending on whether the outcome of the $k \cdot q$-sided die is smaller or larger than $k \cdot p$. In order to simulate a $k \cdot q$-sided die, we know from Section 2.1 that all we need to be able to do is to simulate one flip of a coin of bias $1/kq$, and $2 \log(k \cdot q)$ flips of a fair coin.

We will show how to simulate a coin of bias $1/kq$ in three stages, each stage requiring at most $O(\log n)$ new coins. First, we will show how to simulate a coin of bias $1/2^j$, for any j such that $2^j < n!$. Then we show how to simulate a coin with bias $2^j/n!$ for any such j, and finally for any $h < n!$ we show how to simulate a coin of bias $h/kn!$. By taking h to be $n!/q$, we get a $1/kq$ coin.

We now explain how each stage works.

Stage One. For any j

$$\frac{1}{2^j} = \prod_{i \text{ s.t. } i\text{th bit of } j \text{ is } 1} \frac{1}{2^{2^i}}.$$

Therefore, given j such that $2^j < n!$, a coin of bias $1/2^j$ can be simulated by flipping a coin of bias $1/2^{2^i}$ for all i such that $2^{2^i} \leqslant n!$. We output heads iff

for every i such that the ith bit of j is a 1 we flipped a head on the coin of bias $1/2^{2^i}$. Then the probability of a head is exactly $1/2^j$.

To be able to execute this algorithm for any $j < n!$, we will need to be able to have access to all coins of bias $1/2^{2^i}$ for all i less than $\log\log(n!)$, a total of $O(\log n)$ coins.

Stage Two. Given j, a coin of bias $2^j/n$ can be simulated by simulating a coin of bias $1/2^{k-j}$ using the method of stage one and flipping a coin of bias $2^k/n!$ once. The output is heads if both coins are heads. The probability of heads is

$$\frac{1}{2^{k-j}} \cdot \frac{2^k}{n!} = \frac{2^j}{n!}.$$

For this stage, we have added only one new coin bias, namely $2^k/n!$. This coin is flipped only once and stage one is carried out only once.

Stage Three. For any $l < n!$ let b_j be $2^j/n!$ if the jth bit of l is one, and zero otherwise, for $0 \leqslant j \leqslant k$. Then the average of $\{b_1, b_2, ..., b_k\}$ is

$$\frac{1}{k} \sum_{i=1}^{k} b_i = \frac{l}{k \cdot n!}.$$

From stage two we know that we can simulate any bias in $\{b_1, b_2, ..., b_k\}$. What we need in addition is to be able to simulate a k-sided die, but this is easy, as we know from Section 2. For this stage, we need only one more coin type, a coin of bias $1/k$. We will need to flip this coin once, a fair coin $2 \log k$ times, and carry out one call to stage two.

A total of $O(\log n)$ coins are needed. Each coin, except for the fair coin, is flipped once, and the fair coin is flipped $2 \log k \cdot q + 2 \log k$ times, so we have a total of $O(\log n)$ coin flips. ($k = \lfloor \log(n!) \rfloor$, so $\log(k)$ is $O(\log n)$, and $q < n$.) Hence we have

THEOREM 4. *For any n, we can fix $O(\log n)$ coins such that for any bias p/q, $q \leqslant n$ we can simulate a p/q coin with $O(\log n)$ coin flips.*

The algorithm described above has several drawbacks, some of them unavoidable.

We tried to make our algorithm efficient in the length of n, but the length of the bias of the coins is exponential in the length of n. This is, however, unavoidable. As we noted in Section 2.2, every prime up to n must divide at least one of the biases of the coins that we fixed. There are $\Theta(n/\log n)$ primes up to n [Har72]. Therefore, since we want to have only a few coins, at least one coin must have a very large denominator. However, we are assuming that the time required to flip a coin is independent of the bias.

In this problem of simulating biased coins, we can identify at least three different kinds of complexity measures: one is the number of coins we need, the second is the number of coin flips, and the third kind is the computational complexity of the mapping from the sequence of coin flips to the output of the coin we are simulating. The algorithm we gave is efficient in the first two complexity measures, and, as we will see in the next section, is optimal in this respect. As for the third kind, we do not have an efficient algorithm to map from the sequence of coin flips to an outcome of the p/q coin. We need, in order to decide the outcome, to know what the jth bit of $n!/q$ is. We know of no algorithm efficient in the length of n to compute it, and suspect that none exists. However, for our purposes, it would be sufficient to find an algorithm that is nonuniform in n or one that works for n a power of 2. At the very least, our results are an upper bound on the lower bound one can prove for this problem.

3.2. *Lower Bound for Efficient Simultaneous Simulation*

The goal in this section is to give tight bounds on the tradeoff between the number of coins c and the number of coin flips d required to simulate all j-sided dice, for $1 \leqslant j \leqslant n$. By Section 2 this is equivalent (up to an additive factor of $O(\log n)$) to the simulation of all biases of the form $1/j$ for $1 \leqslant j \leqslant n$. The bounds we have is that c is $\Theta(\log n/\log d)$. We first show, in Lemma 3.2.1, that if you can simulate these dice, then you can simulate many different biases as well. Then, in Lemma 3.2.2, we give a counting argument, bounding the number of different biases that can be simulated by c coins in d coin flips. From this we deduce the lower bound in Theorem 5. The lower bound holds even if we allow the biases of the coins to be irrational. The upper bound is obtained by observing that the algorithm of Section 3.1 can be generalized to give the desired tradeoff.

LEMMA 3.2.1. *If there exist c coins with which any j-sided die with $1 \leqslant j \leqslant n$ can be simulated in d coin flips, then these coins can simulate at least $\binom{n/\log n}{n/2 \log n}$ different biases in $2d$ coin flips.*

Proof. For any subset of $\{1/2, 1/3, 1/4, \ldots, 1/n\}$, we can simulate a bias which is the average of that subset in $2d$ coin flips by first simulating a die with as many sides as the size of the set, using the method of Section 2, and then simulating $1/x_j$, where j is the outcome of the die roll and x_j is the jth smallest element in the subset. We wish to give a lower bound for the number of different such averages. Let P be the number of prime numbers less that n; then $P = \Theta(n/\log n)$ [Har72]. The sum of the reciprocals of any subset of these P primes will be different from the sum of the reciprocals of any other subset. This is so since when reciprocals of primes are added up, the denominator is always the product of all these primes. So the

averages of all subsets of exactly $P/2$ reciprocals of primes will all be different. The lemma follows since there are $\binom{P}{P/2}$ such subsets.

We now give a counting argument, which bounds the number of different biases that can be simulated by c coins in d coin flips.

LEMMA 3.2.2. *c coins can simulate at most $2^{dc(d+1)^c}$ different biases in d coin flips.*

Proof. Let S be the set of all possible outcomes of flipping each coin d times. It is clear that, if a certain bias b can be simulated by an algorithm using these coins with at most d coin flips, then

$$b = \sum_{t \in T} \text{probability of } t$$

for some $T \subset S$. We found the number of different probabilities for subsets of outcomes. First, we count the number of different probabilities of single outcomes of the coins. The probability of an outcome depends only on the number of times each one of the c different coins was heads. Each coin is flipped d times, and thus can come up heads between 0 and d times. Thus there are $d+1$ possibilities for the number of heads for each coin. It follows that there are at most $(d+1)^c$ different probabilities for a single outcome.

Now we bound the number of different probabilities of subsets of outcomes. The probability of a subset is determined by the number of outcomes of each distinct probability. The number of outcomes of a certain probability is certainly bounded by the total number of outcomes which is 2^{dc}. Thus the number of different probabilities is bounded by

$$(2^{dc})^{(d+1)^c} = 2^{dc(d+1)^c}.$$

Combining the last two lemmas we get the following theorem.

THEOREM 5. *If a set of c coins can simulate any j-sided die, for $1 \leqslant j \leqslant n$, using at most d coin flips, then $c = \Omega(\log n/\log d)$.*

Proof. By Lemma 3.2.1 these coins can simulate at least $\binom{n/\log n}{n/2 \log n}$ different biases in $2d$ coin flips; thus according to Lemma 3.2.2 we get that

$$\binom{n/\log n}{n/2 \log n} \leqslant 2^{2dc(2d+1)^c}$$

and the lower bound follows.

The algorithm we gave in Section 3.1 can be generalized to get a tradeoff between the number of coins we fix and the number of coin flips a simula-

tion requires. We can get any point on the $\log n/\log d$ curve. In the algorithm of Section 3.1, at stage 1, we assumed we had, for all i such that $2^{2^i} \leqslant n!$, a coin of bias $1/2^{2^i}$. If instead we take every mth such coin, (that is for every i such that $2^{2^{i \cdot m}} \leqslant n!$ we take a coin of bias $1/2^{2^{i \cdot m}}$), then we can simulate any coin of bias $1/2^{2^j}$ such that $2^{2^j} < n!$ using at most m coin flips. Therefore, we can execute stage 1 of the algorithm of Section 3.1 by having $\log n/m$ coins and in time $m \cdot \log n$. Let $c(n)$ be any function which is $O(\log n)$. By taking m to be $O(\log n/c(n))$ we get:

THEOREM 6. *For any function $c(n)$ which is $O(\log n)$ one can design an algorithm that uses $O(c(n))$ coins, and on input j, $1 \leqslant j \leqslant n$, simulates a fair j-sided die using $2^{O(\log n/c(n))}$ coin flips.*

3.3. *Necessary Conditions for the Simulation of Many Coins*

The question we consider in this section is whether there exists one coin, or some finite set of coins that can simulate all rational biases. From Section 3.2, we know that there will be no efficient way of doing so; we include this section merely to wrap up the loose ends. The problem is equivalent to being able to simulate all integer dice. We show that for any finite set of coins, there are only finitely many primes p so that a bias of $1/p$ can be simulated by these coins. We start by showing a necessary condition on the kind of biases that can be simulated by a single coin:

LEMMA 3.3.1. *A single coin with algebraic bias α cannot simulate any bias of the form $1/p$ for any prime p which does not divide the leading coefficient of M, where M is the minimal polynomial for α.*

Before we prove this lemma, we need the following technical lemma:

LEMMA 3.3.2. *Let p be any prime integer, $A = \sum_{i=0}^{n} a_i X^i$ a non-constant polynomial with integer coefficients such that p does not divide a_n, and $B = \sum_{i=0}^{m} b_i X^i$ a polynomial with integer coefficients such that p divides all the coefficients of B except for the last one, b_0; then A does not divide B in $Q[X]$, the ring of polynomials over the rationals.*

Proof. By induction on $\deg(B) - \deg(A) + 1$. For $\deg(B) - \deg(A) + 1 \leqslant 0$, the degree of A is greater than the degree of B, and since B is not the zero polynomial, A cannot divide it. Assuming the lemma holds for $\deg(B) - \deg(A) \leqslant k - 1$, we prove that it holds for $\deg(B) - \deg(A) = k$. Look at the following polynomial:

$$B'(x) = a_n B(x) - b_m A(x).$$

B' has lower degree than B; moreover A divides B if and only if A divides B'. However, since p divides b_m and p does not divide a_n, we get that B'

has all its coefficients but the last divisible by p. Thus, by the induction hypothesis, A does not divide B', so A does not divide B.

Proof (of Lemma 3.3.1). Assume, to the contrary, that we can simulate $1/p$ with a coin of bias α in d coin flips. The probability of getting a specific sequence which has exactly i heads is $\alpha^i(1-\alpha)^{(d-i)}$ which is an integer polynomial in α. Thus, the probability of getting any set of sequences is an integer polynomial in α. Let S be the set of sequences which are mapped to heads. If we simulate $1/p$ then a certain set of sequences has that probability; thus we get that

$$R_S(\alpha) = 1/p,$$

where R_S is the polynomial that gives the probability of falling in this set. This implies that

$$p \cdot R_S(\alpha) - 1 = 0.$$

Since M is the minimal polynomial of α and α is a root of $p \cdot R_S(x) - 1 = 0$, then M has to divide $p \cdot R_S(x) - 1$. But $p \cdot R_S(x) - 1$ is clearly a polynomial with all coefficients, except the last, divisible by p; and p does not divide the leading coefficient of M. Thus, by Lemma 3.3.2, M cannot divide $p \cdot R_S(x) - 1$, which is a contradiction.

We wish to generalize the last lemma, and prove that any finite set of coins cannot simulate all the integer dice:

THEOREM 7. *Any finite set of coins with biases* $\alpha_1, \alpha_2, ..., \alpha_d$ *cannot simulate an infinite number of biases of the form* $1/p$ *for prime* p.

Proof. Suppose that all the biases are algebraic. From Galois theory there exists a single algebraic number α such that

$$Q(\alpha_1, \alpha_2, ..., \alpha_d) = Q(\alpha)$$

where $Q(\alpha)$ is the extension field for α over the rationals [Wae70]. Thus, for each i, α_i is a rational polynomial in terms of α. By clearing denominators, if necessary, we can ensure that for each i, α_i is actually an integer polynomial in terms of α. In an algorithm using coins of biases $\alpha_1, \alpha_2, ..., \alpha_d$ the probability of getting each sequence is an integer polynomial in terms of $\alpha_1, \alpha_2, ..., \alpha_d$. In our case it is also an integer polynomial in terms of α. Thus, by the same reasoning as in the previous theorem, the only biases of the form $1/p$, for a prime p that can be simulated, are for primes that divide the leading coefficient of the minimal polynomial of α.

The theorem then follows for all coins, algebraic and transcendental. The reason for this is a result derived from the Hilbert Nullstellensatz which states that any set of polynomial equations with rational coefficients which has a solution over the complex numbers also has a solution over the algebraic numbers [Har77]. Given an algorithm which uses coins $c_1, ..., c_n$ to simulate a certain bias, we can express the bias simulated by the algorithm as a rational polynomial in the biases of the coins. Thus, we look at the set of equations that say that a set of biases is simulated by some fixed algorithms with a set of transcendental coins. Since that set of equations has a transcendental solution, it also has an algebraic solution. It should be noted that although we are not guaranteed that the solutions are valid biases for coins, or even real, the rest of the proof did not rely on anything but the *algebraic* properties of the biases.

4. Open Problems

As we discussed in Section 3.2, the algorithm given there does not provide a fast mapping procedure from the sequence of coin flips and the outcome of the coin. Providing such a procedure, or giving an alternative simulation algorithm that has such a procedure, is the major open problem this paper suggests.

Another area for future research would be to classify those random generation problems for which bounded time algorithms, as in our model, actually exist. In particular, does this class include precisely those problems for which there are efficient counting algorithms?

Acknowledgments

We thank Amos Fiat for his contributions to Theorem 1 and for introducing the sixth author to the problem. We had many stimulating discussions with Sampath Kannan and Ronitt Rubinfeld during the research process. Mark Gross provided us with an explanation for why transcendental biases do not help. We thank Dalit Naor and Raimund Seidel for making helpful comments over previous drafts.

Received April 16, 1988; final manuscript received April 10, 1991

References

[Eve79] Even, S. (1979), "Graph Algorithms," Comput. Sci. Press.

[Gil77] Gill, J. (1977), Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6**, 675–695.

[Har72] HARDY AND WRIGHT (1972), "Introduction to Number Theory," Academic Press, New York.

[Har77] HARTSHORNE, R. (1977), "Algebraic Geometry," Springer-Verlag, Berlin/New York.

[Jer86] JERRUM, M., VALIANT, L., AND VAZIRANI V. (1986), Random generation of combinatorial structures from a uniform, *Theoret. Comput. Sci.* **43**, 169–188.

[Kar83] KARP, R., AND LUBY, M. (1983), Monte Carlo algorithms for enumeration and reliability problems, *in* "Proceedings, 24th IEEE Symposium on Foundations of Computer Science," pp. 56–64.

[Wae70] VAN DER WAERDEN, B. L. (1970), "Algebra," Ungar, New York.